Introduzione

In questo libretto sono illustrati i quesiti di informatica che sono stati pubblicati sul sito `http://www.kangourou.it` e nel manifesto "Kangourou dell'Informatica 2009", assieme a quelli proposti nella gara svoltasi per via telematica il 31 marzo 2009.

Per ogni quesito viene commentata la soluzione, dando una spiegazione su come ottenerla, un'indicazione di possibili difficoltà o errori e un cenno piú o meno ampio al contesto in cui il quesito può essere inquadrato nell'ambito dell'informatica. Vengono anche individuate delle parole chiave che possono essere utili per ricerche in rete o per trovare connessioni tra i diversi quesiti proposti.

Il libretto si rivolge sia agli alunni, che abbiano o no partecipato alla gara, sia agli insegnanti, nell'intento di proporre qualche approfondimento e di rinnovare l'interesse e il divertimento suscitati dai quesiti e dalla gara. Naturalmente lo scopo ultimo è promuovere l'informatica come disciplina scientifica.

Indice

Parte I.

Quesiti nel volantino

Quesito 1: Trettrerobòt

Trettrerobòt è in grado di eseguire soltanto i seguenti comandi:

- `A` seguito da un numero: ordina a Trettrerobòt di fare quel numero di passi avanti;
- `S`: ordina a Trettrerobòt di fare un salto in avanti;
- `D`: ordina a Trettrerobòt di girarsi a destra di 90 gradi.

Quale sequenza di comandi verrà compresa da Trettrerobòt?

A `DA4A2D3`

B `AASDA3`

C `A1S2A4`

D `A1A1A1`

E `DDRRS`

Soluzione. La risposta corretta è la **D**. Nel quesito sono formulate tre regole, ciascuna delle quali definisce un comando. L'unica sequenza che rispetta tali regole è `A1A1A1`, corrispondente alla tripla ripetizione in sequenza del comando `A1` (che ordina a Trettrerobot di fare 1 passo in avanti). In tutte le altre risposte la sequenza non è conforme alle regole:

- nella risposta A l'ultima `D` è seguita da un numero;
- nella risposta B la prima `A` non è seguita da alcun numero (lo stesso vale per la seconda `A`);
- nella risposta C la lettera `S` è seguita da un numero;
- nella risposta E è presente la lettera `R` che non è prevista da alcuna regola.

Difficoltà e errori frequenti. Per risolvere il quesito è necessario analizzare ogni sequenza e verificare se rispetta le regole previste. Il significato delle regole non fornisce alcun aiuto nel determinare la risposta corretta.

Contesto informatico. La soluzione del quesito coinvolge una competenza tipica dell'informatica che consiste nella possibilità di descrivere oggetti e/o operazioni attraverso l'uso di *linguaggi formali*, definiti appositamente. La definizione di tali linguaggi si basa su due aspetti fondamentali:

- la *sintassi* specifica quali simboli possono essere usati e il modo in cui combinarli,
- la *semantica* fornisce il significato da dare alle *frasi* che rispettano la sintassi del linguaggio.

In particolare, nel quesito viene definito il linguaggio delle frasi riconosciute da Trettrerobòt:

- la sintassi prevede che le frasi siano formate usando soltanto le lettere `A`, `S`, `D` e i numeri `1`, `2`, `3`,...; inoltre le regole di sintassi prevedono che le frasi inizino sempre con una lettera, che la lettera `A` sia sempre seguita da un solo numero, che le lettere `S` e `D` non siano mai seguite da numeri;
- la semantica fa corrispondere ad ogni porzione di frase un movimento di Trettrerobòt: ad esempio `A3` fa procedere il robot di 3 passi, `S` fa saltare il robot in avanti, `D` lo fa ruotare a destra di 90 gradi.

Parole chiave e riferimenti: linguaggi formali; sintassi e semantica.

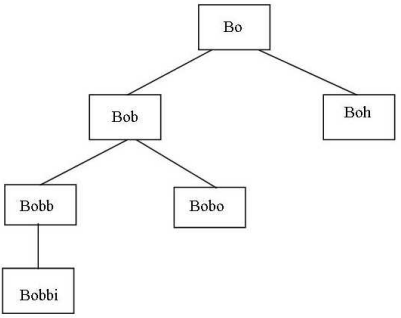
Quesito 2: Sul pianeta Alber

Sul pianeta Alber è facile stabilire le parentele: il nome di una persona è formato aggiungendo semplicemente una lettera in fondo al nome del padre. Per esempio, Carla è figlia di Carl. Chi, tra questi, è lo **zio** di Bobbi?

A	Bobb	B	Bobo	C	Bob	D	Boh	E	Carl

Soluzione. La risposta corretta è la **B**. Per stabilirlo, è opportuno... fare come i gamberi, ossia procedere all'indietro. Come si chiama il *papà* di Bobbi? Date le regole, si deve chiamare Bobb (il figlio *aggiunge* una lettera in fondo al nome, dunque il padre la *toglie*). Il *nonno* di Bobbi e padre di Bobb si chiama Bob. Lo *zio* di Bobbi è *figlio del nonno* di Bobbi e quindi si chiamerà Bob piú una lettera che non conosciamo, ma diversa dalla 'b' che identifica il padre di Bobbi, quindi l'unica possibilità tra le 5 elencate nelle risposte è Bobo (risposta B).

Difficoltà e errori frequenti. Può essere una difficoltà il dover procedere all'indietro, ma forse piú il dover ricostruire il fatto che lo zio è un figlio del nonno diverso dal padre. Come suggerito dal nome del pianeta, *Alber*, stiamo in realtà parlando di *strutture ad albero*, che traggono la loro origine proprio dagli *alberi genealogici*. L'albero genealogico di Bobbi è illustrato in figura.



Tali alberi riportavano spesso soltanto la discendenza maschile e i rami si estinguevano in assenza di figli maschi. La cosa è ovviamente irrilevante per la struttura e, se si preferisce, si può conservare invece il nome materno. Negli alberi i figli hanno un unico genitore.

Contesto informatico. Gli alberi sono strutture di dati essenziali in informatica. Ci sembra importante anche mostrare come una stessa struttura possa essere rappresentata in modi diversi, piú o meno astratti. Nel nostro caso, un *linguaggio*, ossia un insieme di parole (i 'nomi'), con opportuni vincoli (quali?), rappresenta l'albero. Naturalmente, il nonno Bob può avere i figli Bobb, Bobo, Boba, Bobi: tanti quanti i simboli disponibili nell'*alfabeto* (se l'alfabeto ha solo due lettere, allora ogni nodo dell'albero avrà al massimo due figli, in questo caso si parla di alberi *binari*).

Parole chiave e riferimenti: alberi, alberi binari, linguaggi ereditari.

Quesito 3: Euclide

Il signor Euclide gioca con due numeri interi positivi e ha inventato la seguente procedura:

1. Scrive il numero piú grande sul foglio A
2. Scrive il numero piú piccolo sul foglio B
3. Sottrae il numero in B dal numero in A e scrive il risultato su di un foglio C
4. Se in C c'è scritto 0, il risultato è il numero contenuto in A e la procedura è quindi terminata
5. Altrimenti ripete dal passo 1 con i numeri contenuti in B e in C.

Se parte dai numeri 12 e 15, qual è l'ultimo numero che sarà scritto sul foglio A alla fine della procedura?

A 0 B 15 C 3 D 9 E 6

Soluzione. La risposta corretta è la **C**.

Per rispondere al quesito occorre eseguire passo passo le *istruzioni* della procedura aggiornando man mano i numeri sui fogli A, B e C, e valutando, dopo ogni esecuzione dell'istruzione 3, se procedere con l'istruzione 4 e quindi terminare, oppure se procedere con l'istruzione 5 e quindi ripartire dall'istruzione 1. In questo secondo caso occorre confrontare i numeri scritti su B e C e copiare il piú grande su A (istruzione 1) e il piú piccolo su B (istruzione 2).

Proviamo a eseguire la procedura passo passo segnando su una tabella, istante per istante, quale istruzione viene eseguita e che numeri vengono scritti man mano su A, B e C. Quando si arriva ad eseguire l'istruzione 4, la procedura è terminata e il risultato è il numero scritto su A, cioè 3 (risposta C).

Istante	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
N. istr.	1	2	3	5	1	2	3	5	1	2	3	5
Foglio A	15	15	15	15	12	12	12	12	9	9	9	9
Foglio B		12	12	12	12	3	3	3	3	3	3	3
Foglio C			3	3	3	3	9	9	9	9	6	6

Istante	13	14	15	16	17	18	19	20
N. istr.	1	2	3	5	1	2	3	4
Foglio A	6	6	6	6	3	3	3	3
Foglio B	3	3	3	3	3	3	3	3
Foglio C	6	6	3	3	3	3	0	0

La procedura calcola il massimo comun divisore (MCD) tra due numeri utilizzando l'*algoritmo di Euclide*, ma di fatto non è assolutamente necessario saperlo per risolvere il quesito. Ovviamente se si capisce invece che il risultato calcolato dalla procedura è questo, non è necessario simularne l'esecuzione passo passo, basta calcolare il MCD tra 15 e 12 per rispondere.

Difficoltà ed errori frequenti. La procedura da eseguire non è difficile in sé, tuttavia si tratta di una procedura piuttosto astratta che comporta l'evoluzione di alcune variabili (i numeri scritti sui fogli A, B, C). Il fatto di eseguire delle istruzioni, anche facili, ma di cui non si coglie il senso, può rappresentare una difficoltà. Occorre inoltre, ogni volta che si ritorna al passo 1, aggiornare correttamente i valori in A e B utilizzando i valori in B e C.

Contesto informatico. La procedura presentata nel quesito fornisce un esempio di *algoritmo* per il calcolo del MCD. In informatica, col termine algoritmo si intende un metodo per la soluzione di un problema, descritto attraverso passi elementari. In questo caso l'algoritmo è descritto in pseudo-codice (vedi Quesito 5).

Il quesito richiede che l'esecuzione dell'algoritmo venga simulata sui dati forniti in ingresso, cioè 12 e 15, gestendo correttamente l'evoluzione di tre *variabili* (A, B e C) e lo scambio di valori tra esse, oltre all'*iterazione*, che in questo algoritmo è implementata combinando la *selezione* con il *salto*.

Parole chiave e riferimenti: algoritmo, variabile, pseudo-codice, simulazione di esecuzione di algoritmo, strutture di controllo: selezione, salto, iterazione, variabile.

Quesito 4: Analisi grammaticale

Sul pianeta Terralo le frasi sono costruite nei seguenti modi:

- `frase_semplice`: facendo seguire un `articolo` da un `nomelo`
- `frase_composta`: facendo seguire un `articolo` da un `aggettivo`, poi da un `nomelo`, poi da un `verbo` e poi da una `frase_semplice` o da una `frase_composta`

dove `nomelo` è un nome del vocabolario italiano seguito dal suffisso 'lo' e `articolo`, `aggettivo` e `verbo` sono rispettivamente un articolo, un aggettivo e un verbo del vocabolario italiano.

Quale di queste **non** può essere una frase usata su Terralo?

A il gattolo

B il bianco gattolo mangia il topolo

C il bianco gattolo

D il nero gattolo mangia il furbo topolo scappa la bucalo

E il cavolo

Soluzione. La risposta corretta è la **C**. Per controllare se una frase può essere usata su Terralo occorre controllare se rispetta la "grammatica" terralese, che prescrive che una frase sia costituita da un `articolo` e un `nomelo` e nient'altro (`frase_semplice`) oppure abbia una struttura (`frase_composta`) che è definita in termini di se stessa: infatti una `frase_composta` è una frase che a sua volta può contenere una `frase_composta`. Tale tipo di definizione è detta *ricorsiva*. Di fatto la definizione ci dice che una `frase_composta` è costituita da una o piú sequenze `articolo-aggettivo-nomelo-verbo` seguite da una `frase_semplice` (`articolo-nomelo`).

Verifichiamo la struttura grammaticale delle frasi. Le frasi A ed E sono sicuramente utilizzabili su Terralo in quanto sono costituite da un `articolo` seguito da un `nomelo` (il gatto-lo, il cavo-lo) e quindi ciascuna di esse è una `frase_semplice`. La frase B è una sequenza `articolo-aggettivo-nomelo-verbo` (il bianco gattolo mangia) seguita da una `frase_semplice` (il topolo) e quindi è una `frase_composta`. La frase D è costituita da due sequenze `articolo-aggettivo-nomelo-verbo` (il nero gattolo mangia; il furbo topolo scappa) seguite da una

`frase_semplice` (la bucalo), e quindi è anch'essa una `frase_composta`. La frase C non può essere una `frase_semplice` (contiene un `aggettivo`) né una `frase_composta` (non ha il `verbo`), e quindi è l'unica che non può essere utilizzata sul pianeta Terralo.

Difficoltà e errori frequenti. Per poter risolvere il quesito è necessario usare un approccio *analitico*: bisogna esaminare le frasi per controllare se contengono tutti gli elementi previsti dalla grammatica e nell'ordine prescritto. Analizzando sintatticamente ogni frase si ottiene la risposta al quesito.

La difficoltà dell'esercizio risiede nel fatto che la `frase_composta` è definita in modo ricorsivo, cioè facendo riferimento a se stessa, e in questo modo permette di definire un numero infinito di possibili strutture, cioè:

- nessuna sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un `articolo` e un `nomelo` (osserva che questa è una `frase_semplice` ottenuta applicando la prima regola);
- una sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un `articolo` e un `nomelo`;
- una sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un'altra sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un `articolo` e un `nomelo`;
- una sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un'altra sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un'altra sequenza `articolo-aggettivo-nomelo-verbo`, seguita da un `articolo` e un `nomelo`;
- ...

Un'ulteriore difficoltà può stare nel fatto che cavolo è sí un nome del vocabolario italiano, ma è anche un `nomelo` (cavo-lo) di quello di Terralo. Senza fare questa osservazione si può essere tentati di rispondere, erroneamente, E. Infine, osserviamo che la frase D risulta corretta sul pianeta Terralo anche se non "suona" corretta in italiano.

Contesto informatico. La soluzione del quesito coinvolge alcune competenze tipiche dell'informatica:

- la capacità di capire e utilizzare una definizione ricorsiva;
- la capacità, data una grammatica, di analizzare sintatticamente una sequenza di simboli per verificare se la sequenza appartiene al linguaggio definito dalla grammatica stessa.

Parole chiave e riferimenti: ricorsione, grammatiche e linguaggi.

Quesito 5: Pseudo-codice

Che cosa fa il seguente programma?

```
leggi a, b, c
b ← a
a ← c
c ← b
```

A Scambia il contenuto delle variabili a e b

B Confronta il contenuto delle variabili c ed a, a e b, b e c

C Legge l'input e calcola il valore delle variabili a e b

D Scambia il contenuto delle variabili a e c

E Niente di quanto scritto sopra

Soluzione. La risposta corretta è la **D**.

Questo esercizio presuppone che si abbia un'idea di cos'è un *programma* (una successione di *istruzioni*), di cosa sia una *variabile* (un "contenitore" di un dato) e anche di cosa possa significare un'istruzione come b ← a ossia *trasferire* il valore di a in b (quello che si chiama un *assegnamento*). Nel programma non ci sono *confronti* di variabili (del tipo b < a) e neppure *calcoli*, quindi le risposte B e C sono sbagliate.

La prima istruzione legge i valori di tre variabili a, b e c da qualche dispositivo di input che non ci interessa precisare. Saltiamo per un attimo alla terza istruzione a ← c: "il valore di c viene trasferito in a". Questo ci fa pensare alla risposta D. Se però il valore di a non venisse salvato da qualche parte, *prima* di eseguire questa istruzione, andrebbe perduto: ecco lo scopo della seconda istruzione b ← a, che salva appunto in b il valore di a. Per completare lo scambio, la quarta istruzione trasferisce il valore di a, salvato in b, nella variabile c.

Difficoltà ed errori frequenti. Scrivere programmi corretti non è facile. È facile invece dimenticare qualche... "piccolo dettaglio", come il fatto di aver alterato il contenuto di a, che invece vogliamo trasferire in c: il programma

```
a ← c
c ← a
```

non funziona per scambiare a e c!

Anche il programma del quesito in un certo senso è criticabile perché il valore di `b` letto in input non viene usato, anzi viene cancellato con l'istruzione `b` $\leftarrow$ `a`.

Quando si vogliono scambiare due valori numerici, funziona invece (provare per credere!) il seguente programma:

```
a ← a + c
c ← a - c
a ← a - c
```

che non usa altre variabili oltre a quelle da scambiare.

Contesto informatico. Qui abbiamo a che fare con un vero e proprio *programma*, che realizza l'operazione, elementare ma importante, di *scambio* tra i valori di due variabili. Questa operazione può servire, per esempio, per *ordinare* i due valori. Il programma è scritto in quello che si è soliti chiamare *pseudo-codice*, perché non si tratta di un vero e proprio *linguaggio di programmazione*, codificato in qualche modo preciso, così da poter essere eseguito da un elaboratore. Come abbiam detto, per esempio, l'istruzione di lettura era un po' vaga e non precisava il tipo delle variabili. Lo pseudo-codice dovrebbe essere abbastanza preciso da non lasciare dubbi sulla sua esecuzione a un lettore intelligente, e potrebbe anche comprendere frasi che spiegano operazioni ovvie per il lettore, ma che richiedono istruzioni dettagliate per la macchina. Per esempio, nello pseudo-codice una frase come "`scambia a e c`" potrebbe riassumere le tre istruzioni esaminate sopra.

Parole chiave e riferimenti: programma, istruzione, scambio, pseudo-codice, linguaggio di programmazione.

Quesito 6: Babbo Natale

Babbo Natale ha preparato dei pacchi dono di vari colori: rossi, gialli e blu, poi li ha messi nei suoi due magazzini, mescolando i colori. Ora vuole sapere quanti pacchi rossi ha preparato facendosi aiutare da alcuni folletti. Ciascuno dei folletti sa fare un'unica operazione:
Arvo sposta i pacchi blu da un magazzino all'altro;
Bjork sposta i pacchi rossi da un magazzino all'altro;
Ceula sposta i pacchi da un magazzino all'altro ed è daltonico;
Dinø conta i pacchi in un magazzino.
Babbo Natale può scegliere soltanto tre folletti. Quale di questi **non** sceglierà?

A Arvo

B Bjork

C Ceula

D Dinø

E sono necessari tutti e quattro

Soluzione. La risposta corretta è la **A**. Infatti, per contare il numero totale di pacchi rossi, basta che i folletti Bjork, Ceula e Dinø seguano questa procedura:

1. Ceula sposta tutti i pacchi dal primo magazzino al secondo;
2. Bjork sposta tutti i pacchi rossi dal secondo magazzino al primo;
3. Dinø conta i pacchi contenuti nel primo magazzino.

Verifichiamo che la procedura sia corretta: all'inizio, i pacchi saranno mescolati; dopo il passo 1, il primo magazzino sarà vuoto, mentre il secondo conterrà tutti i pacchi; dopo il passo 2, tutti i pacchi rossi saranno nel primo magazzino e non ci saranno pacchi rossi nel secondo magazzino; siamo quindi certi che il numero di pacchi contato da Dinø fornisce la risposta corretta al problema di Babbo Natale!

Osserviamo che in questa procedura il "primo magazzino" e il "secondo magazzino" possono ovviamente essere scambiati.

Difficoltà e errori frequenti. Per poter risolvere il quesito è necessario usare un approccio *costruttivo*: bisogna combinare le operazioni

fondamentali che i folletti sanno eseguire, in modo da definire una strategia, ovvero una procedura (o *algoritmo*) che permetta a Babbo Natale di risolvere il suo problema. Una volta costruita una procedura che usa solo 3 delle 4 operazioni proposte, è immediato ottenere la risposta al quesito.

La difficoltà dell'esercizio risiede nel fatto che è consentito soltanto l'uso di alcune operazioni predefinite (quelle che sanno svolgere i folletti e basta!). Questo impedisce di usare alcuni approcci che risultano forse piú naturali. Si pensi, ad esempio, alla procedura che consiste nel contare i pacchi rossi in ciascun magazzino e poi sommare i due risultati: tale metodo non è ammissibile dal momento che nessun folletto è in grado di fare la somma di due numeri!

Per questo quesito, dunque, è impossibile procedere per esclusione o facendo dei ragionamenti di carattere generale. In particolare, un errore comune potrebbe essere quello di fare delle valutazioni qualitative sulle abilità dei folletti, privilegiando quelli che sanno fare compiti piú specializzati. Questo approccio semplicistico porterebbe, ad esempio, a concludere che sia da scartare Ceula, il folletto che ignora il colore dei pacchi (in effetti, questo errore risulta essere molto frequente).

Contesto informatico. La soluzione del quesito coinvolge alcune competenze tipiche dell'informatica:

- la ricerca di soluzioni *costruttive*, realizzabili concretamente;
- la scomposizione di un compito complesso in una sequenza di passi elementari e dunque l'ideazione di un *algoritmo*;
- l'utilizzo di alcuni "mattoni" predefiniti per costruire qualcosa di nuovo.

Parole chiave e riferimenti: algoritmo, problem solving, composizione di primitive.

Quesito 7: Falegnameria

In una falegnameria vi sono 3 tipi di macchine utensili.

Tondoforatrice produce fori rotondi. Su una tavola quadrata è possibile eseguire i seguenti due tipi di operazioni:

T1 ⇒  **T2** ⇒ 

Lineografo produce una striscia sottile orizzontale

LG ⇒ 

Rotore ruota la tavola di un certo numero di gradi in senso orario (mantenendola nello stesso piano):

R45 ruota la tavola di 45°

R90 ruota la tavola di 90°

Identificare la sequenza necessaria per trasformare

A T1; LG; T1

B R45; LG; R45; T2

C T2; R90; LG

D T2; R45; LG; R45

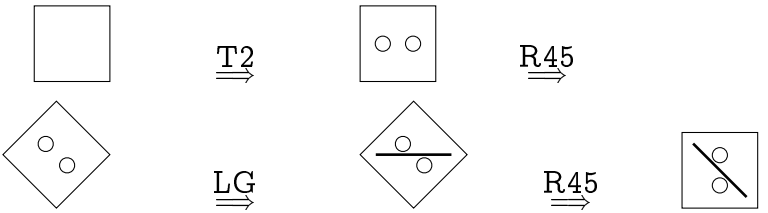
E T2; R45; LG

Soluzione. La risposta corretta è la *D*. Per ottenerla si può procedere in diversi modi.

Osservando la figura si può notare che sicurarmente l'operazione T1 non è stata applicata, questo esclude la risposta A. Sia T2 che LG sono invece necessarie. Nella tavola finale, la prima risulta ruotata di 90° e la seconda di 45°. Poiché inoltre T2 è ruotata di 45° rispetto a LG,

l'ordine in cui comporle è prima T2 e poi LG nella sequenza: T2; R45; LG; R45, cioè la D.

Verifichiamo che la sequenza sia corretta:



Alternativamente si può simulare ciascuna delle sequenze proposte nelle risposte per trovare quale produce la tavola in figura. In questo modo si potrà verificare che l'unica sequenza che produce la tavola in figura è la D e che tutte le altre sequenze producono risultati diversi. Questo secondo metodo permette di rispondere al quesito in modo piú meccanico, ma è piú lungo rispetto al primo metodo.

Si può anche utilizzare un approccio "costruttivo", cercando il modo di comporre le lavorazioni disponibili per ottenere il risultato voluto.

Difficoltà ed errori frequenti. L'approccio "analitico" alla soluzione di questo quesito, cioè il primo proposto, è sicuramente quello che permette di arrivare piú velocemente alla soluzione, ma è anche quello che presuppone una capacità di analisi che forse non tutti i ragazzi hanno sviluppato a quest'età. Occorre infatti identificare innanzitutto le lavorazioni utilizzate, poi notare che la tavola può essere ruotata in un senso solo, e infine ricavare l'ordine in cui sono state composte e l'ampiezza delle rotazioni applicate.

Contesto informatico. La soluzione del quesito coinvolge alcune competenze tipiche dell'informatica:

- scomposizione di un compito complesso in una sequenza di compiti elementari;
- simulazione e verifica di un algoritmo;
- composizione di operazioni primitive.

Parole chiave e riferimenti: simulazione, verifica, scomposizione in operazioni primitive, composizione di operazioni primitive.

Quesito 8: Tuttosò

Jimmy Tuttosò si vanta di essere un genio dell'informatica e dice frasi incomprensibili piene di termini astrusi:
1. Un virus ha infettato il mio computer anche se era spento.
2. Hai cancellato per sbaglio tutti i dati memorizzati nella scheda di rete!
3. WWW non è sinonimo di Internet.
4. Un cd contiene circa 700 megabyte.
5. Se non ricordo il nome di un file, non c'è modo di recuperare il suo contenuto.

Solo due di queste frasi in realtà sono sensate. Quali?

A	1 e 3	B	1 e 2	C	2 e 5	D	3 e 4	E	4 e 5

Soluzione. La soluzione corretta è la **D**.

Contesto informatico. Vediamo di commentare le affermazioni:

1. I normali computer sono macchine che elaborano segnali elettrici: quando sono spenti tutte le elaborazioni diventano perciò impossibili. In particolare, l'architettura più comune di un calcolatore è quella detta di *von Neumann* (pronuncia "fon noiman", dal nome del matematico ungherese e poi statunitense John von Neumann), che usa il medesimo componente funzionale per conservare i dati e le istruzioni necessarie per elaborarli. Tale componente è detto *memoria (centrale o primaria)* e al giorno d'oggi è realizzato generalmente con tecnologia elettronica, che necessita della tensione elettrica per poter funzionare. In effetti, quando la tensione è assente, nessun dato può essere conservato né elaborato, visto che anche le istruzioni dovrebbero essere conservate elettronicamente nel medesimo dispositivo. Proprio per questo motivo i computer sono dotati anche di *memorie secondarie o di massa* per conservare dati e istruzioni anche in caso di mancanza di corrente, sfruttando tecnologie diverse (tipicamente il magnetismo). Solo all'accensione, durante la cosiddetta sequenza di *bootstrap* (il termine deriva dalla locuzione inglese "to pull oneself up by one's own boot-straps", che significa sollevarsi attaccandosi alle linguette dei propri stivali, come si dice abbia fatto il Barone di Münchhausen per trarsi fuori dalle sabbie mobili) vengono caricati dalla memoria

di massa alla memoria centrale i dati e le istruzioni necessarie per le operazioni successive. Ma allora dove sono conservati i dati e le istruzioni necessarie alla sequenza di bootstrap? Sono contenuti in una terza memoria che, pure funzionalmente analoga alla memoria centrale, contiene dati e istruzioni permanenti e non facilmente modificabili (Read Only Memory, ROM).

2. Le schede di rete sono i dispositivi che permettono di scambiare dati con altri computer: dal punto di vista logico sono equivalenti ad un connettore fra il calcolatore e il mezzo trasmissivo (tipicamente un cavo, ma sempre più comuni sono le trasmissioni radio, dette "wireless", cioè senza fili) necessario alla comunicazione. In linea di principio nella scheda di rete non viene quindi memorizzato alcun dato: di fatto però le schede di rete (come quasi tutti i dispositivi complessi moderni) conservano alcuni dati e istruzioni necessari alle loro operazioni in una ROM (vedi punto precedente); questi dati e istruzioni sono elaborati da un processore dedicato che è parte della scheda stessa. L'evento che tali dati e istruzioni vadano perduti o alterati, se pure possibile, è da ritenersi estremamente raro e improbabile.

3. Con il termine *World Wide Web* (la "ragnatela con estensione mondiale"), abbreviato anche in *web* o semplicemente *WWW*, si intende un insieme di documenti conservati sui computer connessi alla rete *Internet* che fanno reciprocamente riferimento l'uno all'altro tramite collegamenti specificati secondo alcune particolari convenzioni. Ciò permette di "saltare" da un documento all'altro durante la lettura, tramite appositi collegamenti *ipertestuali*. Perché ciò possa funzionare è necessaria un'infrastruttura molto complessa: limitandoci all'essenziale possiamo descrivere il web considerando i seguenti elementi logicamente fondamentali:

   - alcuni computer in grado di scambiare informazioni fra loro (ossia collegati tramite una *rete telematica*);
   - alcuni documenti scritti rispettando alcune convenzioni particolari, in particolare essi conterranno *hyperlink* (o *collegamenti ipertestuali*): il documento A sulla macchina X potrebbe contenere un riferimento al documento B sulla macchina Y;
   - programmi per visualizzare i documenti (*web browser*), in grado fra l'altro di riconoscere gli hyperlink e richiedere nel

modo opportuno un documento quando si trova conservato su una macchina diversa da quella in uso;
- programmi (*web server*) in grado di rispondere alle richieste dei web browser, trasferendo tramite la rete documenti presenti sulla macchina in uso.

Internet quindi definisce la tecnologia con la quale sono realizzati i collegamenti di rete, mentre il web è costituito dalle modalità con le quali le informazioni contenute in una moltitudine di documenti sono collegate. Sarebbe possibile realizzare un web anche con *protocolli* di rete diversi da quelli che costituiscono Internet, anche se probabilmente sarebbe molto più difficile, almeno all'inizio, averne uno con l'estensione mondiale che esso ha grazie alla popolarità di Internet. E, viceversa, è possibile usare Internet per altre attività diverse dal collegamento ipertestuale di documenti: per esempio per scambiare messaggi di *posta elettronica* (che magari poi possono essere utilizzati come documenti del web) o per giocare a scacchi con amici lontani.

4. I Compact Disc (CD) sono memorie di massa (vedi sopra) ottiche: in pratica la memorizzazione è ottenuta con microscopici buchi in una sottile pellicola metallica che viene esaminata tramite un raggio laser. Per questo motivo essi sono generalmente "a sola lettura", ossia una volta scritti (bucati) non è più possibile ripristinare lo stato iniziale: si parla dunque di CD-ROM ("Read Only Memory" significa "memoria a sola lettura"). Inizialmente nati per conservare informazione musicale digitalizzata in uno specifico formato riconosciuto dai "CD player", oggi vengono usati per conservare qualsiasi tipo di dati, organizzati secondo una struttura a *file system* (vedi sotto) analoga a quella delle altre memorie di massa. Vengono commercializzati in tre formati, secondo la seguente tabella:

	diametro	capienza	capacità audio
Standard size	12 cm	650–703 MB	74–80 min
Mini-CD	8 cm	185–210 MB	21–24 min
"Biglietto da visita"		$\sim$ 55 MB	$\sim$ 6 min

L'unità di misura utilizzata generalmente è il Byte (B) = 8 bit (b). Qui sorge un piccolo problema, perché gli informatici trovano generalmente utile ragionare con i multipli di 2, piuttosto che con quelli di 10 com'è comune nelle altre discipline: ciò è dovuto al fatto che il più delle volte si ha che fare con cifre binarie, ciascu-

na delle quali rappresenta un bit d'informazione, ossia distingue fra due stati possibili. I prefissi usati normalmente per le unità di misura (kilo, mega, giga, tera; rispettivamente mille, milioni, miliardi e migliaia di miliardi) sono però basati sulle potenze di 10: un kilometro significa 10^3 metri. Questi prefissi sono abitualmente usati anche in ambito informatico, dove però spesso con un megabit si intende 1024 bit (ossia 2^{10} bit). Per evitare confusioni esistono perciò appositi prefissi, da utilizzare quando occorre una precisione assoluta: *kibi* (Ki = 2^{10}), *mebi* (Mi = 2^{20}), *gibi* (Gi = 2^{30}), *tebi* (Ti = 2^{40}). Attenzione: se qualcuno vi vende un disco da 1TB e voi capite 1TiB la differenza potrebbe deludervi, infatti

$$1\ \text{TB} = 1.000.000.000.000\ \text{byte}$$
$$1\ \text{TiB} = 1.099.511.627.776\ \text{byte}$$
$$1\ \text{TiB} - 1\ \text{TB} = 92{,}68\ \text{GiB}$$

5. Un *file* (pronuncia "fail") è un insieme di dati permanenti caratterizzato da un nome e altre informazioni che permettono di organizzare un intero *file system*, ossia una moltitudine ordinata di file. Generalmente la struttura del file system è di tipo gerarchico e consente di reperire un file tramite un *percorso* logico da un insieme generale ad uno più specifico del tipo: tutti i miei file / i file che riguardano la nonna / le ricette / la ricetta della torta di mele. Ciò presuppone di conoscere il nome o almeno la classificazione logica del file. Nulla vieta però di avere programmi che ci aiutano nell'identificazione del file che ci interessa, per esempio andando a cercare fra i dati contenuti nei file (es: c'è un file che contiene la parola "mele"?) o fra altri dati ausiliari (es: c'è un file che è stato modificato il 25 dicembre 2008?). Al momento i programmi più comuni per svolgere ricerche di questo tipo sono limitati all'informazione testuale, ma in futuro potrebbe essere possibile cercare tutte le fotografie che contengono una mela, magari fornendone una d'esempio.

Parole chiave e riferimenti: macchina di von Neumann, Internet, World Wide Web, CD-ROM, bit, byte, unità di misura binarie, file, file system.

Quesito 9: Sicurezza informatica

Filippo deve scegliere una password per proteggere la propria posta elettronica. Quale di queste strategie è quella che promette una maggiore sicurezza?

A `filippo1995`, come la sua data di nascita

B `f1l1pp0`, cambiando in numeri alcune lettere del suo nome

C `Filipp0`, mettendo maiuscola la lettera finale

D `filippopostaelettronica`, per distinguerla dalle altre password che già usa

E `Qpelpchp!`, come nella frase "Questa password è la prima che ho pensato!"

Soluzione. La soluzione corretta è la **E**.

Contesto informatico. La bontà di una password (parola d'ordine) dipende sostanzialmente da due fattori: la lunghezza massima prevista dal sistema e la difficoltà che essa possa essere indovinata facendo riferimento ad informazioni esterne. Immaginate un lucchetto che si apre con una combinazione di 4 levette che possono assumere ciascuna 10 posizioni. In questo caso il numero di combinazioni possibili è $10 \cdot 10 \cdot 10 \cdot 10 = 10\,000$. Supponendo che per provare (a mano) ciascuna delle combinazioni servano 10 secondi, una ricerca esaustiva nel caso pessimo impiegherebbe 100 000 s, quasi trenta ore di lavoro: in media è probabile che troveremo la combinazione in poco piú di una mezza giornata. Il tempo necessario diventerebbe però radicalmente minore nel caso sapessimo che chi ha chiuso il lucchetto ha usato il suo anno di nascita, perfino senza conoscere perfettamente i suo dati anagrafici! Perché una combinazione assicuri una buona protezione dovrà quindi essere scelta in maniera totalmente casuale fra le 10 000 possibili. In questo caso si presenta un ulteriore problema: una password generata in maniera casuale oltre ad essere difficile da indovinare è anche difficile da ricordare. Scriverla sarebbe un rimedio peggiore del male, però, perché a questo punto per essere tranquilli bisognerebbe poi essere sicuri di

proteggere anche la nota, cadendo cosí in un circolo vizioso. La soluzione migliore è quindi quella di avere una qualche tecnica mnemonica per ricordare una password di tipo sostanzialmente casuale. Ma attenzione! La tecnica mnemonica deve essere il piú originale possibile: quelle descritte alle risposte B,C e D sono assai comuni e quindi sfruttate anche da chi cerca di indovinare!

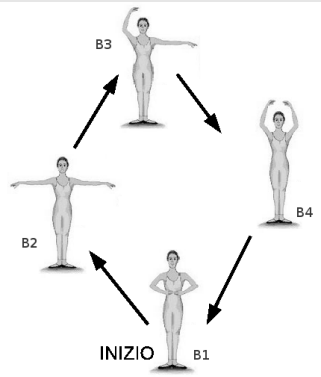
Quando le password da ricordare sono tante (p.es. per la posta elettronica, la banca *on-line*, la *chat*, ...), adottarne una unica, uguale per ogni servizio, non è una buona strategia. Infatti anche se la password scelta è ottima, essa diventa particolarmente critica: qualora fosse scoperta, *tutti* i servizi risulterebbero compromessi. La situazione può essere migliorata gestendo due livelli di sicurezza differenti: un livello base e un livello "ultra-sicuro" nel quale si adottano maggiori protezioni (per esempio scegliendo una password molto lunga):

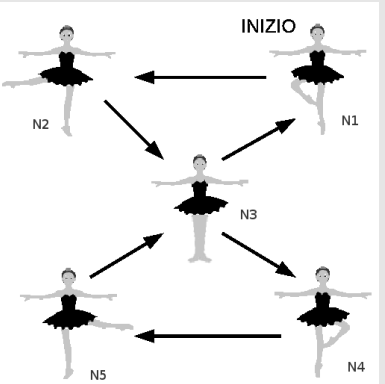
- per ogni informazione/servizio da proteggere si sceglie una password casuale di livello base;
- le varie password casuali vengono conservate proteggendole con una nuova password di livello "ultra-sicuro".

Parole chiave e riferimenti: password, sicurezza informatica, generazione casuale

Quesito 10: Surlepunt

La coreografa Madame Surlepunt ha insegnato alle sue ballerine alcuni passi da eseguire in sequenza. Bianca si muoverà secondo quando descritto dal diagramma di sinistra; Nerina secondo quanto descritto dal diagramma di destra.





Se le due ballerine eseguono ciascuna una delle loro sequenze, muovendosi a ritmo di musica e partendo nello stesso istante, dopo quanti passi si troveranno sicuramente nella stessa posizione?

A 8 passi

B 5 passi

C 2 passi

D mai

E sempre

Soluzione. La risposta corretta è la **B**. Per spiegare perché, cominciamo con l'interpretare i due diagrammi. Essi sono composti da alcune immagini, rappresentanti determinate *posizioni* del corpo, e da frecce che rappresentano lo *spostamento* da una posizione del corpo ad un'altra. Per semplicità chiameremo *Bianca* la ballerina rappresentata nel diagramma di sinistra e *Nerina* quella nel diagramma di destra. Inoltre aggiungiamo delle etichette alle varie posizioni del corpo: per Bianca

useremo etichette del tipo B1, B2, ... per Nerina useremo etichette N1, N2, ...

Solo alcune posizioni sono consentite (quelle illustrate nelle immagini), ad esempio Bianca non avrà mai le gambe aperte. In particolare, si vede che c'è un'unica posizione comune alle due ballerine, caratterizzata da gambe chiuse e braccia in fuori: per Bianca si tratta della posizione B2; per Nerina si tratta della posizione N3. Le altre posizioni si differenziano tutte per la posizione di braccia (Bianca) o gambe (Nerina). Questa osservazione basta ad escludere le risposte A e C: infatti dopo 2 o 8 passi Bianca si trova sempre in una posizione diversa da B2, in cui Nerina non potrà mai trovarsi.

Gli spostamenti sono vincolati, infatti sono consentiti solo gli spostamenti indicati dalle frecce. Osserviamo che gli spostamenti di Bianca sono definiti in modo unico da ogni posizione in cui essa si trovi; per Nerina questo invece non è vero, infatti dalla posizione N3 essa può decidere se spostarsi nella posizione N1 oppure nella posizione N4. Tuttavia, ciò non ha importanza per rispondere alla domanda, perché in ogni caso, dopo tre passi, Nerina ritornerà nella posizione N3.

	Bianca	Nerina
1	B2	N2
2	B3	N3
3	B4	N1 oppure N4
4	B1	N2 oppure N5
5	B2	N3
6	B3	N1 oppure N4
7	B4	N2 oppure N5
8	B1	N3
9	B2	N1 oppure N4
10	B3	N2 oppure N5
11	B4	N3
...	...	...

Per rispondere al quesito, bisogna procedere un passo alla volta, osservando in quale posizione possono trovarsi le due ballerine ad ogni passo. Bisogna quindi verificare dopo quanti passi le due ballerine si trovano nella stessa posizione (Bianca in B2 e Nerina in N3).

Dalla tabella a fianco, che riporta le posizioni ad ogni passo, si vede chiaramente che la risposta corretta è la B.

Osservate inoltre che, dopo il quinto passo, Bianca si ritroverà nella posizione B2 ogni 4 passi, mentre Nerina si ritroverà nella posizione N3 ogni 3 passi: pertanto, ogni 12 passi (il minimo comune multiplo tra 4 e 3) le posizione saranno le stesse (quindi dopo il passo 17, dopo il passo 29, eccetera).

Difficoltà e errori frequenti. Nel testo di questo quesito, i diagrammi forniscono una descrizione precisa di quali sequenze sono consentite

a ciascuna ballerina e quali no. Per risolvere il quesito, è necessario interpretare i diagrammi simulando passo passo gli spostamenti delle ballerine.

Una difficoltà aggiuntiva è data dal fatto che non si ha un solo diagramma, ma due; questo costringe ad eseguire in parallelo due sequenze (una per ogni diagramma) verificando la posizione raggiunta ad ogni passo. Infine, un'ulteriore difficoltà è data dal fatto che, nel caso di Nerina, la sequenza non è definita in modo univoco ad ogni passo (come è invece per Bianca).

Contesto informatico. Il quesito è basato su un classico argomento di informatica teorica, gli *automi a stati finiti* (nel seguito ASF).

Gli ASF sono definiti da un insieme finito di possibili *stati* e da un insieme di *transizioni* che permettono di passare da uno stato all'altro. Gli ASF sono usualmente rappresentati con diagrammi chiamati *grafi*, formati da: *nodi*, che rappresentano gli stati, e *frecce*, che rappresentano le transizioni da uno stato all'altro. In un automa, si chiama *percorso* una sequenza di nodi collegati da frecce consecutive.

Nel quesito sono in realtà definiti due automi, uno per ciascuna ballerina. In entrambi i casi, gli stati sono dati dalle posizioni delle ballerine e le transizioni sono date dagli spostamenti da una posizione all'altra.

Per risolvere il quesito è necessario seguire *in parallelo* i percorsi definiti dai grafi dei due automi, ovvero costruire due percorsi, uno per ciascun automa, eseguendo ad ogni passo una sola transizione per automa.

Un ASF si dice *deterministico* quando, dato un qualunque stato, è definita al massimo una transizione verso un nuovo stato. In caso contrario, l'automa si dice *non deterministico*. Nel nostro caso, l'automa che descrive i movimenti di Bianca è deterministico, mentre quello che descrive Nerina non lo è (infatti dalla posizione N3 può spostarsi sia nella posizione N1 che nella posizione N4).

Parole chiave e riferimenti: automi a stati finiti, grafi, esecuzione in parallelo, automi deterministici e non.

Parte II.

Gara del 31 marzo 2009

Quesito 11: Scarabeo

Bruna e Dino stanno giocando a Scarabeo e hanno formato le 5 parole

cane pesca carpe sopra fosso

usando le 9 lettere

a, c, e, f, n, o, p, r, s.

Bruna osserva che per usare le 9 lettere elencate non è necessario prendere tutte e 5 le parole. Quali parole può scegliere per usare tutte le 9 lettere con meno parole possibili?

Soluzione. Ci sono due possibile risposte corrette: **cane-carpe-fosso** oppure **cane-sopra-fosso**

Per trovare la soluzione, si può procedere considerando le lettere rare, cioè quelle che compaiono in poche parole. Nel nostro caso, la 'f' compare solo in "fosso" e quindi questa parola dovrà comparire in ogni soluzione. La stessa cosa si può dire per la 'n' di "cane". Se cancelliamo le lettere presenti in "cane + fosso", restano solo 'p' e 'r', che possono essere ottenute sia con "carpe" sia con "sopra". Abbiamo quindi le due risposte minime **cane carpe fosso** e **cane sopra fosso**: una soluzione minima non deve per forza essere unica!

Contesto informatico. *Scarabeo* è la versione italiana del gioco *Scrabble* ("scarabocchio"), in cui si devono incrociare parole formate da lettere con un diverso punteggio: chi fa più punti vince. Il gioco pone un *problema di ottimizzazione*: tutte le parole permesse dalle lettere disponibili sono *soluzioni*, ma non tutte le soluzioni hanno lo stesso *valore*!

Nei problemi di ottimizzazione cerchiamo dunque soluzioni *ottime* (ossia le migliori possibili). Quando il parametro che prendiamo in considerazione è però un *costo* (come nel Quesito 15."Città d'Europa"), il problema di ottimizzazione consiste nel trovare una soluzione di costo *minimo*... Proprio come in questa prova, in cui era richiesto il minimo numero di parole che contenessero tutte le lettere; si osservi che non si può arrivare a stabilire tale numero senza trovare anche le parole che lo determinano!

Potremmo chiederci se esiste un metodo generale per risolvere rapidamente i problemi di ottimizzazione. Un metodo interessante si basa

sulla *strategia golosa* a cui accenneremo nei Quesiti 15. "Città d'Europa" e 12. "Archivio multimediale": scegliamo di volta in volta l'elemento che ci piace di più. Purtroppo però questa strategia funziona per certi *sistemi d'indipendenza* (vedi a proposito del Quesito 14. "Hitori") ma non sempre.

È estremamente improbabile che esista un metodo generale per risolvere il nostro problema delle parole. Se pensiamo alle parole come a insiemi di lettere, il problema di trovare il minimo numero di parole che contengono un determinato insieme di lettere risulta appartenere a una famigerata (e assai numerosa) classe di problemi (i *problemi NP-completi*) per i quali nessuno ha mai trovato soluzioni *efficienti* (cioè rapide e utilizzabili praticamente anche per problemi di grandi dimensioni). Ci sono ottimi motivi per pensare che questi metodi generali non esistano, anche perché si può dimostrare che, se si scoprisse un metodo efficiente per risolvere uno solo di tali problemi, allora li si potrebbe risolvere *tutti* in modo efficiente! Si tratta di una delle "questioni aperte" o sfide più significative e più stimolanti dell'informatica teorica, e costituisce uno dei cosiddetti "problemi per il nuovo millennio"!

Parole chiave e riferimenti: problemi di ottimizzazione, problemi NP-completi, sistemi d'indipendenza, algoritmi golosi.

Quesito 12: Archivio multimediale

Little Jimi vuole archiviare su chiavette USB i video dei suoi saggi di chitarra. Ogni chiavetta ha una capacità di 1 GB. I video hanno le seguenti dimensioni:

320, 800, 450, 221, 257, 132, 562, 307 MB.

Ricordando che 1 GB è pari a 2^{10} MB, come deve raggruppare i video sulle chiavette USB in modo da utilizzarne il minor numero possibile?

Soluzione. La soluzione ottima richiede tre chiavette, e si può scegliere tra due alternative:

- $800 + 221 = 1021$ (MB)
- $562 + 450 = 1012$ (MB)
- $320 + 307 + 257 + 132 = 1016$ (MB)

oppure:

- $800 + 221 = 1021$ (MB)
- $450 + 307 + 257 = 1014$ (MB)
- $562 + 320 + 132 = 1014$ (MB)

Contesto informatico. Dal punto di vista informatico, si tratta di un'istanza di un altro problema della famigerata classe degli "NP-completi" (vedi anche i Quesiti 11."Scarabeo" e 15."Città d'Europa"). Come nel caso delle "Città d'Europa", anche qui abbiamo a che fare con un *problema di ottimizzazione*, conosciuto come *problema dell'imballaggio* (o *bin packing*) e che in generale può essere formulato nel seguente modo. Data una sequenza di numeri che rappresentano ad esempio pesi (o altri tipi di misure, nel nostro caso si tratta di dimensioni di spazio in memoria) e dato un numero che rappresenta la capacità massima di un certo tipo di contenitore, il compito consiste nel ripartire i pesi nel minimo numero di contenitori. Il problema di ottimizzazione dell'imballaggio si presenta spesso nella realtà, in una varietà di forme: quello proposto è già un caso significativo; si pensi poi, come ulteriore esempio, al compito di tagliare una serie di tubi di varie lunghezze da un numero minimo di tubi di lunghezza standard. Nella pratica, dato che procedimenti efficienti non se ne conoscono (ammesso che ve ne siano), si può ricorrere a una *strategia sub-ottima*, realizzata mediante questo algoritmo "goloso": ordinare i pesi dal più grande al più piccolo e poi

assegnarli –in quest'ordine– ciascuno al primo contenitore (di una fila di contenitori lunga quanto si vuole) che sia sufficiente a contenerlo. È stato dimostrato (Johnson, 1973) che nessun'altra strategia efficiente (non ottima) può garantire (nel caso peggiore) un margine d'errore inferiore rispetto a questa... Nel caso in esame, la procedura sub-ottima riempie esattamente tre chiavette e fornisce la prima delle due alternative scritte sopra: quindi dà un risultato ottimo. Ma se, ad esempio, si avesse un video da 329 MB al posto di quello da 320, allora la procedura sub-ottima darebbe:

- $800 + 221 = 1021$ (MB)
- $562 + 450 = 1012$ (MB)
- $329 + 307 + 257 = 893$ (MB)
- 132 (MB)

con l'impiego di una chiavetta in più del necessario, che come si può verificare è sempre tre:

- $800 + 221 = 1021$ (MB)
- $562 + 329 + 132 = 1023$ (MB)
- $450 + 307 + 257 = 1014$ (MB)

Parole chiave e riferimenti: problema di ottimizzazione, procedura sub-ottima, algoritmi golosi, bin packing.

Quesito 13: Ping pong

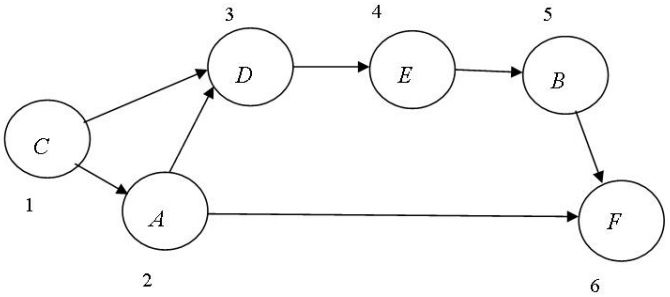
Sei giocatori disputano un torneo di ping pong stabilendo (arbitrariamente!) che se A batte B e B batte C, allora A è più bravo di C e lo precederà nella classifica finale.
I risultati delle partite disputate sono i seguenti: Bruna batte Franco, Carla batte Aldo, Dino batte Enzo, Aldo batte Franco, Carla batte Dino, Enzo batte Bruna e infine Aldo batte Dino.
Quale sarà la classifica finale?

Soluzione. La classifica finale è la seguente: **1. Carla, 2. Aldo, 3. Dino, 4. Enzo, 5. Bruna, 6. Franco.**

Carla è di sicuro la vincitrice perché non ha mai perso; similmente, Franco è l'ultimo classificato perché non ha vinto nessuna partita. Restano da ordinare gli altri 4 giocatori: poiché Aldo batte Dino, che a sua volta batte Enzo, che a sua volta batte Bruna, la classifica è fatta!

Difficoltà e errori frequenti. Individuare da subito il primo e l'ultimo classificato rende praticamente immediata la soluzione del problema!

Un approccio generale per risolvere questo tipo di problemi consiste nel disegnare un *grafo delle precedenze* che rappresenti le informazioni disponibili:



Disegniamo sei *nodi*, all'interno di ciascuno scriviamo il nome (è sufficiente la lettera iniziale) di uno dei giocatori, e poi tracciamo un *arco orientato* per ogni partita disputata: se Aldo ha battuto Dino, tracciamo un arco dal nodo A al nodo D con una freccia verso il nodo D, e così via.

Iniziamo a numerare i nodi da 1, partendo da un nodo al quale non arrivano archi: nel nostro caso, il solo nodo C. Poi proseguiamo la

numerazione considerando ogni volta un nodo i cui "predecessori" (cioè i nodi dai quali esce un arco che arriva al nodo considerato) sono stati già tutti numerati: così facendo, nel nostro caso si numerano nell'ordine: A, D, E, B, F. Si ottiene così la classifica finale.

Nel caso in questione, bastano 7 partite a determinare completamente la classifica, ma in generale non è detto che questo succeda. Vediamo ad esempio queste varianti.

1. Nell'ultima partita, invece di giocare Aldo e Dino, giocano Enzo e Franco e vince Franco: in questo caso non è possibile compilare alcuna classifica, perché avremmo una contraddizione, infatti Franco è più bravo di Enzo (dal risultato dell'ultima partita), ma per la nostra regola anche Enzo è più bravo di Franco (infatti Enzo batte Bruna, la quale batte Franco)... è per questo che la regola si usa quando si fanno i tornei ad eliminazione e non nei campionati "all'italiana" (nei quali invece occorre computare e confrontare il numero di vittorie di ciascun giocatore)!

2. Nell'ultima partita vince Dino anziché Aldo: in questo caso non siamo in grado di dire chi tra Aldo e Enzo è più bravo, bisognerebbe ad esempio farli gareggiare fra loro per stabilirlo...

Contesto informatico. Un problema analogo a quello del quesito si ha ad esempio quando si devono svolgere delle attività che dipendono l'una dall'altra: in che ordine temporale è opportuno programmarle? Non ci avrete mai fatto caso, ma risolvete un problema analogo a questo tutte le mattine, prima di vestirvi: le scarpe vanno messe dopo i pantaloni e i calzini, la camicia va prima del maglione, e le mutande?

In informatica, questo tipo di problema va sotto il nome di *ordinamento topologico* e in genere viene formalizzato facendo riferimento a *grafi orientati* (vedi Quesito 10."Surlepunt"). I nodi del grafo sono gli oggetti da ordinare (in questo caso, i giocatori) e si ha una freccia da A a B se A batte B in una partita. Si cerca un ordinamento totale, in cui cioè ogni nodo ha una sua posizione precisa rispetto a tutti gli altri.

La regola con cui si stabilisce che "se A batte B e B batte C, allora A è più bravo di C e lo precederà nella classifica finale" prende il nome di *transitività*. Nel grafo la transitività corrisponde alla combinazione di più frecce consecutive lungo dei *cammini*: dati due nodi X e Y, possiamo dire che X è pi'u bravo di Y se c'è un cammino che conduce da X a Y.

Avete notato che nel grafo dell'esempio non ci sono *cicli* (ovvero cammini che partono e finiscono nello stesso nodo)? Se ci fossero non sarebbe

possibile stabilire un ordinamento totale: è quello che succederebbe ad esempio nel caso della variante 1.

Se invece il grafo è *aciclico*, cioè senza cicli, come nel quesito, allora è possibile sempre stabilire un ordinamento totale che rispetta le frecce del grafo. Non è detto però che questo si possa fare in modo unico. È quello che succederebbe ad esempio nel caso della variante 2, in cui sarebbero accettabili due classifiche diverse:

1. Carla, 2. Dino, 3. Aldo, 4. Enzo, 5. Bruna, 6. Franco
oppure
1. Carla, 2. Dino, 3. Enzo, 4. Aldo, 5. Bruna, 6. Franco.

e per sciogliere il dubbio bisognerebbe far gareggiare fra loro Aldo e Enzo (ovvero aggiungere al grafo una freccia tra loro).

Insomma, nel caso di grafi orientati aciclici, è possibile stabilire un unico ordinamento solo se esiste un cammino che passa una sola volta da ciascun nodo.

Vale la pena osservare che, nel particolare problema considerato, il numero *minimo* di partite che potrebbero farci stilare una classifica (se siamo fortunati, ossia nel *caso migliore*) è 5: per giungere alla stessa classifica finale, i cinque risultati devono essere C batte A, A batte D, D batte E, E batte B, B batte F.

Ma qual è invece il numero minimo di partite che è necessario giocare per giungere alla (stessa) classifica finale, nel caso piú sfortunato (o *caso peggiore*), pur usando la strategia migliore possibile?

Nel nostro caso, con sei giocatori, questo numero è 10. Nel primo turno si disputano tre partite con giocatori tutti diversi: per esempio, come nel testo del quesito, Bruna batte Franco, Carla batte Aldo, Dino batte Enzo. A questo punto si deve determinare l'ordine dei tre perdenti: Aldo batte Franco, poi Aldo batte anche Enzo e quindi Enzo batte Franco. Dopo queste sei partite abbiamo una classifica parziale che vede in testa Carla seguita nell'ordine da Aldo, Enzo e Franco. Ora inseriamo chi ha battuto l'ultimo nella classifica parziale: Bruna ha battuto Franco al primo turno, quindi lo precederà nella classifica finale. Ma come si piazza Bruna rispetto ai primi tre della classifica parziale? Facciamola giocare con quello a metà classifica: se è piú brava giocherà poi con un giocatore piú forte, se meno con uno piú debole. Quindi Aldo batte Bruna e poi anche Enzo batte Bruna. Ci resta da collocare Dino che, avendo battuto Enzo, attualmente terzo, deve vedersela con i primi due. Carla batte Dino e infine anche Aldo batte Dino, dandoci la classifica finale, quella che abbiamo visto, dopo 10 incontri.

Il modo di procedere per disputare il minimo numero possibile di incontri anche nel caso peggiore è piuttosto macchinoso già per sei giocatori. Sorprendentemente, nessuno sa come minimizzare il numero di incontri nel caso generale di un numero n qualsiasi di giocatori! Peccato, perché sarebbe anche il minimo numero di confronti necessari in generale per ordinare n elementi, e le operazioni di ordinamento sono, come sappiamo, assai comuni. C'è ancora molto da scoprire in informatica (e non solo)!

Parole chiave e riferimenti: grafo delle precedenze, cammini e cicli, transitività, grafi orientati aciclici, ordinamento topologico, caso migliore e caso peggiore.

Quesito 14: Hitori

Gli abitanti della città giapponese di Hitori hanno deciso di pavimentare la piazzetta quadrata del paese con grandi mattonelle con le prime 5 lettere dell'alfabeto, che gli operai hanno disposto a caso. A molti non piace però che ci siano lettere uguali lungo le righe e le colonne.

D	A	E	B	C
D	E	A	D	C
A	C	E	E	E
B	D	C	B	A
C	E	E	A	B

Per fortuna il sindaco scopre che si possono coprire alcune mattonelle con delle fioriere in modo che:
1. non ci siano piú lettere uguali in nessuna riga o colonna
2. due fioriere non siano attaccate lungo un lato (possono avere al piú un angolo in comune)
3. sia possibile andare da una mattonella scoperta a ogni altra, senza scavalcare fioriere.

Indicate le mattonelle da coprire con le fioriere.

Soluzione. Hitori (parola giapponese che significa, piú o meno, solitario) è un tipo di rompicapo logico pubblicato per la prima volta nella rivista "Puzzle Communication Nikoli" nel marzo 1990. Kangourou Italia ha appena pubblicato un libro sull'Hitori nella collana "Biblioteca di Xla Tangente", a cui rimandiamo per le regole e le tecniche risolutive, oltre che per piú di 160 esempi di varia difficoltà.

La griglia proposta si risolve facilmente cominciando dalle tre "E" sulla riga centrale.

D	A	❀	B	❀
❀	E	A	D	C
A	C	❀	E	❀
B	D	C	❀	A
C	❀	E	A	B

Difficoltà ed errori frequenti. L'errore piú comune è quello di procedere a caso, il che comporta, a meno di essere molto fortunati, di dover cancellare le soluzioni errate e riprendere da un punto che si ritiene corretto. Questo non è di per sé sbagliato, e può essere l'unico modo di risolvere puzzle particolarmente difficili, ma è certamente dispendioso e sconsigliabile se si riescono viceversa a individuare delle tecniche sicure che fungono da scorciatoie.

Contesto informatico Hitori è un gioco non banale che, oltre a introdurre la consueta griglia in cui non si vogliono avere ripetizioni dei simboli (numeri o lettere) lungo le righe e le colonne, richiede che le caselle non annerite siano *connesse*, un concetto ben noto nella *teoria dei grafi*. Come spiegato nel testo della prova, la connessione significa semplicemente il poter andare da una casella a qualsiasi altra, attraversando caselle vicine (ossia con un lato in comune), senza attraversare caselle annerite (o occupate da fioriere).

La connessione è ovviamente importante in Internet, e la rete Internet può essere vista come un enorme *grafo connesso*.

L'insieme delle caselle annerite in una griglia Hitori costituisce quello che si chiama un *insieme indipendente* nel grafo, ossia un insieme di caselle a due a due non adiacenti. Questo insieme ha una proprietà interessante: se ne considero un sottoinsieme, ossia tolgo qualche casella annerita, le caselle restanti rimangono ovviamente non adiacenti. Se annerisco meno caselle, inoltre, quelle non annerite saranno ancora connesse: sarà piú facile andare da una casella a qualsiasi altra. Se considero, per esempio, tutti i quadrati di 5x5 caselle e tutti i possibili insiemi di caselle annerite che lasciano connesso l'insieme delle caselle non annerite, ho un esempio di quello che si chiama un *sistema d'indipendenza* (o, se volete usare un termine ancora piú... spettacolare, un *complesso simpliciale*).

Un sistema d'indipendenza è una famiglia di insiemi che hanno appunto la proprietà che, preso un qualunque insieme della famiglia, anche tutti i suoi sottoinsiemi appartengono alla famiglia.

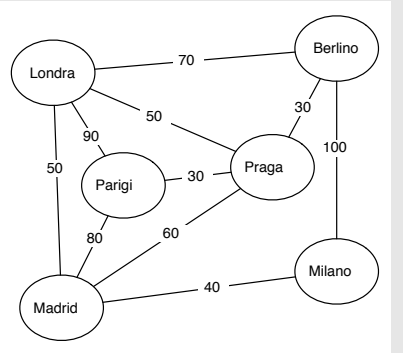
Se questo concetto vi risulta difficile, vediamone uno... piú difficile (per certi versi), ma di fatto facile da capire. Quante caselle possiamo annerire al massimo in un Hitori 5x5? Possiamo provare a caso, annerendo una casella qui e una là senza violare le regole. Potremmo ottenere l'esempio della prova, in cui ci sono 7 caselle annerite (o mattonelle con fioriere), e non se ne può aggiungere nessun'altra (provate per esempio a coprire una delle tre "B"). Se invece annerite le caselle ai bordi e quella in mezzo in una riga, una riga sí e una no, riuscite ad annerire 9 caselle.

Questo dimostra almeno che... essere golosi non è sempre la strategia migliore: annerendo a caso quante piú caselle possibile non sempre si ottiene il massimo. Potreste facilmente mostrare che, in tutti i quadrati con un numero dispari $2k-1$ di caselle su un lato, si possono annerire almeno k^2 caselle. Ma in generale non è il massimo...

Parole chiave e riferimenti teoria dei grafi, connessione, sistemi d'indipendenza, algoritmi golosi.

Quesito 15: Città d'Europa

Sveva e Italo hanno amici in 5 città europee. Partono in aereo da Milano e vogliono passare per le 5 città, spendendo il meno possibile (e contando su un passaggio degli amici per il ritorno dall'ultima città visitata a Milano). Sulla mappa sono segnate le città che vogliono visitare con la spesa in euro per ciascun volo.



Elencate le città in ordine di visita nel percorso più economico.

Soluzione. La soluzione è: **Milano, Madrid, Londra, Berlino, Praga, Parigi**.

Ecco un classico *problema di ottimizzazione*: una soluzione ottima *minimizza i costi*! Qui si può seguire, fino a un certo punto, una *strategia golosa*: scegliamo di volta in volta i costi minimi. Per partire da Milano abbiamo due voli, uno costa 40, l'altro 100: quale scegliereste? Ora siamo a Madrid: la città più economica da raggiungere (senza tornare indietro) è Londra. Da qui ci converrebbe andare a Praga, ma poi ci troveremmo a mezza via tra Parigi e Berlino, che nel nostro schema non sono collegate direttamente. Ci vediamo quindi costretti ad andare a Berlino (costa meno di Parigi) e a concludere il nostro viaggio (senza ritorno) a Parigi.

Difficoltà ed errori frequenti. Provare tutti i percorsi possibili non è certamente un errore, anzi, è la cosa giusta da fare, in generale, ma presenta qualche difficoltà (specie se le città sono tante e tutte collegate a due a due)! È invece certamente un errore aspettarsi di poter sempre

scegliere il collegamento più economico: è probabile che non ci consenta di completare il nostro viaggio, con o senza ritorno a casa.

È meno ovvio che la scelta golosa non ci garantisca il percorso più economico, ma è facile convincersene. Supponiamo che da Milano possiate andare a Parigi (costo 100) o a Berlino (costo 200). Voi scegliete Parigi, e da qui potete andare a Berlino (costo 200) o a Londra (costo 100). Scegliete Londra, ma non volete perdervi Berlino. Peccato che il volo Londra-Berlino costi 400: avete speso in tutto 600 euro. Se invece aveste scelto l'itinerario Milano-Berlino-Parigi-Londra avreste speso 100 euro in meno.

Contesto informatico. Siamo di fronte a una variante del classico *problema del commesso viaggiatore*, che deve visitare determinate città minimizzando le spese. Di solito il commesso viaggiatore torna a casa dopo aver compiuto un giro tra le città, ma ci sono altri problemi in cui si tratta, per esempio, di *minimizzare la distanza* da percorrere per andare da un punto a un altro. Per certi problemi esistono metodi di soluzione (*algoritmi*) efficienti (come per il problema di minimizzare la distanza), per altri no, e spesso non è facile sapere se siamo in un caso o nell'altro. Il problema del commesso viaggiatore appartiene alla classe dei *problemi NP-completi* su cui diciamo qualcosa in più a proposito del Quesito 11. "Scarabeo".

Nella pratica s'incontrano spesso problemi per risolvere i quali non si conosce alcun algoritmo efficiente e pertanto si deve ricorrere ad algoritmi esaustivi che sostanzialmente esaminano tutti i casi possibili, escludendone al più una parte non preponderante. Questi algoritmi effettuano la ricerca sistematica di una soluzione, fino a trovarla o a rilevarne la non esistenza.

Il problema del commesso viaggiatore è un esempio di questa classe di problemi. Tra le altre, sono state studiate, già negli anni '60, delle tecniche di *interruzione* (*branch and bound*) per evitare di continuare la ricerca dopo una scelta che si riveli incapace di generare una soluzione ottima. Tali tecniche possono ridurre sensibilmente il tempo di esecuzione; tuttavia le prestazioni non migliorano al punto da poter risolvere problemi con un numero di città significativamente maggiore. Se si vuole fare un giro completo di n città e ognuna di queste è collegata direttamente a ciascuna delle altre, allora i possibili giri completi (tecnicamente, i *cicli hamiltoniani*) sono $(n-1)!/2$ (prescindendo dal senso di percorrenza); il punto esclamativo dopo $n-1$ si legge "*fattoriale*" e indica il prodotto di tutti i numeri interi da 1 a $n-1$. Quando n

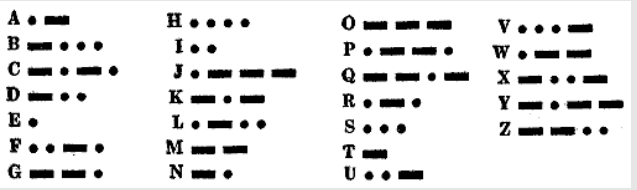
è 20, il numero dei possibili giri completi è circa $6 \cdot 10^{16}$ (60 milioni di miliardi), e ad esaminarli al notevole ritmo di cento milioni al secondo s'impiegherebbe una ventina d'anni!

In fondo, però, se si trova una soluzione *soddisfacente* a un problema, anche se *non la migliore possibile*, possiamo accontentarci. Per i problemi difficili in cui non è praticamente possibile (o è troppo costoso) trovare una soluzione *ottima*, esistono spesso degli *algoritmi di approssimazione* (alcuni basati sulla *strategia golosa* sopra accennata) che consentono di trovare soluzioni più o meno buone in tempi accettabili.

Parole chiave e riferimenti: problema del commesso viaggiatore, problemi di ottimizzazione, problemi NP-completi, algoritmi di approssimazione, algoritmi golosi.

Quesito 16: Alfabeto Morse

Il piccolo Romeo e la sua nuova fiamma Giulietta vogliono comunicare a distanza senza spendere in telefonate e hanno deciso di usare il codice Morse senza pause tra una lettera e l'altra. In pratica usano il telefonino facendolo solo squillare: uno squillo breve corrisponde al punto ($\cdot$) e uno squillo lungo alla linea ($-$).



Presto però hanno scoperto che il loro modo di comunicare ha un piccolo difetto: due parole diverse potrebbero essere codificate con la stessa sequenza di punti e linee!
Associate ad ogni parola tra: *abbaiai, abito, emersi, pianti, andato*
la parola che ha la stessa codifica tra: *pesto, addii, acuto, perviene, adepti*

Soluzione. Le associazioni corrette sono: **abbaiai-perviene, abito-pesto, emersi-addii, pianti-adepti, andato-acuto**. Inserendo uno spazio tra il codice di una lettera e il successivo si avrebbero infatti le seguenti sequenze di punti e linee:

abbaiai	·− −··· −··· ·− ·· ·− ··
perviene	·−−· · ·−· ···− ·· · −· ·
abito	·− −··· ·· − −−−
pesto	·−−· · ··· − −−−
emersi	· −− · ·−· ··· ··
addii	·− −·· −·· ·· ··
pianti	·−−· ·· ·− −· − ··
adepti	·− −·· · ·−−· − ··
andato	·− −· −·· ·− − −−−
acuto	·− −·−· ··− − −−−

Difficoltà ed errori frequenti. Il quesito non presenta particolari difficoltà: basta calcolare il codice Morse di ciascuna parola e poi accoppiare le parole con lo stesso codice. Tuttavia, questa soluzione

può risultare lunga! I tempi possono essere accorciati facendo alcune osservazioni...

Ad esempio, si può notare che la codifica delle lettere iniziali comincia sempre nello stesso modo e questo suggerisce di partire dal fondo. Poiché la lettera 'i' si codifica con $\cdot\,\cdot$ mentre la lettera 'o' si codifica con $-\,-\,-$ possiamo da subito stabilire delle sotto-associazioni: "abito" e "andato" andranno associati necessariamente con "pesto" e "acuto"; a questo punto basta guardare il codice della terzultima lettera per capire che "abito" va con "pesto" e "andato" con "acuto".

Per quanto riguarda le altre tre coppie di parole, si può ad esempio osservare che il codice Morse della parola "emersi" termina con una sequenza di 6 punti e di conseguenza va associato con "addii" (per escludere "adepti" e "perviene" basta osservare che sia 't' che 'n' contengono un trattino nel loro codice).

Contesto informatico. In tutte le attività nelle quali si trattano informazioni, come nelle telecomunicazioni e nell'informatica, per *codice* si intende una modalità per rappresentare un insieme di oggetti di varia natura, mediante un insieme prefissato di *simboli*.

In questa versione semplificata del codice Morse, si usano solo 2 simboli, il punto e il trattino, quindi possiamo dire che si tratta di un codice *binario*. Nella versione originale del codice Morse invece i *simboli* sono 5: oltre al punto (segnale breve) e al trattino (segnale lungo), vanno considerati anche altri 3 simboli, che corrispondono a 3 tipi di pause (pausa breve tra lettere, pausa media tra parole, pausa lunga tra frasi).

Anche nei computer le informazioni sono codificate con un codice binario, in questo caso i simboli usati sono lo zero (0) e l'uno (1). Numeri, testi, colori, suoni, eccetera, sono tutti rappresentati come sequenze di 0 e 1! La parola *bit*, che avrete sicuramente sentito usare qualche volta, serve proprio ad indicare una di queste due cifre binarie (in inglese *BInary digiT*).

Per poter essere effettivamente usato, è necessario che un codice sia *non ambiguo*: con questo intendiamo dire che si deve poter decodificare in modo unico ogni sequenza. Ovvero, data la codifica di una parola (o in generale di una frase), si deve poter risalire senza dubbi alla parola (o frase) di partenza. Nel caso del codice Morse originale, le pause garantiscono che non ci sia ambiguità; nel caso del nostro codice Morse semplificato, invece, non si può garantire questa condizione.

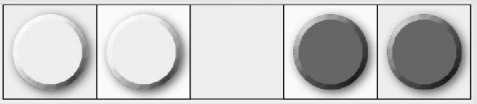
Parole chiave e riferimenti: rappresentazione dell'informazione, codici, ambiguità

Quesito 17: Hop!

Risolvete questo solitario: dovete trovare una sequenza di mosse per portare le pedine nere al posto di quelle bianche e viceversa.

- La casella centrale è inizialmente libera.
- Le pedine bianche si spostano soltanto verso destra, quelle nere soltanto verso sinistra.
- Una pedina può spostarsi di una casella in avanti, se la casella davanti è libera, oppure può saltare una pedina dell'altro colore, se la casella successiva a questa è libera.

All'inizio dovete muovere una pedina bianca.



Soluzione. Il gioco si risolve in esattamente 8 mosse... anche se le potenze di 2 qui non c'entrano! Numerando i cinque posti da sinistra a destra con -2, -1, 0, 1 e 2, l'unica sequenza di mosse che risolve il gioco è: -1, 1, 2, 0, -2, -1, 1, 0 (senza incorrere in ambiguità, si è indicato ogni volta il posto della pedina che muove). Si noti che le mosse *non* si alternano tra i due colori, e nulla era stato detto in proposito nel testo della prova!

In generale, nel gioco con n pedine bianche e n nere, la soluzione rimane unica e comporta $n(n+2)$ mosse; il gioco classico ha 4 pedine per ciascun colore (ovviamente il tavoliere di gioco ha 9 caselle), e quindi si risolve in 24 mosse. Se ciascun numero indica la posizione della pedina da spostare, a partire da $-n$ con la prima a sinistra fino a n per la prima a destra, si trova che le sequenze di mosse risolutive (se inizia a muovere la pedina bianca) sono, al variare di n, le seguenti:

per $n = 1$: -1, 1, 0

per $n = 2$: -1, 1, 2, 0, -2, -1, 1, 0

per $n = 3$: -1, 1, 2, 0, -2, -3, -1, 1, 3, 2, 0, -2, -1, 1, 0

per $n = 4$: -1, 1, 2, 0, -2, -3, -1, 1, 3, 4, 2, 0, -2, -4, -3, -1, 1, 3, 2, 0, -2, -1, 1, 0

...

Che cosa notate in queste sequenze? Se escludiamo l'ultima mossa (l'ultimo 0), ciascuna di queste sequenze è *speculare* rispetto al centro, cambiando i segni: ciò rispecchia il fatto che la configurazione iniziale del tavoliere di gioco è speculare rispetto a quella finale, la configurazione *dopo* la prima mossa è speculare rispetto a quella che si ha *prima* di fare l'ultima mossa, la configurazione *dopo* la seconda mossa è speculare rispetto a quella che si ha *prima* di fare la penultima mossa, e così via. Inoltre, come si può notare, ciascuna sequenza risolutiva è ottenuta dalla precedente (quella corrispondente a n inferiore di un'unità) aggiungendovi opportune mosse "al centro" (in numero dispari, via via crescente: dapprima 5, poi 7, poi 9, poi 11,...).

Contesto informatico. Dal punto di vista informatico, si potrebbe pensare di realizzare un programma che risolve il gioco, magari procedendo per tentativi, se non si conosce o non si ha voglia di pensare a come codificare una strategia che raggiunga l'obiettivo senza mai commettere errori, cioè – come si dice – *in modo deterministico*. Che cosa significa che un algoritmo procede per tentativi? Nel nostro caso, significa che riceve il tavoliere di gioco nella sua configurazione attuale (all'inizio, ovviamente, sarà quella iniziale!); per prima cosa, guarda se corrisponde alla configurazione finale: se sì, allora ha trovato una soluzione; altrimenti, guarda quali sono le mosse possibili a partire da questa configurazione. Se ve ne sono, prova a fare la prima, arriva in una nuova configurazione, e da lì il procedimento si ripete, *ricorre* (come in *ricorsione*); se non giunge alla soluzione, prova a fare la seconda mossa e cosa via.... Quando non è più possibile tentare una nuova mossa, perchè non ce n'erano o si sono provate tutte senza arrivare a una soluzione, torna indietro di un passo (operazione che, tecnicamente, si chiama *backtracking*), rimuovendo l'ultima mossa fatta, e così via. In questo modo, l'algoritmo esplora tutto l'albero di gioco, fino a trovare una soluzione, se esiste, o a provare *esaustivamente* (o *per esaurimento*) che non ve ne sono! In una configurazione *legalmente raggiungibile* del gioco Hop, quante sono al più le mosse possibili?

Parole chiave e riferimenti: albero di gioco, algoritmo esaustivo, ricorsione, backtracking.

Quesito 18: Reazioni chimiche

In un laboratorio di chimica il professor Natta sta studiando tre diverse molecole che chiameremo A, B, C. Se queste molecole vengono allineate opportunamente reagiscono, trasformandosi secondo queste regole:
1. AA diventa BC
2. BCC diventa CCA
3. CAB diventa A (mentre B e C... evaporano)

Se due o piú regole possono essere applicate si verifica una reazione a caso tra quelle possibili: ad esempio, partendo da BCCAA, il professore ha ottenuto diverse volte CCAAA, perché è stata applicata la regola 2, ma altre volte ha ottenuto BCCBC, perché è stata applicata la regola 1. Se il professore studia la sequenza AACAAB, che cosa potrebbe ottenere dopo esattamente tre reazioni?

A CACB

B CCABCB

C CCBA

D BCCACB

E Nessuna delle elencate

Soluzione. La risposta corretta è **B**.

Per affrontare questo quesito bisogna innanzitutto capire come si applicano le regole date ad una sequenza di molecole. Nell'esempio si dice che BCCAA si può trasformare in CCAAA, questo perché le prime tre lettere di BCCAA, cioé BCC, possono essere trasformate secondo la regola 2 in CCA. Quindi si ottiene CCAAA. Però anche le ultime due lettere di BCCAA, cioé AA, si possono trasformare secondo la regola 1 in BC, ottenendo BCCBC. Una volta capito come applicare le regole si deve cominciare a trasformare la sequenza AACAAB studiata dal professore e vedere quali sono le varie possibilità. Innanzitutto ci sono due coppie AA che si possono trasformare in BC usando la regola 1. Se la applichiamo ad entrambe le coppie otteniamo BCCBCB. A questo punto possiamo solo trasformare le prime tre lettere BCC in CCA, ottenendo CCABCB.

Difficoltà ed errori frequenti. Questo quesito può essere difficile perché non è detto che applicando le regole si ottenga una sequenza di molecole presente nelle risposte possibili. In effetti avremmo potuto applicare la regola 1 alla prima coppia AA e quindi, applicando la regola 2 a BCC, avremmo ottenuto: AACAAB $\rightarrow$ BCCAAB $\rightarrow$ CCAAAB. A questo punto si va fuori strada applicando la regola 1 alla prima coppia AA: CCAAAB $\rightarrow$ CCBCAB, ottenendo una soluzione non presente tra le risposte possibili. Comunque anche in questo caso possiamo applicare la regola 1 sulle ultime AA ottenendo CCAAAB $\rightarrow$ CCABCB, la risposta B.

Alla sequenza CCABCB possiamo ancora applicare la regola 3 ottenendo la risposta A (CACB), ma questa sequenza è ottenuta con quattro reazioni e non tre come richiesto.

In effetti il metodo piú sicuro per affrontare questo quesito è quello di scrivere ad ogni passo ogni possibile applicazione delle regole per tre passi consecutivi, cosí facendo si generano tutte le possibilità e si è sicuri di trovare la risposta corretta.

Contesto informatico. Le regole che trasformano opportunamente sequenze di simboli si chiamano *sistemi canonici di Post* (in inglese "Post canonical systems"). Questo tipo di sistema è stato presentato in un articolo per la prima volta da Emil Post nel 1943. In pratica regole come le 1, 2, 3 riscrivono una sequenza di simboli in maniera opportuna ed interessa capire quali sono le proprietà di questi sistemi. Nel caso di questo quesito possiamo ad esempio osservare che le sequenze possono solo diminuire la loro lunghezza, mai aumentarla. Un'altra osservazione è la presenza di termini bloccati, su cui non possiamo applicare regole, come ad esempio CACB. Ovviamente la domanda poteva essere posta al contrario, chiedendo da quale sequenza tra quelle disponibili si sarebbe arrivati ad AACAAB in tre reazioni. Cosí facendo si chiede di leggere le regole da destra verso sinistra invece che da sinistra verso destra. Un sistema canonico di Post simile a questo è stato proposto nel libro: "Gödel, Escher, Bach: un'Eterna Ghirlanda Brillante" di Douglas Hofstadter, vincitore del premio Pulitzer nel 1980. Nel libro il gioco si chiama "MU".

Parole chiave e riferimenti: sistemi canonici di Post, sistemi di riscrittura dei termini, gioco Mu, grammatiche libere dal contesto.

Quesito 19: In pizzeria

In una pizzeria Gianni, il cameriere, e Ciro, il pizzaiolo, si mettono d'accordo su uno schema per prendere le ordinazioni. Hanno due vassoi, che chiameremo A e B, dove vengono appoggiate le ordinazioni scritte su un foglietto: ogni volta che Gianni ha un'ordinazione la appoggia in cima al vassoio A, invece non appena Ciro ha tempo per preparare una pizza prende il foglietto in cima al vassoio B, lo getta via e prepara quel tipo di pizza. Nel caso in cui il vassoio B risulti vuoto, allora Ciro deve prendere il foglietto in cima al vassoio A e spostarlo in cima al vassoio B e ripetere questa operazione fino a svuotare il vassoio A, dopodiché deve prendere il foglietto in cima al vassoio B.
Una sera succedono le seguenti cose:

1. Gianni porta l'ordine per una MARGHERITA
2. Gianni porta l'ordine per una QUATTRO STAGIONI
3. Ciro ha tempo per preparare una pizza
4. Gianni porta l'ordine per una QUATTRO FORMAGGI
5. Ciro ha tempo per preparare una pizza
6. Gianni porta l'ordine per una NAPOLETANA
7. Gianni porta l'ordine per una MARGHERITA
8. Ciro ha tempo per preparare una pizza
9. Gianni porta l'ordine per una VEGETARIANA

Riproduci quello che è successo in pizzeria spostando i foglietti delle ordinazioni, uno alla volta, esattamente come farebbero Ciro e Gianni.

Soluzione. Per rispondere al quesito occorre fare passo passo ciò che farebbero Gianni e Ciro, spostando correttamente i foglietti delle ordinazioni. Associamo un nome ad ogni operazione possibile in modo da poter descrivere la soluzione in modo sintetico. Le operazioni possibili sono 6:

- A+: metti il primo foglietto rimasto nell'elenco sul vassoio A;
- B+: metti il primo foglietto rimasto nell'elenco sul vassoio B;
- A-: togli il foglietto in cima al vassoio A e mandalo in preparazione;
- B-: togli il foglietto in cima al vassoio B e mandalo in preparazione;
- AB: sposta il foglietto in cima al vassoio A in cima al vassoio B;
- BA: sposta il foglietto in cima al vassoio B in cima al vassoio A.

In realtà, di queste operazioni, solo tre (A+, AB, B-) sono effettivamente compiute da Ciro e Gianni; in particolare Gianni ne fa solo una (A+) e Ciro, a seconda che il suo vassoio sia vuoto o meno, fa AB un certo numero di volte (tante quanti sono gli ordini sul vassoio A) e poi B-, oppure solo B-.

La soluzione al quesito è data dalla sequenza: A+ A+ AB AB B- A+ B- A+ A+ AB AB AB B- A+. Infatti:

- i primi due eventi riguardano Gianni (A+ A+)
- poi tocca a Ciro; il suo vassoio è vuoto e sul vassoio A ci sono due ordini, quindi (AB AB B-)
- poi tocca di nuovo a Gianni (A+)
- poi a Ciro, e il suo vassoio ora non è vuoto (B-)
- poi a Gianni per due volte (A+ A+)
- poi a Ciro; il suo vassoio ora è vuoto e sul vassoio A ci sono tre ordini, quindi (AB AB AB B-)
- poi a Gianni (A+)

Difficoltà ed errori frequenti. Lo schema concordato da Gianni e Ciro per gestire le ordinazioni è di fatto un algoritmo, la cui esecuzione non è banale, contenendo selezioni e iterazioni, per di più annidate. L'algoritmo è espresso in linguaggio naturale e occorre quindi riconoscere correttamente le strutture di controllo le condizioni di terminazione delle ripetizioni.

Contesto informatico. La soluzione del quesito coinvolge alcune competenze tipiche dell'informatica:

- Comprensione di un algoritmo e sua esecuzione passo passo
- Valutazione delle condizioni per la selezione
- Capacità di gestire correttamente l'iterazione

Si può notare che l'algoritmo in questione permette di gestire la preparazione delle pizze rispettando l'ordine in cui sono state ordinate, infatti la prima pizza che viene ordinata viene preparata per prima e poi man mano le altre, sempre secondo l'ordine di ordinazione. Per far questo sui vassoi vengono create due *pile* di ordini (strutture LIFO = "Last in First out", cioè "l'ultimo arrivato è il primo ad uscire") che vengono gestite in modo da eseguire le ordinazioni secondo una *coda* (struttura FIFO = "First In First Out", cioè "il primo arrivato è il primo ad uscire"). L'algoritmo è quindi un esempio di gestione di *strutture di dati dinamiche*.

Parole chiave e riferimenti: algoritmo, strutture di controllo, pile, code.

Quesito 20: La damigiana e il bottiglione

Il signor Sbevazzo ha a disposizione due recipienti vuoti (una damigiana e un bottiglione), in grado di contenere rispettivamente 9 litri e 4 litri. Vuole riempire la damigiana con esattamente 6 litri di acqua, ma le uniche operazioni che sa fare sono le seguenti:

RIEMPI D riempire la damigiana,

SVUOTA B svuotare il bottiglione,

VERSA versare il contenuto della damigiana nel bottiglione, fino a riempire il bottiglione o a svuotare la damigiana nel bottiglione.

Qual è la *piú breve* sequenza di queste operazioni che gli permette di raggiungere il suo obiettivo?

Soluzione. La sequenza corretta di operazioni è quella che compare nella prima colonna della tabella a fianco, che riporta anche lo *stato* dei due contenitori dopo l'esecuzione di ciascuna operazione: la prima colonna riporta l'operazione appena eseguita, la seconda e la terza riportano la quantità di acqua contenuta nella damigiana e nel bottiglione, rispettivamente. Si vede chiaramente che al termine della sequenza la damigiana contiene esattamente 6 litri d'acqua.

operazione	D	B
RIEMPI D	9	0
VERSA	5	4
SVUOTA B	5	0
VERSA	1	4
SVUOTA B	1	0
VERSA	0	1
RIEMPI D	9	1
VERSA	6	4

Ok, direte voi... ci hai convinto che la soluzione è giusta, ma come hai fatto a trovarla? In questo caso non esistono strategie precise per "scoprire" la soluzione. Si può cominciare ad osservare che le prime 3 operazioni sono praticamente obbligate, dato che qualsiasi altra alternativa "annullerebbe" le istruzioni precedenti e quindi non sarebbe conveniente visto che cerchiamo la sequenza *più breve*.

Un modo di procedere può essere quello di partire dal fondo... per avere 6 litri nella damigiana, posso cominciare a riempirla e trovare un modo per svuotarla di soli 3 litri. Ma 3 si ottiene sottraendo 1 da 4, che è la capacità del bottiglione. Allora, se riesco a riempire il bottiglione con 1 solo litro di acqua, ho trovato la soluzione (con l'operazione VERSA potrò infatti riempire il bottiglione,

aggiungendo al litro già presente 3 litri provenienti dalla damigiana, e ritrovandomi quindi con 6 litri nella damigiana stessa!). Come fare allora per riempire il bottiglione con un solo litro? Posso riempire la damigiana, togliere due volte 4 litri (travasandoli nel bottiglione e svuotando quest'ultimo) e infine versare l'ultimo litro rimasto nel bottiglione!

Difficoltà ed errori frequenti. Il quesito richiede di risolvere il problema del signor Sbevazzo costruendo un *algoritmo* che usa solo le tre *operazioni fondamentali* RIEMPI D, VERSA e SVUOTA B. Il quesito è simile a quello di Babbo Natale, ma questa volta la costruzione dell'algoritmo è richiesta esplicitamente.

Come in quel caso, la difficoltà dell'esercizio risiede nel fatto che è consentito soltanto l'uso di alcune operazioni predefinite, cosa che impedisce di usare alcuni approcci piú "naturali". Se il signor Sbevazzo avesse a disposizione una caraffa da un litro, basterebbe riempirla e svuotarla nella damgiana 6 volte!

Contesto informatico. La soluzione del quesito coinvolge alcune competenze tipiche dell'informatica:

- la scomposizione di un compito complesso in una sequenza di passi elementari e dunque l'ideazione di un algoritmo;
- l'utilizzo di alcuni "mattoni" predefiniti per costruire qualcosa di nuovo.

Parole chiave e riferimenti: algoritmo, problem solving, composizione di primitive

Quesito 21: Biancaneve e il nano goloso

Uno dei sette nani (Dotto, Brontolo, Pisolo, Mammolo, Gongolo, Eolo o Cucciolo) ha mangiato la torta lasciata sul tavolo della cucina. Per scoprire chi è stato, Biancaneve interroga i nani con domande alle quali i nani possano rispondere in coro, senza bugie, solo con un sì o con un no.
Se Biancaneve sceglie le domande con cura, qual è il minimo numero di domande che le garantisce di scoprire il colpevole?

Soluzione. La risposta è **3**.

Biancaneve mette Cucciolo, Brontolo e Pisolo da una parte e chiede "il colpevole è uno di loro?". I nani in coro rispondono "no!". Ora Biancaneve prende in disparte Eolo e Dotto e ripete la stessa domanda. Questa volta la risposta è "sì!". Biancaneve chiede "è Eolo il colpevole?" e tutti rispondono "no!". Biancaneve prende Dotto e se lo mangia...

Come si vede, 3 domande sono *sufficienti* se ogni volta si *dimezza* il numero dei possibili colpevoli.

Difficoltà ed errori frequenti. Si potrebbe credere che siano *necessarie*, nel caso più sfortunato, tante domande quanti sono i possibili colpevoli, ma dipende da come si fanno le domande. Se avete pensato che ancora una volta fossero coinvolti algoritmi golosi, avete sbagliato...

Contesto informatico. In Informatica molti *problemi di ricerca* (non di ricerca scientifica, ma di ricerca di "oggetti") possono essere risolti proprio con una tecnica *dicotomica* simile a quella illustrata: si cerca ogni volta di dimezzare il numero degli oggetti da considerare. Per esempio, gli oggetti possono essere distribuiti in una struttura ad *albero binario di ricerca*, in cui ogni oggetto ha al più due *figli*, e si esamina ogni volta uno solo dei due figli. Se l'albero è "fatto bene", ossia, come si dice, *bilanciato*, per ogni oggetto esaminato vengono scartati metà degli oggetti rimanenti. In questo modo esaminando solo 10 oggetti (o meno) è possibile trovare quello cercato tra altri 1023: un buon modo per cercare un ago in un pagliaio!

La *ricerca dicotomica* o *binaria* permette di risolvere un problema tramite la *suddivisione del problema in sottoproblemi* di dimensioni più piccole, che vengono a loro volta risolti per suddivisione, in maniera *ricorsiva*. L'idea rispecchia una strategia generale, quella del *dìvide et ìmpera*, come l'avrebbe detta Giulio Cesare.

Spieghiamo meglio questa ricerca *dicotomica* o *binaria* esaminando la ricerca di un nome in un elenco *ordinato* (come quello telefonico). Supponiamo di avere un elenco di 1024 nomi, in ordine alfabetico, ciascuno scritto su una riga e preceduto dal suo numero d'ordine, da 1 a 1024. Se dobbiamo cercare un certo nome, possiamo iniziare guardando quello al posto 512. Ci sono tre possibilità.

1. Il nome al posto 512 *è proprio* quello cercato: in tal caso abbiamo finito la ricerca!
2. Il nome cercato *precede* il 512-esimo: allora ripeteremo il procedimento coi primi 511 nomi.
3. Il nome cercato *segue* il 512-esimo: allora ripeteremo il procedimento con gli ultimi 512 nomi.

Con al più 10 confronti (dato che ogni volta resta da esaminare un elenco di nomi che ha lunghezza al più pari alla precedente potenza di 2, e 2 elevato alla potenza 10 fa 1024) possiamo trovare il nome cercato nell'elenco (e quale posto occupa)! Come nel caso di Biancaneve, quando restano solo due nomi, se quello cercato non è quello esaminato allora dev'essere l'altro... a meno che il nome non manchi dall'elenco. Solo in questo caso occorre fare un ultimo confronto, ma Biancaneve sapeva già che uno dei nani era colpevole!

Qualcuno obietterà che potrebbe essere più conveniente iniziare non dal nome di mezzo nell'elenco, ma ad esempio da quello posto a tre quarti, se il nome cercato inizia con la lettera R... In effetti, questa può essere una buona idea, nell'ipotesi di nomi le cui iniziali siano distribuite più o meno uniformemente sull'alfabeto italiano. Tuttavia, pensate un po' al caso sfortunato in cui quasi tutti i nomi nel nostro elenco inizino proprio con la lettera R!

Parole chiave e riferimenti: problemi di ricerca, ricerca dicotomica, suddivisione in sottoproblemi, strategia dìvide et ìmpera, ricorsione

Quesito 22: Le gemelle

Alice, Bice e Circe sono tre gemelle indistinguibili, ma Alice dice sempre la verità mentre Bice e Circe dicono sempre bugie. Giancarlo il bibliotecario incontra una delle tre e vuole sapere se si tratta di Circe, per farsi restituire un libro prestato. Con quale domanda può scoprirlo?

A Sei Alice? B Sei Bice? C Sei Circe?
D non può scoprirlo E deve fare altre domande

Soluzione. La risposta corretta è la **B**.

Chiedere direttamente alla gemella incontrata se è Circe ci lascia nel dubbio: infatti Circe risponderebbe di no ma anche Alice, mentre Bice risponderebbe sì, e quindi potremmo riconoscere Bice ma non Circe. La domanda "sei Alice?" riceve tre risposte affermative. Se invece chiediamo "sei Bice", Alice e Bice risponderebbero no e soltanto Circe risponderà affermativamente (mentendo). Possiamo quindi smascherare Circe senza bisogno di altre domande.

Contesto informatico. Il problema delle gemelle è un classico rompicapo di tipo logico, precisamente del tipo *cavalieri e (fur)fanti*: i cavalieri dicon sempre la verità mentre i (fur)fanti (tradotti come "fanti" nel libro di Raymond Smullyan "Fare il verso al pappagallo", Bompiani, 1990) mentono sempre. Su Smullyan (logico di primo piano e personaggio interessante) e su questi rompicapo si può vedere la *Wikipedia*, per esempio `http://it.wikipedia.org/wiki/Cavalieri_e_furfanti`.

L'uso della logica (o semplicemente del buon senso) sembra talvolta affievolirsi con l'età degli studenti (Smullyan racconta di una soluzione brillante a uno dei suoi quesiti trovata da una lettrice di 9 anni!). I problemi logici sono spesso un po'... contorti, ma la logica serve anche in questioni semplici e pratiche. Se, per esempio, in un certo tipo di tabelle si prosegue una ricerca fino a quando si arriva a una casella vuota, che cosa succederà se la tabella è completamente piena? E come si misura quanto è piena una tabella (o un'aula)?

Parole chiave e riferimenti: problemi logici, tabelle hash