Politecnico di Torino

Relazione d'Esame

27/01/2022

Andrea Grillo, S282802

Nella presente relazione verranno discussi gli approcci e le strategie adottate per la risoluzione dell'esame di *Algoritmi e Strutture Dati* del *27/01/2022*, con particolare riguardo alle strutture dati utilizzate, al formato dei file di input e agli algoritmi scelti ed implementati.

Strutture dati e formato del file

Il testo richiede di lavorare su una griglia, che è stata memorizzata in una matrice con allocazione dinamica. Essa viene allocata conseguentemente alla lettura della prima riga del file *griglia.txt*, che contiene le dimensioni della matrice. Il compito di deallocare la memoria a fine utilizzo viene dato al client, che alla fine dell'esecuzione del programma richiama la funzione *freeMat*.
Per quanto riguarda la definizione del file *proposta.txt*, si è cercato un formato che rendesse più semplice possibile la verifica delle condizioni date. Si è scelto di definire nella prima riga del file il numero di regioni e poi di inserire una regione per riga secondo il seguente formato:

i j dim1 dim2

dove *i* e *j* sono le coordinate del primo vertice della regione, e *dim1* e *dim2* sono le dimensioni della regione. Tale memorizzazione ha reso molto efficiente la verifica della prima condizione, infatti per verificare che le regioni siano quadrate è sufficiente controllare che *dim1* e *dim2* fossero uguali.
Si è ritenuto opportuno, per eseguire la funzione di verifica, l'allocazione di un'altra matrice e la copia dei dati dalla matrice originale. Tale scelta è sembrata utile allo scopo di migliorare la complessità di caso medio. È infatti necessario che la matrice non venga modificata alla fine dell'esecuzione dell'algoritmo di verifica, però ciò, nel caso in cui non si fosse creata la copia, avrebbe comportato la necessità di effettuare una nuova visita alla fine della verifica per ricostituire la matrice originale. Con tale soluzione invece, pur utilizzando uno spazio maggiore, una volta terminato l'algoritmo, che la soluzione sia valida o meno, è sufficiente liberare la matrice copiata, e la principale non viene modificata in nessun modo. Non è quindi necessario andare ad eseguire ripristini.
Per la soluzione dell'ultimo quesito, che richiedeva la stesura di una funzione ricorsiva di esplorazione dello spazio delle soluzioni, è stato necessario utilizzare, oltre alla matrice utilizzata per la verifica di validità delle soluzioni, anche dei vettori per il salvataggio della soluzione trovata. I primi due vettori (*sol_i*, *sol_j*) servono a memorizzare le coordinate del primo vertice della regione e l'ultimo (*sol_dim*) la dimensione di tale regione. È sufficiente salvare una sola dimensione, in quanto vengono generate unicamente regioni quadrate. Tali

vettori vengono allocati dinamicamente nella funzione *solve*, che è una funzione *wrapper* della vera e propria funzione ricorsiva *solveR*.

Scelte algoritmiche

Primo quesito

Il primo quesito richiede il caricamento in memoria di una matrice. La matrice viene allocata dinamicamente. Durante l'allocazione di ogni singola riga viene letto il file e vengono popolati i singoli vettori. Infine viene ritornato il puntatore alla matrice. Il main passa inoltre alla funzione due puntatori a intero, *nr* e *nc* che vengono utilizzati per salvare le dimensioni della matrice letta e darne visibilità al main.

Secondo quesito

Per quanto concerne la risoluzione del secondo quesito, l'approccio più semplice da implementare è stato quello di andare ad effettuare, per ogni regione letta dal file, un incremento unitario su tutti i campi della matrice appartenenti alla regione. Ciò ha permesso, con un'ulteriore visita della matrice, di verificare la condizione di copertura totale della matrice con una semplice verifica del valore della matrice in ogni punto. I seguenti casi sono possibili:

- *mat[i][j] = 0* matrice non coperta in quel punto
- *mat[i][j] = 1* matrice coperta, soluzione valida
- *mat[i][j] > 1* matrice coperta da più regioni, o una regione e una zona nera

È sufficiente che nella matrice ci sia un valore diverso da 1 affinché la soluzione non sia valida. Per questo motivo, appena si trova un valore errato, si interrompono le iterazioni di visita della matrice. Dato l'utilizzo di un'ulteriore matrice per effettuare la verifica, come spiegato in precedenza, non è necessario effettuare alcuna altra operazione ed è possibile passare direttamente alla liberazione della memoria occupata e ritorno del valore.

Terzo quesito

Il terzo quesito riguarda la soluzione di un problema di ricerca ed esplorazione dello spazio delle soluzioni tramite un algoritmo ricorsivo. È necessario generare l'insieme a cardinalità minima di regioni che riempiono la matrice. A tale scopo si è scelto di generare insiemi a cardinalità via via crescente di regioni, fin quando non si trova una soluzione valida. Con questa strategia si è sicuri che la prima soluzione trovata è quella a cardinalità minima. La variabile *k*, che indica il numero di regioni da generare, è inizialmente nulla, cosa che permette al caso limite in cui la matrice sia già totalmente piena di zone nere di essere gestito correttamente.
All'interno della funzione ricorsiva *solveR* viene inizialmente gestito il caso terminale in cui *pos* >= *k*. In tal caso, viene effettuata una chiamata alla funzione *checkSol*, che ha come unico compito quello di controllare che tutta la matrice sia stata riempita, dato che tutte le altre condizioni vengono verificate durante l'aggiunta di ogni singola regione, utilizzando il *pruning*. Se la soluzione è valida si setta a 1 la variabile *flag_found*, che blocca l'albero di ricorsione.
Nel caso in cui invece *k* sia minore di *pos*, l'algoritmo deve aggiungere una regione valida alla soluzione. Tale regione deve essere di ogni dimensione accettabile e posizionata in ogni

posizione accettabile. La validità del posizionamento della regione è garantita dalle condizioni di uscita dai cicli for, che garantiscono che per ogni dimensione vengano generate solo le coordinate di posizionamento accettabili per non uscire dalla matrice, il che costituisce un primo livello di *pruning*. Il secondo livello viene effettuato dalla funzione *checkRegione*, che controlla, per ogni combinazione di posizione e dimensione, che tali dati siano compatibili con la matrice, in caso affermativo la regione viene aggiunta rispettivamente alla matrice e ai vettori soluzione *sol_i*, *sol_j* e *sol_dim*.

Elenco delle correzioni

Di seguito si espongono le correzioni apportate alla versione consegnata durante la prova, enumerate secondo il numero di riga della versione definitiva in allegato:

- *77:* aggiunta tipo void funzione *freeMat*
- *107:* modifica *,* con *;* in condizioni ciclo *for*
- *120:* aggiunta carattere _ al nome della matrice
- *127,128:* aggiunta condizione su variabile *ret* per migliorare caso medio
- *149:* aggiunta inizializzazione a *0 flag_found*
- *149:* rimossa dichiarazione di variabile intera *j*, non necessaria
- *161,162:* il numero di sottoregioni è *k-1*, non *k*, come già ipotizzato durante la prova
- *193,196,207:* cambiato il nome delle variabili *s_i, s_j* con *i, j*
- *261:* condizione ciclo *for*
    - versione errata: *for(j = s_j < s_j + dim; j++)*
    - versione corretta: *for(j = s_j; j < s_j + dim; j++)*