

Algoritmi e Strutture Dati

Game of (Approximated) Thrones (**got**)

Testo del problema

Melius fieri

Motto della casata Montron

Slides originali su: <https://judge.science.unitn.it/slides/asd18/prog2.pdf>

Le casate di Westeros sono ormai in guerra da anni: numerosi Lord proclamano di essere l'unico vero Re. Fra di essi, un Lord sconosciuto riesce a farsi strada fra tutti e a conquistare il Trono di Spade! Il suo nome è Albert di casa Montron, Primo del suo Nome, Re dei sandali, Padre dei problemi, Protettore degli algoritmi, Distruttore di medie.

Dopo la vittoriosa battaglia per il trono, il nuovo Re deve dividere il territorio fra le varie casate, premiando chi si è alleato con lui e punendo chi si è defilato. Purtroppo molte delle antiche mappe sono andate bruciate - questo succede quando si prende un drago come animale da compagnia invece di un gatto.

Grazie a grandi ricerche, il maestro Criswell Tarly è riuscito a ricostruire una mappa del territorio e ad individuare alcune zone strategiche, contraddistinte dalla presenza dei **castelli**. Dopo importanti discussioni con il primo cavaliere Martya Stark, si è decisa la dimensione delle aree da assegnare per ogni castello. Pur di riuscire ad assegnare più castelli possibili, rispettando le dimensioni scelte, potrà accadere che più castelli vengano assegnati ad una stessa suddivisione. Inoltre, il Re vuole evitare che due suddivisioni della stessa dimensione si trovino vicine. Tra loro non ci sarebbe alcuna rivalità e potrebbero allearsi contro di lui.

Ora non resta che suddividere il territorio fra le casate, rispettando le seguenti regole:

- Le suddivisioni devono essere valide e rispettare l'associazione con i castelli dati in input, secondo le definizioni sottostanti.
- Alcune parti del territorio possono rimanere non assegnate.
- Se non si riesce a assegnare una suddivisione a un castello viene raso al suolo.
⇒ È importante riuscire a fare in modo che **tanti più castelli possibili** facciano parte di una suddivisione.

Aiutate Re Albert, Criswell Tarly e Martya Stark a creare una giusta suddivisione dei territori!

Obiettivo

Il vostro compito è fornire una mappa con delle suddivisioni valide associate ai castelli dati in input, tale che **quanti più castelli possibili** facciano parte di una suddivisione valida.

Suddivisioni valide

Una **suddivisione valida** è definita nel seguente modo:

- Due celle confinano se hanno un lato in comune, ovvero le celle confinano orizzontalmente o verticalmente. Celle con un vertice in comune (in diagonale) non sono confinanti.
- Se due celle confinanti contengono lo stesso valore $k > 0$, allora appartengono alla stessa suddivisione.

- La dimensione di una suddivisione è data dal numero di celle che la compongono.
- Una suddivisione di dimensione k è **valida** se e solo se le sue celle contengono il valore k .

Suddivisioni confinanti: è impossibile che due suddivisioni valide contenenti valori k siano confinanti; infatti, la loro unione sarebbe vista come una suddivisione **non valida**, in quanto contenente il valore k e di dimensione $2k$. Le seguenti regole definiscono il concetto di associazione fra castelli e suddivisioni. L'output **rispetta la suddivisione dei castelli** se:

- Contiene solo suddivisioni valide.
- Per ogni suddivisione valida di dimensione k :
 - *almeno* una delle sue celle contiene un castello di valore k nella corrispondente cella dell'input;
 - nessuna delle sue celle contiene un castello di valore $k' \neq k$ nella corrispondente cella dell'input, non potete "sostituire" un castello con un altro;
- tutte le celle dell'output che non fanno parte di una suddivisione devono contenere il valore 0, anche se nella corrispondente cella dell'input c'era un castello.

Castelli senza terreno: se non si riesce a trovare una suddivisione valida per un castello viene raso al suolo, mettendo uno 0 nella cella corrispondente.

Esempi



Figura 1: Il continente di Westeros con i suoi castelli, ciascuno riporta la dimensione della suddivisione di territorio a cui deve essere associato il castello stesso.



Figura 2: Soluzione parziale: 9 castelli su 12 castelli sono associati con una suddivisione valida.



Figura 3: Soluzione ottima: tutti i castelli (12/12) sono associati con una suddivisione valida.

Input/Output

Input: un file con la mappa del territorio, con i rispettivi castelli e la dimensione della suddivisione di cui potrebbero far parte. Tutti i numeri sono separati da **tabulazioni**.

- La prima riga riporta 2 numeri, N ed M , rispettivamente il numero di righe e di colonne della mappa.
- Le successive N righe sono composte da M interi che descrivono la mappa:
 - Se una cella della mappa è marcata con un numero $k > 0$, questa rappresenta un **castello**, che deve essere posizionato in una **suddivisione di dimensione k**. Tutte le altre celle sono marcate con uno 0.

Output: un file con le vostre proposte di suddivisione della mappa, ogni soluzione è così composta:

- N righe, ciascuna contenente M colonne, ovvero la mappa delle suddivisioni;
- 1 riga contenente i caratteri ***.

Punteggio

- Ci sono 20 casi di test: ogni test assegna un punteggio (max 5 punti), che considera fino ai centesimi;
- Una soluzione proposta è valida se rispetta tutti le regole richieste. Soluzioni non valide fanno **zero** punti!
- Le risposte valide invece ottengono questo punteggio:

$$\max \left(0.0, \frac{N_v - N_1}{N_c - N_1} \right) \cdot 5$$

Dove N_v è il numero di castelli inseriti in suddivisioni valide, N_c è il numero di castelli dati in input, N_1 è il numero di castelli che richiedono dimensione 1.

Esempi (punteggio)

Con riferimento all'input dato in Figura 1, ecco due esempi di calcolo del punteggio:

1. Nell'esempio in Figura 2 è presentata una soluzione parziale: sono stati associati $N_v = 9$ castelli su $N_c = 12$. Tra questi escludiamo $N_1 = 1$ castelli corrispondenti al valore 1 in input, quindi il punteggio sarà $5 \frac{8}{11} = 3,64$ su un massimo di 5.
2. Nell'esempio in Figura 3 è presentata una soluzione ottima: sono stati associati $N_v = 12$ castelli su $N_c = 12$. Tra questi escludiamo $N_1 = 1$ castelli corrispondenti al valore 1 in input, quindi il punteggio sarà $5 \frac{11}{11} = 5$ su un massimo di 5.

Valutazione

Per valutazione del progetto:

- Conta il punteggio dell'**ultimo sorgente** inviato al sistema;
- Il progetto è superato con un punteggio non inferiore a 40 punti;
- C'è un limite di 40 sottoposizioni per gruppo;

Limiti e assunzioni

Limiti generali

- $0 < N, M \leq 200$
- $0 < k < NM$

Casi di test

- I casi di test hanno tutti almeno una soluzione ottima.
- In almeno 11 casi c'è un solo castello per suddivisione.

Dataset di esempio

Per gli input forniti nel dataset di esempio non è stata calcolata una soluzione ottima. Per questo motivo il dataset non contiene anche i relativi output, solitamente messi a disposizione.

Requisiti tecnici

Il main va sempre dichiarato come `int main()` o `int main(void)`. Questo esercizio deve essere svolto in C++, non è possibile usare il C.

```
... include delle librerie di sistema ...
#include "got.h"

int main() {
...
    return 0;
}
```

è importante che il main termini correttamente con un'istruzione `return 0`.

Istruzioni di compilazione

Di seguito riportiamo le istruzioni per testare i vostri programmi su vari sistemi. Si suppone che il sorgente con il vostro codice si chiami file `got.cpp`. I file `got.cpp`, `grader.cpp` e `got.h` devo stare nella stessa cartella.

Sistemi GNU/Linux

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o got got.cpp grader.cpp
```

Sistemi Mac OS X

Su sistemi Mac OS X usate il seguente comando di compilazione:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -o got got.cpp grader.cpp
```

Se ottene un errore del tipo: `use of undeclared identifier quick_exit`, sostituete in `grader.cpp` l'istruzione `quick_exit(EXIT_SUCCESS);` con `exit(EXIT_SUCCESS);`.

Sistemi Windows

Per il sistema Windows 10 potete installare il “Windows Subsystem for Linux”¹. Successivamente potete installare i tool necessari per usare Visual Studio Code² o Visual Studio 2017³ seguendo le relative guide riportate nelle note.

¹<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

²<https://code.visualstudio.com/docs/cpp/config-wsl>

³<https://devblogs.microsoft.com/cppblog/targeting-windows-subsystem-for-linux-from-visual-studio/>

Usando questo sistema fate attenzione a dove salvate i file e a quale nome gli date in quanto potreste avere delle difficoltà con percorsi che contengano spazi e caratteri speciali.

In alternativa, o per sistemi precedenti a Windows 10 potete installare *Cygwin*⁴, un ambiente completamente POSIX-compatibile per Windows. Anche in questo caso esistono guide per configurare i comuni editor disponibili su Windows di modo che utilizzino l'ambiente Cygwin, come per esempio Visual Studio⁵.

Una volta installato Cygwin è possibile simulare quanto avviane su arena compilando il proprio sorgente senza includere l'header `got.cpp` e il grader `grader.cpp`:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o got got.cpp
```

e lanciare il comando come:

```
timeout.exe 2 ./got
```

`timeout.exe` arresterà il programma dopo 2 secondi.

Esempi di input/output

File input.txt	File output.txt
9 5 0 0 0 0 0 0 2 0 0 3 0 0 0 0 0 0 0 7 0 0 0 6 1 0 0 2 0 0 0 9 0 0 0 0 0 5 0 6 0 0 5 0 6 4 0	0 0 0 3 3 0 2 7 7 3 0 2 0 7 0 6 0 7 7 0 6 6 1 7 7 2 6 6 9 9 2 0 6 9 0 0 9 9 9 9 0 9 0 0 9 *** 7 7 0 3 3 7 2 2 0 3 7 7 0 0 9 0 7 7 9 9 2 0 1 9 0 2 9 9 9 9 6 6 6 4 9 0 6 0 4 4 0 6 6 4 0 *** 7 7 7 7 3 6 2 7 3 3 6 2 7 9 9 6 6 7 9 9 6 6 1 9 9 2 2 6 6 9 5 6 6 9 9 5 5 6 4 4 5 5 6 4 4 ***
5 5 0 0 0 6 0 0 2 0 0 0 4 0 0 8 0 0 0 0 0 0 0 0 0 0 5	6 6 6 6 8 2 2 6 6 8 4 8 8 8 8 4 4 5 8 8 4 5 5 5 5 ***

⁴<https://www.cygwin.com/>

⁵<https://devblogs.microsoft.com/cppblog/using-mingw-and-cygwin-with-visual-cpp-and-open-folder/>