Politecnico di Torino

Algoritmi e Strutture dati
Relazione prova d'esame

Andrea Angelo Raineri - s280848

27/1/22

1 Modifiche apportate alla versione originale

- Aggiunti prototipi di funzione in testa al file, prima della funzione main()
- **riga 102**: corretto nome variabile utilizzato all'interno della funzione nei parametri

  `void free2d(int ***mat, int R, int C)`

  in

  `void free2d(int ***griglia, int R, int C)`
- **riga 149 e 153**: aggiunto controllo mancante per verifica validità dimensione regione quadrata (la dimensione deve rispettare i limiti della matrice stessa)

```
149     for (d = 1; d <= R && d <= C; d++) {
150     validDim = 1;
151     for (i = 0; i < d && validDim; i++) {
152         for (j = 0; j < d && validDim; j++) {
153             if ((pos/C + i) >= R || (pos%C + j) >= C || griglia[pos/C + i][pos%C + j] > 0)
154                 validDim = 0;
155         }
156     }
```

2 Strutture dati utilizzate

E' stata utilizzata principalmente un unica struttura dati al fine di memorizzare la struttura della griglia e al fine di mantenere traccia della copertura in fase di costruzione di una soluzione ottima. Si è scelto di utilizzare una **matrice di interi** di NR righe ed NC colonne (ricevuti come parametri durante la lettura del file `griglia.txt` in cui ogni cella assume valore:

- **0** se libera
- **1** se occupata da un ostacolo
- **2** se occupata da una regione

L'algoritmo per il calcolo di una copertura ottima fa uso della matrice per tenere traccia dell'area già coperta da regioni e salva le regioni che vengono inserite in due **vettori** che tengono traccia della posizione dell'angolo in alto a sinistra di una regione all'interno della matrice e della dimensione della regione. Il file `proposta.txt` si è supposto avente un formato simile, con l'eventuale possibilità che una regione avesse differenti dimensioni di altezza e larghezza.

La "posizione" di una regione, sia all'interno del file `proposta.txt` che durante la costruzione della soluzione ottima si intende come distanza dalla posizione in alto a sinistra della matrice in strategia row-major. Viene dunque convertita in indici specifici riga e colonna quando necessario.

3 Strategie algoritmiche

1. **Verifica validità copertura ricevuta da file**

   Per ogni regione, indicata nel file di input da posizione, larghezza e altezza, si verifica che le due dimensioni corrispondano (verifica di regione quadrata) e si salva all'interno della griglia la regione salvando il valore 2 in ogni cella coperta dalla regione. Dopo aver memorizzato tutte le regioni della proposta di copertura si verifica che tutte le celle della griglia abbiano valore non nullo. Il costo di complessità della verifica di completa copertura

delle regioni bianche è $O(NR * NC)$ nel caso di copertura non valida, $\Theta(NR * NC)$ nel caso di copertura valida.

2. **Ricerca copertura ottima**

   L'algoritmo utilizzato per la ricerca di una copertura ottima sfrutta un approccio divide et impera. L'obiettivo di ogni chiamata alla funzione `findCopertura(pos, n)` è quella di inserire a partire dalla posizione *pos* un numero n di regioni.

   Ad ogni passo ricorsivo:

   - se la cella è già occupata (valore 1 o 2) allora non si effettua nessuna operazione e si ricorre direttamente alla posizione successiva
   - se la cella ha valore 0 (cella bianca non coperta) si prova ad inserire una regione di una certa dimensione, si salva la modifica sulla griglia ponendo valore 2 nelle celle interessate e si ricorre alla posizione successiva con un numero *n-1* di regioni ancora da inserire

   La ricorsione termina quando *pos* raggiunge l'ultima posizione sulla griglia o quando sono state inserite tutte le regioni. A questo punto si sfrutta la stessa funzione di verifica implementata al punto precedente per verificare la validità della soluzione.

   Quando si deve inserire una regione in una posizione la dimensione (inizialmente 1) viene incrementata ogni volta che la discesa ricorsiva non ha portato ad una soluzione accettabile, fino ad una dimensione massima accettabile tale da non portare la regione a fuoriuscire dalla matrice stessa o a coprire delle celle già coperte/occupate.

   La funzione wrapper si occupa dell'incremento della variabile n, ovvero del numero di regioni utilizzate per coprire l'area bianca. Partendo da un numero di regioni uguale a 1 si incrementa il valore finché la funzione `findCopertura(0, n)` non trova una soluzione valida. In questo modo si è certi che la prima soluzione valida identificata è anche quella ottima in termini di regioni utilizzate.

   La presenza di backtrack nella scelta della dimensione di una regione ad ogni passo della ricorsione rende l'algoritmo molto costoso da un punto di vista computazionale. La complessità non è inoltre legata unicamente alle dimensioni della matrice ma dipende pesantemente anche dalla presenza di celle ostacolo e dalla loro disposizione sulla griglia.