

Simulazione di Prova Intermedia del 11/01/2022

Questa simulazione di prova intermedia è costituita da due esercizi. Potranno essere consegnati al docente solo gli esercizi che avranno superato i nostri JUnit Test. I test svolti dagli studenti non saranno presi in considerazione per la valutazione. La consegna di almeno un esercizio è condizione necessaria, ma non sufficiente, per passare la prova.

Per la valutazione saranno presi in considerazione aspetti di “buona programmazione”, “pulizia del codice” ed efficienza. Ad es.: formattazione corretta del codice, rendere il codice modulare aggiungendo ove necessario altri metodi rispetto a quelli richiesti dall’esercizio, soprattutto se questi rendono il codice più pulito e leggibile, o se evitano duplicazione di codice. Inoltre, non ci devono essere warning nel codice scritto.

IMPORTANTE: seguire attentamente le specifiche per quanto riguarda i nomi dei metodi e la firma dei metodi, altrimenti i test automatici falliranno rendendo il compito insufficiente.

CONSEGNA ESERCIZI:

Entro il termine ultimo previsto per la consegna dello scritto, gli studenti che intendono consegnare provvedono a caricare il sorgente (o i sorgenti) **.javamm** attraverso l’apposita attività “compito”, disponibile nella pagina MOODLE del corso al seguente link:

<https://e-l.unifi.it/mod/assign/view.php?id=795019>

Fino allo scadere del tempo, lo studente potrà apportare modifiche al proprio lavoro. Non saranno accettate consegne effettuate in ritardo, o con modalità diverse da quelle definite dal docente. All’orario stabilito ad inizio compito il docente dichiara finita la prova e chiude la sessione.

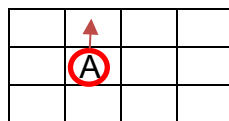
Esercizio Java n. 1: Caccia al Tesoro

- Esercizio estratto dal Compito III Appello del 15/07/2021 -

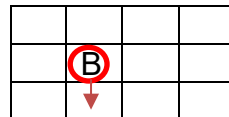
Sia M una matrice di caratteri di dimensione $m \times n$ (con $m > 0$, $n > 0$), i cui elementi possono assumere uno dei seguenti possibili valori: 'A', 'B', 'S', 'D', '?', 'T', 'X'.

La caccia al tesoro consiste nel visitare gli elementi della matrice partendo dalla cella di posizione $(0,0)$, con l'obiettivo di arrivare al tesoro che si trova nella cella di posizione $(m-1, n-1)$, contenente il carattere 'X'. Gli spostamenti all'interno della matrice avvengono in base alle seguenti regole:

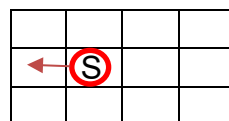
- Carattere 'A' (Alto): la ricerca prosegue spostandosi di una cella verso l'Alto (ovvero decrementando l'indice di riga):



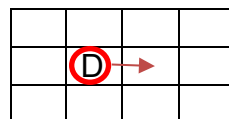
- Carattere 'B' (Basso): la ricerca prosegue spostandosi di una cella verso il Basso (ovvero incrementando l'indice di riga):



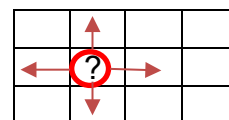
- Carattere 'S' (Sinistra): la ricerca prosegue spostandosi di una cella verso Sinistra; (ovvero decrementando l'indice di colonna):



- Carattere 'D' (Destra): la ricerca prosegue spostandosi di una cella verso Destra (ovvero incrementando l'indice di colonna):



- Carattere '?' (Jolly): è il carattere Jolly, che può assumere il valore di tutti i precedenti quattro caratteri ('A', 'B', 'S', 'D'). Quindi la ricerca prosegue esplorando tutte le quattro possibili direzioni corrispondenti ai quattro caratteri, ovvero si prosegue la ricerca interpretando il carattere '?' come se fosse il carattere 'A', poi come se fosse il carattere 'B', poi come se fosse il carattere 'S' e infine come se fosse il carattere 'D' (l'ordine in cui si esplorano le 4 direzioni non è rilevante):



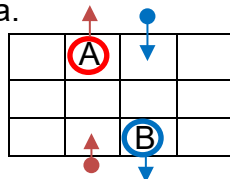
I caratteri rimanenti ('T' ed 'X') hanno il seguente significato:

- Carattere 'T' (Trappola): sei finito in una Trappola, la tua ricerca in questo cammino fallisce senza aver trovato il tesoro!
- Carattere 'X' (Tesoro): complimenti, tesoro trovato!

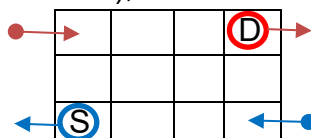
La caccia al tesoro termina con successo se esiste almeno un cammino che, partendo dalla cella di posizione $(0,0)$, arrivi alla cella di posizione $(m-1,n-1)$ contenente il valore 'X'. Altrimenti la caccia al tesoro fallisce.

NOTA BENE:

- La matrice contiene 1 sola cella contenente il carattere 'X', in posizione $(m-1,n-1)$.
- La ricerca nella matrice è da intendersi come circolare, ovvero
 - Uno spostamento verso il basso ('B') da una cella di riga $r=m-1$ (ultima riga della matrice) corrisponde ad uno spostamento nella cella di riga 0 (prima riga della matrice), e viceversa.



- Uno spostamento verso destra ('D') da una cella di colonna $c=n-1$ (ultima colonna della matrice) corrisponde ad uno spostamento nella cella di colonna 0 (prima colonna della matrice), e viceversa.



Scrivere un metodo Java-- chiamato `cacciaAlTesoro` che, data una matrice M di caratteri di dimensione $m \times n$ definita come precedentemente descritto, restituisca `true` se esiste almeno un cammino che, partendo dalla cella di posizione $(0,0)$, arrivi alla cella contenente il valore 'X' in posizione $(m-1,n-1)$, `false` altrimenti.

Esempio 1: se $M = \begin{pmatrix} B & A & T & B \\ D & A & A & T \\ S & D & B & B \\ S & A & D & X \end{pmatrix}$, il metodo deve restituire `true` poiché il tesoro ('X')

viene raggiunto da questo cammino: $B \rightarrow D \rightarrow A \rightarrow A \rightarrow A \rightarrow D \rightarrow B \rightarrow D \rightarrow X$

Esempio 2: se $M = \begin{pmatrix} D & D & D & B \\ D & A & S & S \\ S & B & T & S \\ S & S & T & X \end{pmatrix}$, il metodo deve restituire `false` poiché il tesoro ('X')

non viene raggiunto da nessun cammino (entro in un loop infinito).

Esempio 3: se $M = \begin{pmatrix} B & T & D & T \\ ? & D & A & T \\ S & B & ? & S \\ S & S & T & X \end{pmatrix}$, il metodo deve restituire `true` poiché il tesoro ('X')

viene raggiunto da questo cammino: $B \rightarrow B$ ('?' interpretato come 'B') $\rightarrow S \rightarrow S \rightarrow S$ ('?' interpretato come 'S') $\rightarrow B \rightarrow S \rightarrow S \rightarrow X$.

In questo stesso Esempio 3, interpretando il primo '?' come 'D' avrei ottenuto il cammino: $B \rightarrow D$ ('?' interpretato come 'D') $\rightarrow D \rightarrow A \rightarrow D \rightarrow T$ (finisco in Trappola).

DIFFICOLTA' RIDOTTA: E' possibile svolgere l'esercizio assumendo che la matrice M NON contenga il carattere '?' (Jolly). In questo caso i **JUnit Test** che devono essere superati sono quelli della classe **CacciaAITesoroRidottaTest** (non considerare quelli della classe CacciaAITesoroStandardTest che ovviamente falliranno).

- Suggerimenti per DIFFICOLTA' RIDOTTA:
 - i) E' possibile sviluppare una soluzione iterativa o ricorsiva (nessuna preferenza).
 - ii) Modificare la matrice di partenza per tener traccia dei caratteri già incontrati, che quindi non devono essere esplorati nuovamente (altrimenti rischiate di avere dei loop infiniti – si veda l'Esempio 2).
 - iii) Nel caso di soluzione ricorsiva: richiamare ricorsivamente un metodo con tre parametri formali: la matrice, la riga e la colonna del prossimo carattere da visitare.

DIFFICOLTA' STANDARD (FACOLTATIVA): Gli studenti che vogliono aumentare la difficoltà e la complessità dell'esercizio (e quindi anche ottenere una valutazione migliore in caso di soluzione corretta), possono svolgere l'esercizio nella sua forma originale, ovvero considerando che la matrice M possa contenere anche caratteri '?' (Jolly). In questo caso i **JUnit Test** che devono essere superati sono quelli della classe **CacciaAITesoroStandardTest** (se questi test saranno superati, lo saranno comunque anche quelli dell'altra classe CacciaAITesoroRidottaTest essendo questi ultimi un sottoinsieme dei test standard).

- Suggerimenti per DIFFICOLTA' STANDARD:
 - i) E' consigliabile sviluppare una soluzione ricorsiva in cui si richiama ricorsivamente un metodo con tre parametri formali: la matrice, la riga e la colonna del prossimo carattere da visitare.
 - ii) Per catturare il comportamento del caso carattere Jolly '?', ci saranno quattro chiamate ricorsive (una per ogni direzione) legate da un opportuno operatore logico (restituisco `true` se in almeno uno dei quattro cammini trovo il tesoro, `false` altrimenti).
 - iii) Modificare la matrice di partenza per tener traccia dei caratteri già incontrati, che quindi non devono essere esplorati nuovamente (altrimenti rischiate di avere una serie infinita di chiamate ricorsive – si veda l'Esempio 2).

NOTA BENE:

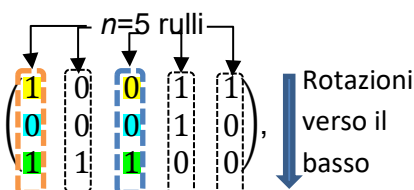
- Gli studenti dovranno consegnare per questo esercizio solo 1 sorgente relativo alla soluzione con “difficoltà ridotta” oppure relativo alla soluzione con “difficoltà standard”.
- Nello svolgere l'esercizio NON devono essere utilizzati i metodi `clone`, o `arraycopy`, o metodi della classe `Arrays`. L'utilizzo di tali metodi renderà l'esercizio automaticamente insufficiente.

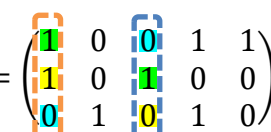
Esercizio Java n. 2: Slot Machine

- Esercizio estratto dal Compito IV Appello del 09/09/2021 -

Sia S una matrice di interi di dimensione $m \times n$, i cui elementi possono assumere due possibili valori: 0 oppure 1. Si assuma che la matrice sia composta da almeno 3 righe e che m sia dispari (quindi m può assumere i valori 3, 5, 7, 9, ...), e che abbia almeno una colonna (quindi $n \geq 1$). Inoltre, sia r un array di dimensione n a valori interi maggiori o uguali a zero.

La matrice S rappresenta in modo semplificato una Slot Machine meccanica costituita da una serie di n rulli, ognuno contenente m simboli a valori 0 oppure 1. Una *rotazione* della Slot Machine S in base ad r consiste nel far ruotare (ciclicamente) verso il basso il contenuto di ogni colonna j della matrice del numero di passi indicato nella posizione j dell'array r (ovvero $r[j]$).

Ad esempio, sia $S =$ , e sia $r = (1 \ 3 \ 8 \ 2 \ 0)$.

La rotazione di S in base a r produce la seguente matrice: $S' =$ , infatti:

- la prima colonna viene ruotata (ciclicamente) verso il basso di 1 posizione;
- la seconda colonna viene ruotata (ciclicamente) verso il basso di 3 posizioni (equivalente ad uno spostamento di 0 posizioni);
- la terza colonna viene ruotata (ciclicamente) verso il basso di 8 posizioni (equivalente ad uno spostamento di 2 posizioni);
- la quarta colonna viene ruotata (ciclicamente) verso il basso di 2 posizioni;
- la quinta colonna viene ruotata (ciclicamente) verso il basso di 0 posizioni (quindi non viene ruotata).

Consegna 1 [OBBLIGATORIA]:

Scrivere un metodo Java-- chiamato `ruota` che, dati una matrice S ed un array r definiti come precedentemente descritto, restituisca una nuova matrice corrispondente alla rotazione di S in base ad r .

I JUnit Test che devono essere superati per la Consegna 1 sono quelli della classe **SlotMachineTest_Ruota** (non considerare gli altri test che ovviamente falliranno).

Suggerimenti: Attenzione a non modificare direttamente la matrice di partenza (potreste sovrascrivere dei valori): è consigliabile creare una nuova matrice in cui andare ad inserire i valori della matrice di partenza nelle posizioni corrette.

Consegna 2 [FACOLTATIVA – da svolgere solo dopo Consegna 1]: Prerequisito: questa consegna sarà valutata solo se saranno superati i JUnit Test previsti per la Consegna 1.

Si supponga che l'unica combinazione vincente (*Jackpot!*) della Slot Machine sia quella in cui tutti i valori della riga centrale siano uguali ad 1. Ad esempio: se $m=3$ si vince il Jackpot se la riga di indice 1 (quella centrale) contiene tutti 1; se $m=5$ si vince il Jackpot se la riga di indice 2 (quella centrale) contiene tutti 1; etc...

Esempio di combinazione vincente: $S = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \leftarrow \text{Jackpot!}$

Scrivere un metodo Java-- chiamato `controllaVincita` che, data una matrice S definita come precedentemente descritto, restituisca `true` se la riga centrale della matrice S contiene tutti valori uguali ad 1 (*Jackpot!*), `false` altrimenti.

I JUnit Test che devono essere superati per la Consegna 2 sono quelli della classe **SlotMachineTest_ControllaVincita** (non considerare quelli di `SlotMachineTest_PartitaCompleta` che ovviamente falliranno).

Consegna 3 [FACOLTATIVA – da svolgere solo dopo Consegna 1 e 2]: Prerequisiti: questa consegna sarà valutata solo se saranno superati i JUnit Test previsti per le Consegne 1 e 2.

Si supponga che una partita completa alla Slot Machine consista nell'effettuare un massimo di k rotazioni (con $k \geq 1$) consecutive a partire dalla matrice iniziale S , cercando di vincere il Jackpot. Tutte le rotazioni sono fatte in base allo stesso array r . Scrivere un metodo Java-- chiamato `partitaCompleta` che, dati una matrice S , un array r ed un intero k definiti come precedentemente descritto, restituisca `true` se durante la partita si riesce a vincere il Jackpot, `false` altrimenti.

I JUnit Test che devono essere superati per la Consegna 3 sono quelli della classe **SlotMachineTest_PartitaCompleta**.

Suggerimenti: E' consigliabile adottare una soluzione ricorsiva, che richiama ricorsivamente lo stesso metodo `partitaCompleta` considerando come nuova matrice quella risultante dall'ultima rotazione, lo stesso array r e considerando un lancio in meno. Utilizzare naturalmente i metodi `ruota` e `controllaVincita` precedentemente sviluppati.