

PROGETTO “IL VIAGGIO DI UNA APE”

IL PROBLEMA

Un ape si muove in un **campo** alla ricerca di fiori dove raccogliere nettare per produrre miele per il suo alveare. Possiamo rappresentare il campo come un insieme di prati fioriti adiacenti come illustrato di seguito.

Dati quattro interi x_0, y_0, x_1, y_1 denotiamo con $R(x_0, y_0, x_1, y_1)$ il **rettangolo** (\mathbb{Z} è l'insieme degli interi)

$$R(x_0, y_0, x_1, y_1) = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} : x_0 \leq x \leq x_1, y_0 \leq y \leq y_1\}$$

Un prato fiorito è un rettangolo $R(x_0, y_0, x_1, y_1)$ in cui ogni punto (x, y) è occupato da un fiore a cui è associato un valore intero $val(x, y)$ (eventualmente negativo) che rappresenta la qualità del nettare estraibile da quel fiore; un valore negativo denota un miele di scarsa qualità. Due prati possono sovrapporsi, anche solo parzialmente. Se un fiore appartiene a due prati il suo valore è dato dalla somma algebrica dei valori che il fiore ha in ciascun prato a cui appartiene.

Due prati P_1 e P_2 sono **adiacenti** se la loro intersezione è non vuota. Un campo è un insieme di $n \geq 1$ prati $\{P_1, P_2, \dots, P_n\}$ tali che P_1 è adiacente a P_2 , P_2 è adiacente a P_3 , P_3 è adiacente a P_4 , etc.

L'ape, inoltre, può muoversi da un fiore in posizione $p = (x_p, y_p)$ ad un fiore in posizione $q = (x_q, y_q)$ se e solo se le seguenti condizioni sono soddisfatte:

- $y_q = y_p + 1$;
- $x_q = x_p - 1$, oppure $x_q = x_p$, oppure $x_q = x_p + 1$.

In questo caso, q è raggiungibile da p . Un **viaggio** da p a q è una sequenza di $n \geq 1$ fiori $p = p_1, \dots, p_n = q$ tale che ogni p_{i+1} è raggiungibile da p_i . Il valore del viaggio è dato dalla somma dei valori associati ai fiori che lo

compongono, ossia da: $\sum_{i=1}^n val(p_i)$

Assumete che, nel decidere come muoversi, la nostra ape sia in grado di valutare il rapporto tra il valore di un fiore e la distanza che deve compiere per raggiungerlo e che ad ogni passo essa scelga il fiore per cui tale rapporto è maggiore. Ad esempio, se l'ape inizia il suo viaggio dal fiore $p_1 = (3, 1)$ del campo descritto in Disegno 1 ha tre scelte alternative:

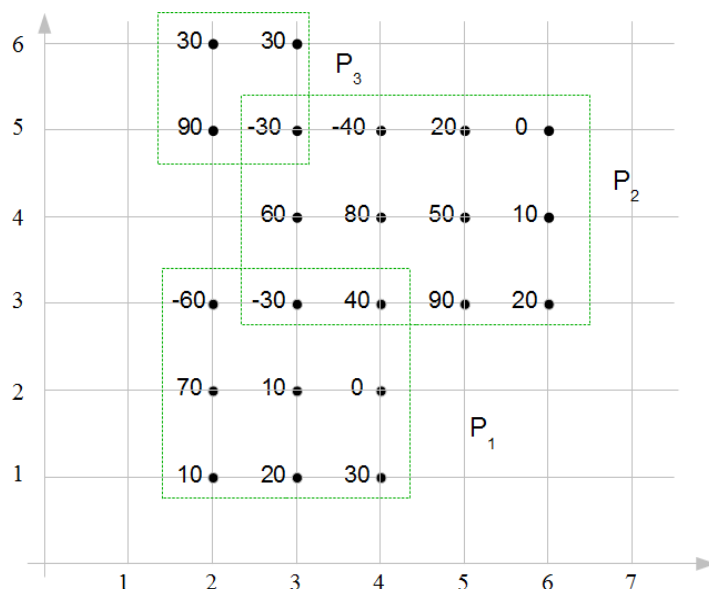
1. muoversi verso il fiore (4,2) – il valore di questo fiore è 0 e la sua distanza da p_1 è $\sqrt{2}$;
2. muoversi verso il fiore (3,2) – il valore di questo fiore è 10 e la sua distanza da p_1 è 1;
3. muoversi verso il fiore (2,2) – il valore di questo fiore è 70 e la sua distanza da p_1 è $\sqrt{2}$;

Poiché $70/\sqrt{2} = 49,49 > 10/1 = 10 > 0/\sqrt{2} = 0$, l'ape deciderà di muoversi verso il fiore $p_2 = (2, 2)$. Una volta in p_2 l'ape si sposterà in $p_3 = (3,3)$ perché $-30/\sqrt{2} = -21,21 > -60/1 = -60$. Da p_3 si muoverà in $p_4 = (3,4)$ perché $60/1 = 60 > 80/\sqrt{2} = 56,56$ e così via.

Si richiede di implementare una serie di classi che permettano di gestire un viaggio di un ape attraverso il campo. In particolare, si richiede di implementare:

- una classe **Grass** per rappresentare un prato fiorito

- una classe **Field** che gestisce un insieme di prati adiacenti ed i viaggi dell'ape.



Disegno 1: Un campo costituito da tre prati. I valori indicati per i fiori che appartengono a più di un campo rappresentano somme algebriche di valori che tali fiori hanno nei singoli campi

IMPLEMENTAZIONE

La classe **Grass** deve fornire i seguenti metodi:

- Grass(int x0, int y0, int x1, int y1, String fileName).**

Un costruttore che crea un nuovo prato fiorito leggendo i valori dei fiori in esso contenuto dal un file il cui nome è **fileName**. Assumete che il file contenga i valori dei fiori riga per riga (si vedano gli esempi allegati al progetto).

L'unica difficoltà nell'implementazione della classe **Grass** consiste nel decidere come rappresentare i valori associati ai fiori che esso contiene. Una possibile soluzione è quella di memorizzare i valori dei fiori in una matrice bidimensionale di opportune dimensioni. Ad esempio, il prato P_2 del campo in Disegno 1 può essere rappresentato dalla seguente matrice bidimensionale:

	0	1	2	3
0	-30	-40	20	0
1	60	80	50	10
2	-30	40	90	20

Osserviamo che:

- P_2 è un rettangolo $R(3, 3, 6, 5)$ ed ha 3×4 fiori con $3 = 5 - 3 + 1 = y_1 - y_0 + 1$ e $4 = 6 - 3 + 1 = x_1 - x_0 + 1$. In generale, il numero di fiori contenuti in un rettangolo $R(x_0, y_0, x_1, y_1)$ è $(y_1 - y_0 + 1) \times (x_1 - x_0 + 1)$. Possiamo quindi rappresentare i valori dei fiori nel prato con una matrice di dimensione $(y_1 - y_0 + 1) \times (x_1 - x_0 + 1)$.

- Inoltre, se disponiamo i valori nella matrice riga per riga, possiamo stabilire i seguenti legami tra un elemento della matrice ed il fiore di cui questo elemento rappresenta il valore:
 - $A[i, j]$ è il valore del fiore $(x_0 + j, y_1 - i)$;
 - viceversa, se il fiore (x, y) appartiene al prato $R(x_0, y_0, x_1, y_1)$ il suo valore è dato da $A[y_1 - y, x - x_0]$. Se invece (x, y) non appartiene al prato il suo valore è zero.

Ovviamente, avete anche bisogno di quattro interi per rappresentare le coordinate x_0, y_0, x_1 e y_1 del rettangolo che costituisce il vostro prato.

- **boolean isIn(int x, int y).**
Restituisce **true** se il fiore appartiene al prato falso altrimenti.
- **int getValue(int x, int y).**
Restituisce il valore di un fiore (x, y) se questo appartiene al prato, **maxint** altrimenti.
- **boolean adjacent(int a0, int b0, int a1, int b1).**
Restituisce **true** solo se il prato in esame è adiacente (e quindi ha intersezione non nulla) con un prato il cui rettangolo è $R(a_0, b_0, a_1, b_1)$; altrimenti restituisce **false**.

La classe **Field** invece deve fornire i seguenti metodi:

- **Field().**
Crea un nuovo campo vuoto.
La classe **Field** deve utilizzare una qualche struttura dati dinamica per rappresentare i prati di cui il campo è composto.
- **addGrass(Grass g).**
Aggiunge il prato g al campo. Se il campo è non vuoto, g può essere aggiunto al campo solo se adiacente all'ultimo prato inserito. Il prato g può essere comunque aggiunto in un campo vuoto.
- **int getValue(int x, int y).**
Restituisce il valore di un fiore (x, y) . Un fiore può appartenere a più di un prato del campo. In questo caso il suo valore è dato dalla somma algebrica dei valori che il fiore ha in ciascun prato a cui appartiene. Se il fiore (x, y) non appartiene a nessun prato del campo questo metodo deve restituire **MAX_VALUE.**
- **int[] journey(int x, int y, int steps)**
Restituisce un viaggio di **steps** passi a partire dal fiore $p = (x, y)$ secondo la logica descritta nella sezione precedente. Il metodo deve restituire un array di interi di dimensione 3 che memorizza le coordinate del fiore raggiunto dall'ape ed il valore complessivo del viaggio.

CONSEGNA DEL PROGETTO

Inviare (per email) al docente una cartella compressa contenente:

- il sorgente delle classi richieste;
- il sorgente di una classe aggiuntiva contenente un **main** che consenta di testare le specifiche richieste;
- la documentazione del progetto generata da **javadoc** (ricordo che questa documentazione ha un qualche significato solo se il codice è stato opportunamente commentato);
- Una breve relazione (al massimo una decina di pagine – escluso il testo del progetto) che descriva sinteticamente le principali scelte algoritmiche effettuate.
- Il progetto va consegnato tassativamente entro 5 settimane dall'assegnazione del progetto. La data di consegna verrà comunicata dal docente per email.