Algoritmi e Programmazione
Esame del 09/02/2022 – Prova da 18 punti

Simone Tumminelli
274608

1. Strategie risolutive e strutture dati adottate

-Strutture Dati

Le strutture dati su cui opera il programma sono due: vettori e matrici, entrambe allocate dinamicamente e di tipo intero. La griglia di gioco è rappresentata dalla matrice "griglia" di dimensioni "r" e "c"; le mosse di gioco sono rappresentate dal vettore "proposta", i cui "n" elementi sono numeri da 0 a 3 (ogni numero rappresenta una direzione, rispettivamente: 0=su, 1=giù, 2=destra, 3=sinistra).

-Strategie Risolutive

Parte I: Acquisizione)

L'acquisizione da file delle strutture dati avviene tramite le funzioni "leggiMatrice" e "leggiProposta" che prendono in input le rispettive dimensioni (righe e colonne per la matrice, numero n per il vettore di mosse) come puntatori, in modo che il loro valore possa essere modificato dall'interno delle funzioni una volta acquisito dai file e usato dalla funzione chiamante. Dopo aver acquisito il valore delle dimensioni vengono allocate dinamicamente le quantità di memoria necessarie, si itera sui file leggendo i dati e infine viene ritornato l'oggetto allocato.

Parte II: Verifica)

Funzionamento di "checkSol": la funzione ritorna il numero di cambi di direzione del percorso della proposta del vettore di mosse "sol" nel caso in cui la proposta sia valida, altrimenti ritorna -1. Innanzitutto, tramite la funzione "contaBianche", viene verificato che il numero di mosse della soluzione sia compatibile col numero di caselle percorribili nella griglia (in particolare, per una griglia con n caselle bianche, se quest'ultime sono tutte effettivamente percorribili con un percorso, allora il numero di mosse necessarie sarà n-1, perché si prende sempre in considerazione la posizione di partenza, cioè l'estremo in alto a sinistra della griglia). La verifica della soluzione si basa sulla percorrenza della copia della griglia "m": per ogni mossa contenuta nel vettore "sol", scandito tramite un ciclo for, si verifica la correttezza della mossa in base alla posizione corrente della matrice (scandita tramite gli indici "k" e "j", inizializzati a 0 (la posizione di partenza è sempre [0][0]) opportunamente incrementati o decrementati), e, in caso positivo, viene segnata la casella della matrice come percorsa (1) e quindi non più percorribile; inoltre, ad ogni passo, viene confrontata la mossa precedente con quella corrente, per contare il numero totale di cambi di direzione (il caso iniziale, per cui la mossa è "INIZIO" (-1), è opportunamente

trattato dalla funzione di confronto “cambioDir”). Alla fine del ciclo, se tutte le caselle inizialmente bianche sono state attraversate, allora la proposta contenuta in “sol” è valida.

Parte III: Ottimizzazione)
Funzionamento di “solve”: “solve” è la funzione wrapper della funzione “solveR”, riceve in input la matrice “m” e le sue dimensioni “r” e “c” e predispone i dati necessari al funzionamento della funzione ricorsiva: “val” è il vettore delle 4 mosse possibili (su, giu, destra e sinistra), “n” è la dimensione di ogni soluzione (calcolato come le caselle percorribili della matrice -1) “sol” e “bestSol” sono i vettori delle soluzioni e infine “mark” è il vettore dei marcaggi multipli inizializzato con tutti gli elementi uguali a “n”.
Funzionamento di “solveR”: l’algoritmo si basa sulle disposizioni ripetute, l’obbiettivo è quello di calcolare le sequenze ordinate in cui k elementi (k=4, dimensione di “val”) si dispongono in insiemi di “n” elementi (n può essere quindi maggiore o uguale a k); tra queste disposizioni si verificano quelle idonee, tra cui se sceglie quella con numero minimo di cambi di direzione. I dati usati sono: la posizione “pos”, il vettore di mosse “val” di dimensione “k” (4), la griglia “m” con le sue dimensioni “r” e “c”, i puntatori ai contatori dei cambi di direzione delle soluzioni corrente e ottima e infine i vettori di dimensione “n” “sol”, “bestSol” e “mark” (i cui elementi valgono inizialmente tutti “n”, cioè il numero massimo di volte che un elemento di “val” può essere scelto).
Per ogni elemento di “val”, se è possibile che sia preso (mark[i]>0), si assegna come elemento in posizione corrente della soluzione, si aggiorna mark e si ricorre sulla posizione successiva; dopo la ricorsione si esegue backtrack su mark. Quando la ricorsione termina le soluzioni corrente e ottima vengono confrontate, e in caso di numero di cambi di direzione minore, si aggiorna la soluzione ottima. La funzione solveR fa anche uso di pruning: e se i cambi della soluzione corrente sono maggiori dei cambi di quella ottima, oppure se la soluzione corrente non è valida allora la ricorsione termina.

2. Differenze tra compito sviluppato durante l’appello e codice allegato alla relazione

Nelle parti di acquisizione e verifica non è presente alcuna differenza algoritmica tra codice allegato e compito sviluppato su carta.
Nella parte di ottimizzazione sono state aggiunte le parti mancanti.

Simone Tumminelli
274608