

Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

giovedì 15 dicembre 2016

Questa scheda contiene alcuni quesiti tratti dal kangourou dell'Informatica (edizioni dal 2008 al 2015) e dal Bebras dell'Informatica (edizione 2016/17); per ogni quesito, oltre al testo è riportata la soluzione e una breve spiegazione su come ottenerla, assieme a qualche indicazione di riferimento nella sezione "Anche questa è informatica".

I quesiti hanno difficoltà molto variabili e sono raccolti qui perché fanno tutti riferimento in qualche senso alla struttura dati "albero" e possono offrire diversi spunti di riflessione. Nel contesto di questa esercitazione, non è interessante tanto *risolvere* i quesiti (ovvero trovare la risposta giusta) ma ragionare sui diversi contesti in cui gli alberi possono essere utilizzati per modellare situazioni concrete, su come problemi molto diversi possano essere formalizzati e affrontati mediante gli alberi, sulle proprietà dei vari tipi particolari di alberi (ad esempio: alberi di decisione, alberi binari di ricerca, alberi red-black, heap).

Dal tronco al fiore (3 punti)

Spesso in informatica i dati sono organizzati con un "albero"; in un "albero binario" da una ramificazione partono sempre solo due rami.



Il fiore rosso in figura può essere raggiunto dal "percorso" TSSSD dove

- T significa "partenza dal tronco"
- S significa "ramificazione verso sinistra"
- D significa "ramificazione verso destra"

Quale percorso conduce al fiore blu?

Soluzione. Il fiore blu è raggiungibile tramite il percorso **TDSDDSSS**; si deve partire dal Tronco, alla prima diramazione prendere il ramo di Destra, poi quello di Sinistra e così via.

Anche questa è informatica! Gli *alberi* sono *strutture di dati* fondamentali in informatica e li abbiamo incontrati spesso nei quesiti delle scorse edizioni. Sono utili per rappresentare informazioni di tipo gerarchico, come nel caso degli alberi genealogici (vedi ad esempio i quesiti “La famiglia Pre” - edizione 2010 o “Il pianeta Alber” - edizione 2009) o per visualizzare le possibili mosse che si possono eseguire in un gioco di strategia (vedi ad esempio i quesiti “Una partita a tris” - edizione 2010, “Hop” - edizione 2009).

Parole chiave e riferimenti: Alberi; strutture di dati.

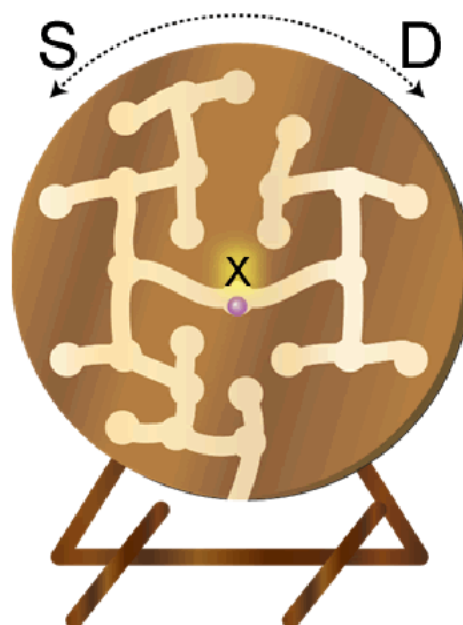
Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras e usato nell'Kangourou dell'Informatica edizione 2011/12.

Un giocattolo rotante (2 punti)

Un canguro giocherellone ha trovato un pezzo di legno in cui i tarli hanno scavato il sistema di buchi e gallerie indicato in figura.

Usando una biglia, il legno può essere sfruttato come giocattolo: all'inizio si mette la biglia al centro (X); l'obiettivo è fare uscire la biglia ruotando il legno a sinistra (S) o a destra (D); dopo ogni rotazione la biglia si ferma nel primo buco o fuori dal pezzo di legno.

Quale sequenza permette di fare uscire la biglia?



Soluzione. La sequenza che consente di far uscire la biglia è SDDSDS.

Anche questa è informatica! Il giocattolo potrebbe essere rappresentato usando un *albero binario*, una *struttura di dati* molto comune in ambito informatico. In particolare, ogni buco viene rappresentato da un *nodo* dell'albero (il buco X è la *radice*) e i nodi sono collegati tra loro se e soltanto se nel giocattolo c'è una galleria che collega direttamente i corrispondenti buchi: il *figlio sinistro* di un nodo corrisponde al buco che si raggiunge da quel nodo ruotando il giocattolo verso sinistra, e similmente per il figlio destro. La sequenza SDDSDS descrive il cammino che porta dalla radice alla *foglia* (nodo senza figli) che rappresenta il buco d'uscita.

Parole chiave e riferimenti: Albero binario, nodi, foglie, cammini.

Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras della Slovenia e tradotto per il Kangourou dell'Informatica edizione 2013/14.

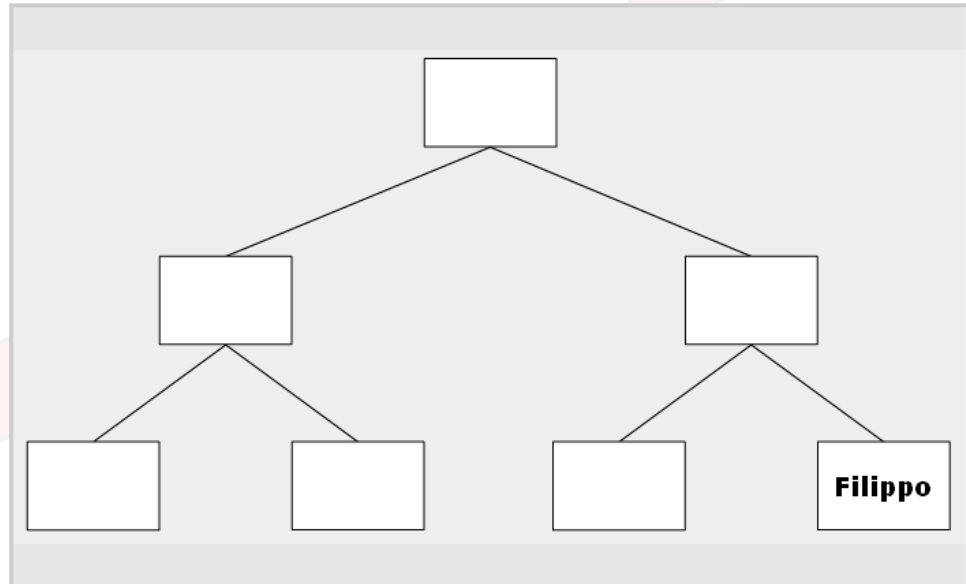
La famiglia Pre (max 6 punti)

Nella famiglia Pre vige una curiosa regola dei nomi: il nonno ha stabilito che a ogni nuovo nato della sua famiglia venga imposto il nome dell'ultimo nato con l'aggiunta, in fondo, di una nuova lettera. La famiglia festeggia la nascita dell'ultimo nato, Filippo, ed è particolarmente felice perché il nome del padre di ciascuno è lungo la metà (arrotondata per difetto quando il nome ha un numero dispari di lettere) del nome del figlio. Per esempio, il papà di Filippo si chiama Fil.

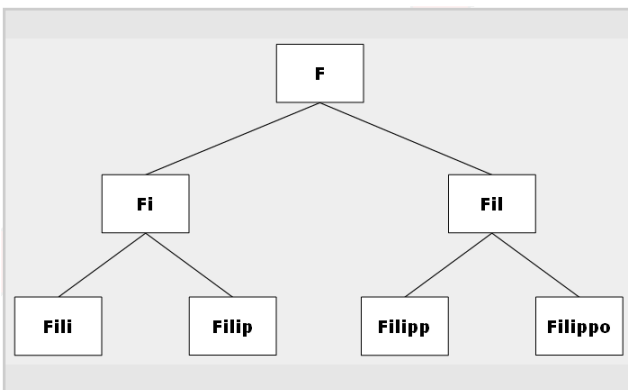
Completa l'albero genealogico inserendo un nome in ogni casella.

I figli devono essere ordinati in modo che il maggiore sia a sinistra.

Per ogni nome corretto, guadagnerai un punto.



Soluzione. La soluzione corretta è riportata nella figura. Poiché il nome del padre di ogni nato nella famiglia è lungo la metà di quello di un figlio, ogni padre può avere *al massimo due figli*: se per esempio, come nel nostro caso, il nome di un figlio è di 7 lettere, il padre avrà un nome di 3 lettere, e potrà avere un altro figlio con un nome di 6 lettere, ma non altri.



Anche questa è informatica! In informatica, il termine *albero* indica una *struttura di dati*, ossia un modo per organizzare delle informazioni. Gli elementi di questa struttura sono organizzati in modo gerarchico, come in un *albero genealogico*, dove la gerarchia è definita dalla relazione *padre-figlio*. Nel nostro caso, l'albero si dice *binario*, perché ogni padre ha al massimo due figli e inoltre vi è una chiara distinzione tra figlio *destro* e figlio *sinistro*. Se non avessimo precisato nel testo della prova che il figlio maggiore (per età) deve essere collocato a sinistra, ci sarebbe stata un'ambiguità.

Le curiose proprietà dei nomi della famiglia Pre garantiscono qualcosa in più per l'albero binario: non solo conosciamo padri e figli, ma possiamo elencare i componenti della famiglia (ossia i *nodi* dell'albero), generazione per generazione, ossia livello per livello nell'albero, andando da sinistra a destra, in perfetta alternanza tra figli sinistri e figli destri. Questa

proprietà consente di tenere i nodi semplicemente in una *tabella* (spesso chiamata *array*), più facile da gestire di quanto lo sia un albero.

Si può così costruire un'altra importante struttura di dati: lo *heap* (*mucchio*, in italiano, ma quasi nessuno lo chiama così). In uno heap i dati devono essere collocati rispettando un'ulteriore regola: quella che *il dato associato a un figlio non sia maggiore del dato associato al padre*. Nel nostro caso un dato che evidentemente rispetta la regola potrebbe essere l'età: nessun padre ha età minore di quella dei figli!

Il punto importante è che nella *radice* dell'albero (il nonno per noi) ci sarà sempre il dato con il valore massimo: gli heap consentono di ottenere facilmente il valore *massimo* senza bisogno di tenere completamente ordinati i nostri dati, e quindi spesso risparmiando tempo. La regola, naturalmente, può anche essere invertita, sostituendo nella definizione *maggiore* con *minore*: in questo caso avremo nella radice il valore *minimo*, come accade per la famiglia Pre se il dato è la lunghezza del nome! Non solo, la nozione di heap è estendibile anche ad alberi in cui il numero dei figli può essere maggiore di due.

Un'ultima curiosità: perché la famiglia si chiama Pre? Perché i nomi dei componenti sono tutti *prefissi* del nome dell'ultimo nato.

Parole chiave: strutture di dati, alberi, alberi binari, heap, prefissi.

Informazioni sul quesito. Il quesito è stato proposto nel Kangourou dell'Informatica edizione 2009/10.

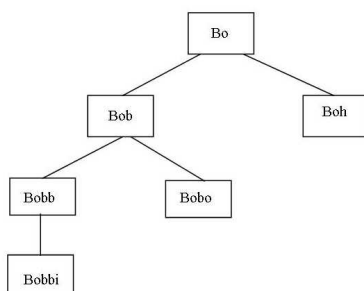
Quesito 2: Sul pianeta Alber

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 *padre* 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).

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.

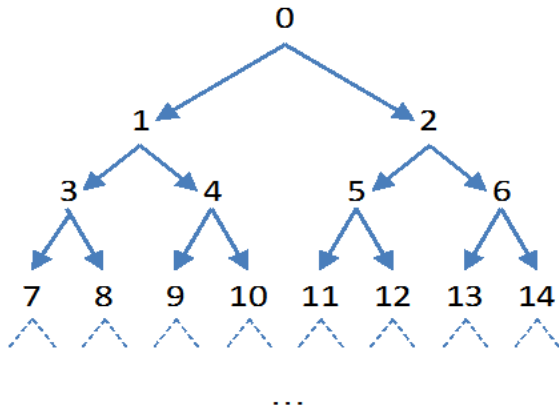
Anche questa è informatica! Gli alberi sono strutture di dati essenziali in informatica. Ci sembra importante anche mostrare come una stessa struttura possa essere rappresentata in modi diversi, piú o meno astratti. Nel nostro caso, un *linguaggio*, ossia un insieme di parole (i 'nomi'), con opportuni 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.

Informazioni sul quesito. Il quesito è stato proposto dal Kangourou dell'Informatica 2008/09.

Scendi dall'albero! (max 7 punti)

Un vostro amico ha cominciato a scrivere i numeri interi a partire da 0 nel modo seguente:



Notate che possiamo andare da 0 a 11 andando a destra (D), sinistra (S), poi ancora a sinistra (S).

Partendo da 0, quale sequenza di mosse verso sinistra (S) e destra (D) porterà al numero 100?

<input type="radio"/>	SDDSDD	<input type="radio"/>	DSSDSD
<input type="radio"/>	DDSSDS	<input type="radio"/>	SSDDSS

Se in una certa posizione c'è il numero i , quale numero si troverà sotto di lui a sinistra? Scrivete la formula giusta.

Soluzione. La sequenza di mosse corretta è DSSDSD. Dato un numero i nel diagramma (un *albero* in gergo informatico) il numero sotto a sinistra è sempre $2i + 1$.

Anche questa è informatica! L'*albero* e l'*albero binario* sono tipi di dato che rivestono notevole importanza in informatica, e già ne parlammo nelle precedenti edizioni delle gare. Il nostro è un albero binario di *numeri naturali*: ciascuno dei suoi nodi contiene un naturale; più precisamente, i nodi dell'albero sono in corrispondenza *biunivoca* con i naturali, e pertanto l'albero è infinito (in altezza): ne potremo rappresentare in concreto soltanto parti finite. Tale compito risulta assai semplice, grazie all'ordine in cui sono stati numerati i nodi. Infatti, per ogni naturale i , il nodo (contenente) i ha sotto di sé a sinistra, come *figlio sinistro*, il nodo $(2i + 1)$ e a destra, come *figlio destro*, il nodo $(2i + 2)$. Volendo generalizzare il problema proposto, dato un naturale i , eseguiamo questa procedura:

```

procedura SCEGLIFIGLIO( $i$ )
 $n \leftarrow i$ 
 fintantoché  $n > 0$   esegui
   se  $n$  è pari  allora
    scrivi D
     $n \leftarrow \frac{n-2}{2}$ 
   altrimenti
    scrivi S
     $n \leftarrow \frac{n-1}{2}$ 
   fine se
 fine fintantoché
 fine procedura
    
```

Per sapere come arrivare al nodo i partendo dalla *radice* dell'albero (nodo 0), basterà leggere la sequenza di lettere in ordine inverso rispetto a come è stata scritta! L'esecuzione della procedura ci permette anche di sapere quante mosse sono necessarie per giungere al nodo i , vale a dire qual è la sua distanza dalla radice. Se qualcuno dei lettori ha dimestichezza coi logaritmi, potrà verificare che, per ogni naturale i , tale distanza $d(i)$ è uguale alla parte intera del logaritmo in base 2 di $(i + 1)$. Inoltre, posto $\min(i) = 2^{d(i)} - 1$, i naturali che distano $d(i)$ dalla radice (cioè che stanno sullo stesso livello del

nodo i nell'albero) sono quelli che appartengono all'intervallo chiuso $[\min(i), 2 \cdot \min(i)]$. Quindi, se si vuole ottenere la sequenza di lettere nell'ordine giusto, basta eseguire quest'altra procedura:

```

procedura STAMPASEQUENZA( $i$ )
 $d \leftarrow$  parte intera del logaritmo in base 2 di  $(i + 1)$            ▷ distanza del nodo  $i$  dalla radice
 $a \leftarrow 2^d$                                                      ▷ ampiezza dell'intervallo di ricerca
 $m \leftarrow a - 1$                                                  ▷ primo numero nell'intervallo di ricerca
ripeti per  $d$  volte esegui
   $a \leftarrow \frac{a}{2}$                                              ▷ anticipatamente dimezzata!
   $x \leftarrow m + a$                                              ▷ primo numero nella seconda metà dell'intervallo di ricerca
  se  $i \geq x$  allora
    scrivi D
     $m \leftarrow x$ 
  altrimenti
    scrivi S
  fine se
fine ripeti per
fine procedura

```

Questo pseudo-codice non è altro che un particolare adattamento dell'algoritmo di *ricerca binaria*: qui si sa che la ricerca avrà successo e, inoltre, si vogliono effettuare tutti i passi che portano a un intervallo di ricerca costituito dal solo elemento cercato.

Parole chiave e riferimenti: Albero binario, ricerca binaria.

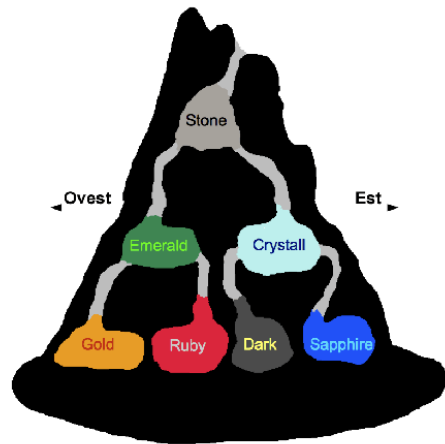
Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras del Canada e usato nel Kangourou dell'Informatica 2013/14.

Esploratori di caverne (4 punti)

Dino e Bruno sono esploratori di caverne. In questi giorni stanno esplorando il labirinto di caverne che vedete nella mappa. Hanno sette giorni per esplorare sette caverne. Ogni mattina si calano e trascorrono l'intera giornata prendendo misure e raccogliendo campioni di roccia.

Dino è un "esploratore in profondità". La sua strategia è quella di entrare in una caverna e, prima di esplorarla, verificare se ci sono caverne più profonde ancora da esplorare; inizia da ovest: se trova una caverna più profonda, si sposta in quella e fa la stessa verifica; altrimenti prova a controllare sul lato est; se non ci sono caverne più profonde ancora da esplorare, esplora quella in cui si trova; una volta finita l'esplorazione, risale nella caverna al livello superiore. Quindi Dino esplorerà Gold lunedì, Ruby martedì e Emerald mercoledì.

Bruno è un "esploratore in ampiezza". La sua strategia è quella di visitare le caverne livello per livello, partendo dalla caverna Stone, la meno profonda. Per ogni livello, esplora le caverne partendo da ovest e andando verso est. Quindi esplorerà Emerald martedì e Crystall mercoledì.



Ci sono dei giorni in cui Dino e Bruno esploreranno la stessa caverna?

<input type="radio"/>	No	<input type="radio"/>	Sì, nella caverna DARK.
<input type="radio"/>	Sì, di sabato.	<input type="radio"/>	Sì, trascorrono due giorni nella stessa caverna.

Soluzione. Non ci sono giorni in cui Dino e Bruno esplorano la stessa caverna. Le esplorazioni di Dino e Bruno sono riassunte dalla seguente tabella

Giorno	Dino	Bruno
lunedí	Gold	Stone
martedí	Ruby	Emerald
mercoledí	Emerald	Crystall
giovedí	Dark	Gold
venerdí	Sapphire	Ruby
sabato	Crystall	Dark
domenica	Stone	Sapphire

Anche questa è informatica! Si tratta della visita dei nodi di un albero binario, che è un tipo particolare di grafo, in ampiezza e in profondità.

Parole chiave e riferimenti: strutture dati ad albero, visita in ampiezza, visita in profondità


Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras della Germania e usato nel Kangourou dell'Informatica 2012/13.

Scambi (4 punti)

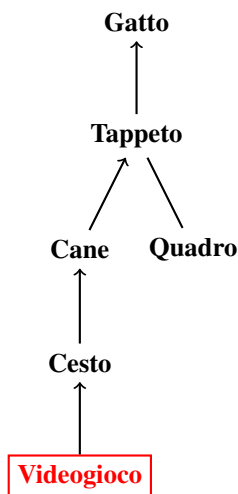
Kang vorrebbe tanto avere un gatto. Si collega allora a K-Bay e cerca di fare degli scambi in modo da avere il suo gatto, partendo dal videogioco che è disposto a scambiare. La tabella mostra gli scambi che Kang ha trovato su K-Bay: il nome identifica chi vuol fare lo scambio, prendendo l'oggetto che segue il nome e offrendo in cambio l'altro oggetto. Per esempio, Pete vorrebbe il videogioco e offre in cambio un pallone.

Trascinate le righe della tabella a sinistra nella tabella a destra in modo da ottenere la giusta sequenza di scambi per ottenere il gatto.

Pete	Videogioco	➔	Pallone
Jack	Videogioco	➔	Cesto
Lucy	Pallone	➔	Barchetta
Molly	Barchetta	➔	Motocicletta
Frank	Pallone	➔	Bicicletta
Steve	Cesto	➔	Barchetta
Mark	Cesto	➔	Cane
Sarah	Cane	➔	Pallone
Jody	Bicicletta	➔	Pallone
Lance	Cane	➔	Tappeto
Mary	Tappeto	➔	Motocicletta
Farida	Quadro	➔	Tappeto
Sam	Bicicletta	➔	Motocicletta
Mindy	Tappeto	➔	Gatto



Soluzione. Le soluzioni possono essere cercate costruendo un *albero* che ha come *radice* l'obiettivo finale dei baratti e ogni *nodo X* dell'albero ha come *figli* i sotto-alberi dei baratti aventi come obiettivo finale l'oggetto X. Se nell'albero così costruito esiste almeno un nodo con l'oggetto posseduto inizialmente, ogni cammino da questo nodo alla radice è una soluzione del problema. In questo caso, come si può vedere dalla figura, c'è un'unica soluzione.



Jack	Videogioco	➔	Cesto
Mark	Cesto	➔	Cane
Lance	Cane	➔	Tappeto
Mindy	Tappeto	➔	Gatto

Anche questa è informatica! La rappresentazione delle informazioni tramite alberi di relazioni o, più in generale, grafi, è estremamente comune in informatica: i nodi rappresentano oggetti fra cui esistono relazioni binarie, rappresentate da *archi* che li congiungono; spesso poi l'elaborazione consiste nella ricerca di cammini con determinate proprietà.

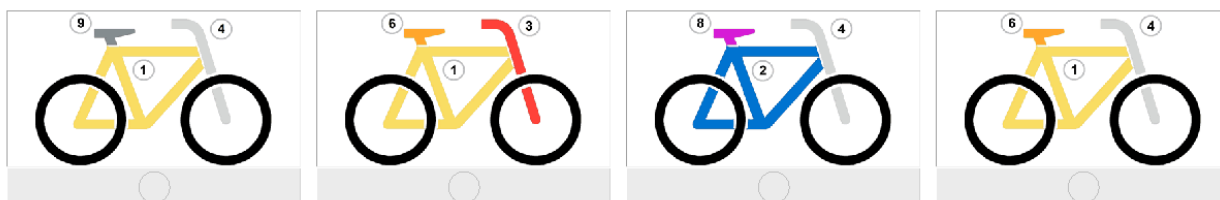
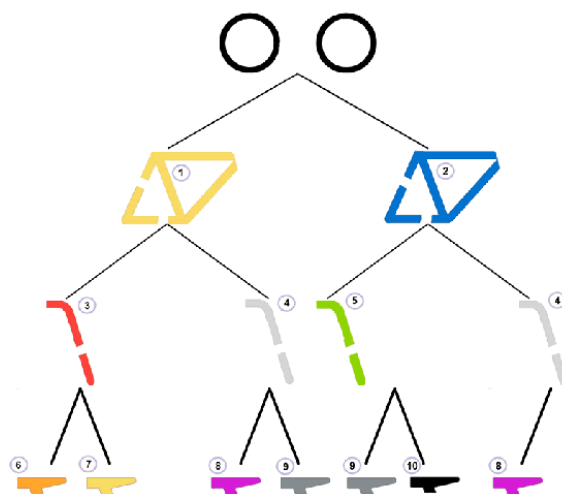
Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras della Francia e usato nel Kangourou dell'Informatica 2012/13.

Biciclette alla moda (5 punti)

Al giorno d'oggi tutti a Kang Town vogliono avere una bicicletta alla moda. Ma la polizia ha imposto un regolamento che specifica in che modo si possono colorare le biciclette. Per capire se una bicicletta soddisfa i vincoli della polizia potete fare riferimento al disegno in figura (un cosiddetto "albero di decisione").

Partendo dall'alto (dalla cosiddetta "radice"), un ciclista deve decidere passo-passo quale opzione preferisce e scendere lungo il "ramo" corrispondente.

Quale delle seguenti biciclette NON soddisfa i vincoli?



Soluzione. La quarta bicicletta (quella con i pezzi 1, 4 e 6) è l'unica non conforme al regolamento: dopo aver scelto il telaio 1 e il manubrio 4, le selle permesse sono solo la 8 e la 9, non la 6.

Anche questa è informatica! Un albero è una struttura comunemente usata in informatica. Questo in particolare è un *albero di decisione*: a seconda delle situazioni che si verificano man mano, non sono più ammesse tutte le opzioni, ma solo quelle consentite dal percorso seguito nell'albero.

Parole chiave e riferimenti: albero di decisione

Informazioni sul quesito. Il quesito è stato proposto dal gruppo Bebras dell'Austria e usato nel Kangourou dell'Informatica 2012/13.

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 $-2, -1, 0, 1, 2$, 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, ...).

Anche questa è informatica! 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.

Informazioni sul quesito. Il quesito è stato proposto dal Kangourou dell'Informatica 2008/09.

Una partita a tris (6 punti)

Sir Cross e Lady Circle stanno giocando una partita al gioco del tris (o "filetto"), che certamente tutti conoscete: vince chi per primo fa tris, in orizzontale o verticale o diagonale, e se nessuno fa tris la partita finisce in parità.

Ha iniziato Sir Cross, mettendo una croce in mezzo a uno dei lati; Lady Circle ha risposto disegnando un cerchio in uno degli angoli a fianco della croce, come mostrato nella figura qui a destra.

○	×	

Quale tra le seguenti mosse deve fare adesso Sir Cross per assicurarsi almeno il pareggio?

Soluzione. La mossa giusta per Sir Cross è:

o	x	
x		

Infatti a questo punto Lady Circle non ha alcuna mossa vincente, e pertanto Sir Cross si assicura almeno il pareggio — se giocherà bene, s'intende! Gli altri due casi vanno esclusi perché c'è sempre almeno una mossa che consentirebbe a Lady Circle di vincere.

Nel caso illustrato nella figura seguente (a sinistra), la mossa vincente di Lady Circle è indifferentemente una delle due riportate al centro e a destra della stessa figura:

o	x	x

o	x	x
o		

o	x	x
o		

mentre nel caso della figura seguente (a sinistra), l'unica possibilità di vittoria per Lady Circle è la mossa illustrata a destra:

o	x	
		x

o	x	
	o	
		x

Per inciso, oltre alle tre proposte nel quesito, Sir Cross aveva altre quattro possibili mosse (in questa situazione non c'è alcuna simmetria!), una soltanto delle quali gli sarebbe stata fatale, e precisamente quella della figura seguente (a sinistra), alla quale Lady Circle, per assicurarsi la vittoria, avrebbe potuto rispondere come nella figura a destra:

o	x	
		x

o	x	
		x
o		

Senza alcun dubbio il tris o “filetto” è universalmente conosciuto e ancora praticato da tanti ragazzi, sebbene sia altrettanto noto che si tratta di un gioco “alla pari”: nessuno dei due giocatori riesce a vincere se l’avversario non commette errori! Occorre soltanto fare un po’ di attenzione proprio nella fase iniziale. Ad esempio, se alla sua prima mossa Lady Circle avesse risposto in uno di questi due modi:

	x	
o		

	x	
o		

ovvero in uno dei due simmetrici, allora Sir Cross avrebbe facilmente vinto...

Anche questa è informatica! Questo gioco è stato oggetto non soltanto di articoli e libri interi ma, ovviamente, anche di analisi al calcolatore e persino del primo videogame, in grado di non perdere mai, progettato nel 1952 per il computer EDSAC. In verità, già un secolo avanti, Charles Babbage — che fu il primo a concepire l’idea di calcolatore *programmabile* — aveva progettato sulla carta un automa capace di giocare a *tit-tat-to* (questo il nome da lui usato per indicare il gioco; tra gli altri tuttora diffusi: *tic-tac-toe* o *ticktacktoe* o, con riferimento ai simboli grafici, *noughts and crosses*, che il grande poeta romantico William Wordsworth chiama *crosses* e *cyphers* in un suo preludio sui ricordi di scuola).

Contando come una sola tutte le posizioni che si possono ottenere l’una dall’altra per qualche simmetria (rotazioni e/o ribaltamenti), quelle possibili sono 765 e le partite 26830: ben poca cosa per un computer, decisamente più arduo compito un’elaborazione manuale. Come si dovrebbe procedere? Per costruire l’*albero di gioco* completo partiamo dall’alto, mettendo in *radice* lo *stato iniziale* (lo schema vuoto, ove inizierà chi mette la croce), e poi procediamo verso il basso, facendo in modo che ciascun nodo abbia come *figli* tutti gli stati che si possono originare con una mossa del giocatore al quale tocca muovere nello stato rappresentato da quel nodo. Ad esempio, i figli della radice saranno tre, poiché la croce può essere messa in sole tre posizioni essenzialmente differenti (al centro, o in uno degli angoli, o in mezzo a uno dei lati). Gli stati *finali* (in cui uno dei due giocatori ha fatto tris o tutte e nove le mosse sono state fatte) saranno le *foglie* dell’albero. E proprio dalle foglie dovremo partire, man mano risalendo, per analizzare completamente (o, come si dice, *risolvere*) il gioco, come tra poco vedremo.

Prima, però, una precisazione è doverosa. Abbiamo parlato di posizioni o stati, ossia in generale configurazioni del tavoliere di gioco (in cui va compresa l’informazione binaria che dice “a chi tocca muovere”, se non è già implicita). Se pensiamo a tutti questi stati collegati da archi, in modo tale che vi sia un arco dallo stato *x* allo stato *y* se e soltanto se con una mossa si può passare da *x* a *y*, allora in realtà non stiamo pensando a un albero ma più in generale a un *grafo* (orientato). Se questo grafo avesse dei cicli, allora il gioco potrebbe continuare all’infinito... Tuttavia, è più semplice e conveniente pensare a un albero, in cui più nodi distinti possono rappresentare la stessa posizione, se a questa si può arrivare mediante diverse sequenze di mosse a partire dallo stato iniziale.

L’analisi del gioco si basa su questo ragionamento: se in un certo stato tocca a me muovere, cercherò la mossa che mi farà guadagnare il massimo possibile; se invece tocca al mio avversario, cercherò quella che farà guadagnare a lui il massimo possibile: suppongo infatti che pure lui giochi, come me, al meglio delle proprie possibilità. Parto dunque dalle foglie dell’albero, attribuisco a ciascuna di esse un punteggio: -1 se perdo, 0 se la partita è patta, 1 se vinco. Poi risalgo via via fino alla radice, attribuendo a ciascun nodo (di cui ho già valutato tutti i figli) il *massimo* punteggio dei figli se in quel nodo tocca a me muovere, il *minimo* se tocca al mio avversario. Quando arrivo a valutare la radice, il gioco è risolto: se il suo punteggio è 1 io ho una strategia vincente, se è -1 ce l’ha il mio avversario, se è 0 la partita non potrà che finire in parità, se nessuno di noi due commetterà errori.

Si intuisce un’ipotesi, implicitamente fatta in questo discorso: che il gioco sia *a somma nulla*, vale a dire che uno svantaggio di uno dei due giocatori equivalga a un vantaggio di pari entità dell’altro giocatore; quindi il minimo guadagno di uno corrisponde al massimo dell’altro. L’utilità di tale ipotesi si apprezza ancor meglio se l’esplorazione dell’albero di gioco arriva soltanto fino a una certa profondità, a partire dallo stato attuale. Ciascuno dei due giocatori cercherà allora di *minimizzare* la propria *massima* perdita possibile (da cui il nome dell’algoritmo: *minimax*): questa idea sta alla

base di un teorema formulato nel 1928 da John von Neumann, uno dei padri dei moderni calcolatori elettronici, che tra il 1926 e il 1944 diede importanti contributi alla teoria economica e alla nascente *teoria dei giochi*.

Quali sono le altre caratteristiche salienti di questo tipo di giochi, in cui ad esempio rientrano anche la dama e gli scacchi?

- I giocatori sono due e muovono a turno, *alternandosi*;
- il gioco è *finito*: ossia, l'albero di gioco è finito, sia in ampiezza (in ogni stato il numero di mosse lecite è limitato), sia in altezza (prima o poi la partita termina);
- il gioco è *a informazione perfetta*: ossia, in ogni momento, in particolare prima della scelta della mossa, entrambi i giocatori conoscono lo stato completo del gioco — nulla è tenuto nascosto o lasciato al caso.

In queste ipotesi il gioco si dice *strettamente determinato*: o uno dei due giocatori ha almeno una strategia che assicura la vittoria (cioè: ad ogni suo turno ha almeno una mossa che può portarlo poi a vincere, qualunque strategia adotti l'avversario), o entrambi hanno almeno una strategia che assicura il pareggio. In effetti, il procedimento che abbiamo delineato permette di decidere quale si verifica dei tre casi possibili già previsti da Ernst Zermelo (celebre per i suoi contributi alla teoria assiomatica degli insiemi) nel 1912: salvo errori, vince chi inizia o chi risponde o è patta teorica.

C'è un "però", come facilmente si intuisce da quanto accennato... L'albero di gioco può essere gigantesco, perfino in modo inimmaginabile: si pensi a Othello o agli scacchi, dove il numero di posizioni diverse pare abbia 29 e 47 cifre decimali, rispettivamente — dunque enormemente più grande di 765 — figuriamoci il numero di partite che possono essere giocate! Come può agire allora un programma che abbia la pretesa di combattere contro un avversario? Anziché arrivare alle foglie, esplora l'albero dalla posizione attuale fino a una certa profondità, cercando di attribuire in modo efficace (e questo è un aspetto critico) un punteggio agli stati raggiunti, senza proseguire oltre (salvo casi particolari). In più, tenta di "potare" l'albero, evitando di scendere lungo rami che non possono migliorare la valutazione finora fatta della posizione attuale. Non sempre è possibile assegnare un punteggio a uno stato che non sia finale, rinunciando ad indagare avanti: si pensi proprio al "filetto"; in alcuni casi può essere invece relativamente semplice, quando si riesce a quantificare bene il vantaggio di un giocatore in base al materiale o alla posizione. Si tratta comunque di una valutazione ispirata da criteri euristici.

Un'ultima curiosità: la dama giocata sulla tradizionale scacchiera 8×8 è stata risolta — a favore della parità — nel 2007, dopo 18 anni di calcoli da parte di qualche centinaio di computer. (Si tratta del più grande gioco per cui sia stato trovato un risultato di questo tipo: il numero di posizioni ha 21 cifre... quantomeno decine di milioni di volte più piccolo di quello stimato per Othello!) Ma questo fatto, ovviamente, non toglie nulla all'emozione di una partita a dama!

Parole chiave: teoria dei giochi, gioco strettamente determinato, albero di gioco, algoritmo *minimax*.

Informazioni sul quesito. Il quesito è stato proposto dal Kangourou dell'Informatica 2009/10.