ISTITUTO TECNICO INDUSTRIALE "E. Fermi"
GARA NAZIONALE ABACUS 2012
PROVA PRATICA
20 Aprile 2012
INFORMATICA

Durante la manifestazione della partita a scacchi si vuole intrattenere il pubblico con spettacoli e giochi, più precisamente si vuole utilizzare la scacchiera della piazza per organizzare un gioco per i ragazzi delle scuole medie ed elementari.
Vengono formate due squadre che dovranno comporre 4 parole ciascuna a turno da inserire nelle righe della scacchiera (prima parola nella prima riga, seconda parola nella seconda riga e così via). La lunghezza delle parole non può superare gli otto caratteri corrispondenti a una riga della scacchiera, ogni parola inizia sempre dalla colonna più a sinistra (colonna zero).
Ciascuna squadra estrae 8 lettere da un sacchetto contenente 128 lettere dell'alfabeto così distribuite:

12 'A', 'E', 'I', 'O'
4 'U',
7 'C', 'R', 'S', 'T';
6 'L', 'M', 'N';
4 'B', 'D', 'F', 'G', 'P', 'V';
2 'H', 'Q' e 'Z'.

Ad ogni lettera è associato il seguente punteggio:

1 punto per ogni A, C, E, I, O, R, S, T;
2 punti per ogni L, M, N;
3 punti per ogni P;
4 punti per ogni B, D, F, G, U, V;
8 punti per ogni H e Z;
10 punti per ogni Q.

Il compito della squadra è quello di trovare una parola, appartenente a un dizionario di parole fornito in precedenza, che restituisca il punteggio più alto sommando tutti i punti delle lettere. Per complicare la ricerca nella riga interessata comparirà una vocale casuale in una certa posizione e un bonus in un'altra posizione diversa dalla precedente. Il compito del bonus è quello di fungere da moltiplicatore per la lettera nella sua posizione (ad esempio se nella casella 4 compare il bonus x3 significherà che la lettera in quella posizione avrà un valore pari al triplo del suo valore originario).
La parola da comporre può essere più corta di 8 caratteri, può non contenere la cella bonus ma deve contenere il carattere vincolo.
Nel caso in cui una squadra non dovesse riuscire a comporre nessuna parola potrà sostituire tutte le sue 8 lettere con altre 8 (questa operazione comporterà una penalità di 5 punti da togliere al totale finale e potrà essere fatta una sola volta), oppure proseguire accettando un punteggio pari a 0 per quel particolare turno.
Al termine di ogni mossa ciascuna squadra pescherà dal sacchetto un numero di lettere tale che, sommato a quelle rimaste in proprio possesso, consenta di averne nuovamente 8 a disposizione.

Si vuole verificare con una simulazione software se le regole del gioco così come impostate sono sufficienti a rendere la gara abbastanza avvincente e fattibile. Viene quindi chiesto di produrre un software che simuli la competizione tra due ipotetiche squadre che siano in grado di trovare ogni volta le parole più vantaggiose per arrivare ad un punteggio finale migliore della squadra avversaria.

Le parole del dizionario sono presenti in un file XML secondo il formato sotto riportato, non sono necessariamente ordinate in ordine alfabetico e possono essere anche più lunghe di otto caratteri:

```
<?xml version="1.0"?>
<words>
        <word>io</word>
        <word>casa</word>
        <word>bacino</word>
        ….
        <word>asciugamano</word>
</words>
```

Tra le problematiche da affrontare per costruire la simulazione vi è quella di riuscire a trovare (abbastanza velocemente) tutte le parole che si possono comporre con le 8 lettere a disposizione e tra queste scegliere quella che possa massimizzare il punteggio secondo i vincoli imposti nella riga che si deve utilizzare sulla scacchiera.
Per risolvere il problema di individuare velocemente gli anagrammi, gli analisti hanno utilizzato la seguente strategia **che dovrà essere implementata** nella simulazione.

Strategia:

Si vuole utilizzare una struttura dati che consenta di individuare tutti gli anagrammi presenti in un dizionario da un insieme di n lettere facendo pochissimi confronti (nell'ordine di O(n)). Per far questo carichiamo le parole dal file xml all'interno di questa struttura nel seguente modo:

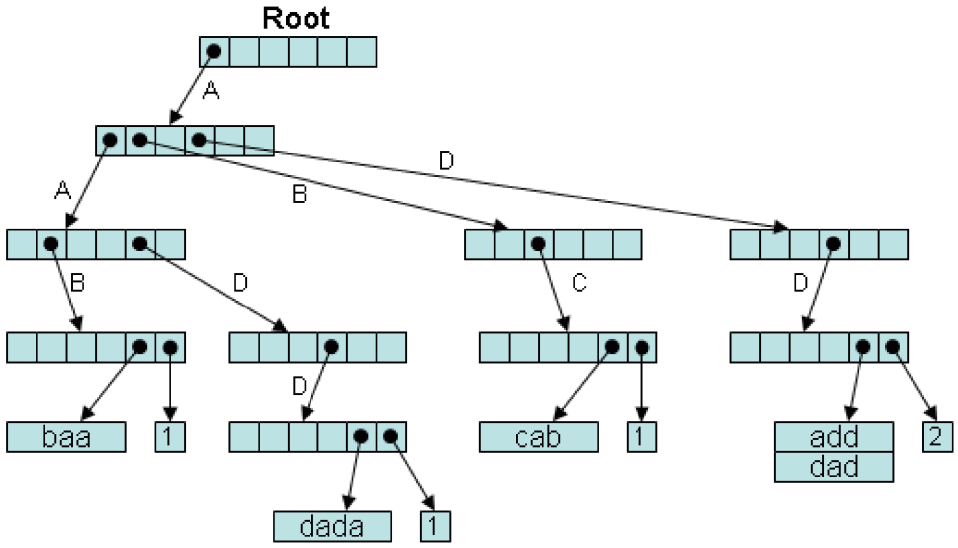
- data una parola del dizionario ad esempio 'fabbrica' creiamo una chiave per questa parola ordinando in modo crescente le sue lettere ottenendo così 'aabbcifr' (nota che questo è un anagramma della parola iniziale), nota che tutti gli anagrammi di 'fabbrica' produrranno la stessa chiave 'aabbcifr';
- quindi inseriamo tutte le parole del dizionario aventi la stessa chiave in una lista che contiene quindi tutti gli anagrammi validi.

Per poter individuare velocemente la lista degli anagrammi di una chiave utilizziamo un Trie ossia un albero digitale di ricerca, una struttura ad albero così composta: ogni nodo dell'albero può avere fino a 21 figli che rappresentano a loro volta un Trie, nel nodo sono inoltre presenti altre due informazioni: una lista L che corrisponde a tutti gli anagrammi del nodo e il numero di elementi presenti nella lista L.
La radice del trie rappresenta quindi le parole formate da zero caratteri, la sua lista L sarà vuota e il numero di elementi sarà 0, partendo da sinistra, il primo figlio indicizza tutte le parole che cominciano con la lettera 'A', il secondo figlio tutte le parole che cominciano con la lettera 'B' e così via fino ad arrivare al ventunesimo figlio che indicizza tutte quelle che cominciano per 'Z', a

questo primo livello le liste di parole ammissibili avranno al più un elemento (es. nel primo figlio potrà esserci la parola ‘A’).
A questo punto si ripete la definizione in modo ricorsivo e per ciascun nodo del primo livello ci potranno essere 21 figli ognuno indicante le parole che proseguono con una determinata lettera.
Per esemplificare il tutto vediamo un esempio supponendo di avere un alfabeto con solo 4 lettere (A, B, C, D) e supponiamo di avere il seguente file XML di parole da memorizzare: add, baa, cab, dad, dada

```
<?xml version=”1.0”?>
<words>
        <word>add</word>
        <word>baa</word>
        <word>cab</word>
        <word>dad</word>
        <word>dada</word>
</words>
```

All’inizio avremo un trie vuoto con una radice senza figli e senza altre informazioni.
Leggendo la parola ‘add’ viene calcolata la sua chiave che corrisponde a ‘add’ e viene inserita la parola nel trie creando il primo figlio della radice quindi creando il quarto figlio del nodo al primo livello e poi ancora il quarto figlio del nodo al secondo livello, questo nodo non avrà figli ma nella sua lista ci sarà la parola ‘add’.
La seconda parola ‘baa’ genera una chiave che corrisponde a ‘aab’ che verrà inserita nel trie seguendo il percorso Root → Primo Figlio → Primo Figlio → Secondo Figlio.
La terza parola ‘cab’ genera una chiave che corrisponde a ‘abc’ che seguirà il percorso Root → Primo Figlio → Secondo Figlio → Terzo Figlio.
La quarta parola ‘dad’ genera una chiave che corrisponde a ‘add’ che seguirà il percorso Root → Primo Figlio → Quarto Figlio → Quarto Figlio.
e così via fino ad ottenere il Trie disegnato qui sotto.



Come si può notare il nodo in basso più a destra contiene i due anagrammi possibili per le parole del dizionario dato avendo a disposizione come lettere due D e una A.

Grazie a questa struttura sarà molto veloce individuare tutti gli anagrammi possibili di un insieme di caratteri.
A questo punto il candidato può sviluppare la simulazione secondo le linee guida indicate:
Bisognerà rappresentare il campo di gioco come una scacchiera 8x8, caricare nel trie il dizionario leggendolo dal file XML di esempio fornito, definire un "Distributore" per consegnare le prime 8 lettere casuali e le successive ogni qualvolta ci sarà bisogno, indicare prima di ciascuna mossa la lettera vincolante e la sua posizione oltre al bonus e alla sua posizione quindi effettuare la mossa scegliendo la parola "migliore" da inserire nella riga.
Tutte le strutture dati dinamiche utilizzate per realizzare il Trie e le liste associate devono essere implementate dal candidato senza utilizzare classi o librerie già presenti nel framework a questo scopo.
L'output della simulazione può avvenire anche su console producendo delle stringhe simili a queste:

Benvenuti al gioco "PAROLANDO"
Giocatore A: A, E, R, T, U, G, O, V
Vincolo 1^ Mossa: x2 - A - - - - -
Mossa Giocatore A: T R A V E
Punti Mossa: 12
Punti totali Giocatore A: 12

Giocatore B: B, U, T, T, I, O, L, V
Vincolo 1^ Mossa: O - - - - x3 - -
Mossa Giocatore B: O T T O
Punti Mossa: 6
Punti totali Giocatore B: 6

Giocatore A: A S B U U G O Z
Vincolo 2^ Mossa: - - - - E - x3 -
Mossa Giocatore A: Cambio Tutte Le Lettere
…...

Saranno oggetto di valutazione

- grado di completezza della soluzione proposta, per permettere la valutazione di soluzioni con implementazione non completa;
- grado di correttezza del programma (correttezza dell'uscita prodotta in corrispondenza di vari test-set di dati in ingresso soprattutto nell'implementazione delle funzioni di inserimento, ricerca nel trie e individuazione della parola migliore);
- chiarezza della soluzione implementata, esplicitata attraverso commenti e l'attribuzione di nomi appropriati a classi, proprietà e metodi meglio se in inglese.
- una eventuale soluzione corretta prodotta con interfaccia grafica sarà valutata maggiormente.
- robustezza del codice (grado di reazione a malfunzionamenti).
- output e/o l'interfaccia del programma in lingua italiana.

I dizionari delle parole, di varie dimensioni, sono memorizzati nel desktop della postazione.