Relazione Esame: Problema da 18 punti

Strutture dati

Per il labirinto ho implementato un ADT Graph non orientato e pesato usando la rappresentazione a matrice di adiacenze madj per i cunicoli e un ADT symboltable per i vertici. Queste scelte sono dettate dalla necessita' di gestire varie informazioni riguardo i cunicoli (trappole) e vertici (profondita', quantita' di oro e valore tesoro) in maniera immediata. Infatti ho strutturato la matrice in modo che contenga i valori: -1 se non è presente arco, 0 se è presente e non contiene trappole, 1 se invece le contiene.
La tabella di simboli è realizzata mediante vettori non ordinati, uno per ogni informazione del vertice, cosi' da poter definire funzioni by_index che ritornano valori di riferimento con costo O(1). Per il cammino ho definito un ADT Path struttura che contiene vari parametri come il numero di passi, il valore del tesoro massimo, la profondita' dell'ultima stanza e il valore complessivo e un vettore di interi che contiene gli identificatori dei vertici toccati.

Acquizizione grafo

Il grafo e' inizializzato dalla funzione GRAPH_init che ritorna un handle Graph, questa alloca una matrice di interi inizializzata al valore -1 mentre si appoggia alla funzione ST_init per allocare la tabella di simboli.
Per gestire l'acquisizione da file nel formato richiesto ho usato due cicli di chiamate a funzioni di libreria necessariamente modificate: ST_insert inserisce V dati relativi alle sale (profondita', quantita' di oro e valore tesoro) nella tabella di simboli, GRAPH_insert_E inserisce i cunicoli nella matrice delle adiacenze fino a che non finisce il file.

Problema verifica e acquisizione percorso

Ho gestito l'acquisizione da file del percorso attraverso la funzione GRAPH_path_load che attraverso un ciclo while per ogni stringa letta aggiunge il suo identificatore intero al vettore stanze del cammino Path grazie ad una ricerca in tabella di simboli O(n). Dato che non conosco il numero di elementi massimi (M non e' presente nel prototipo) se la dimensione del cammino eccede la dimensione massima uso realloc.
Per verificare la validita' del percorso secondo i vincoli (Mosse e Punti Ferita) la funzione GRAPH_path_check scandisce il vettore stanze e ad ogni passo provvede ad aggiornare il valore del tesoro (se è maggiore del tesoro massimo precedente) e quello dell' oro (con vettore visited), inoltre per ogni mossa (attraversamento cunicolo) controlla la presenza di trappola e in caso affertmativo decrementa PF.
Successivamente assegna a TOT il valore totale del cammino attraverso uno switch della profondita' dell'ultima stanza visitata. La funzione ritorna fallimento nei casi in cui M<0 o PF < 0.

Problema ottimizzazione

La funzione wrapper GRAPH_path_best procede ad allocare i vettori camm e camm_max che rappresentano rispettivamente le sequenze di identificatori di stanze del cammino corrente e di quello ottimo. Le variabili len_max e TOT sono passate alla funzione ricorsiva per riferimento in quanto devono essere modificate da quest'ultima. A seguito della chiamata ricorsiva il vettore camm_max viene copiato nel vettore stanze all'interno di path, cosi' come il valore totale e la lunghezza.
La funzione ricorsiva GRAPH_path_best ritorna void perche' per semplicita' ho preferito lavorare con i vettori di interi passati come parametro.
Ad ogni passo ricorsivo la funzione aggiunge un nuovo vertice al vettore camm mentre vengono aggiornati i valori relativi alla quantita' d'oro, controllando se il vertice non sia gia' stato attraversato, e al tesoro, nel caso in cui sia maggiore del tesoro massimo precedente.
Ad ogni mossa M e PF, passati come parametri by-value, vengono decrementati, PF solo nel caso rappresentti un cunicolo con trappola (1 nella matrice delle adiacenze).
La condizione di terminazione, cioe' l'interruzione del percorso, avviene se vengono terminate le mosse o i punti ferita. Nel codice questa condizione e' rappresentata dall'espressione booleana (M<0 || PF<0), quando il programma entra nell'if considera il cammino fino al vertice della ricorsione precedente e ne calcola il punteggio con lo stesso switch con cui era stato calcolato nella funzione di verifica. Se il punteggio e' maggiore del massimo precedente si aggiornano camm_max, TOT e len_max.

Modifiche a codice

- ✓ *[r.155 Graph.c] GRAPH_path_check : uso vettore visited per non far prendere piu' volte oro di una stanza*
- ✓ *[r.159 Graph.c] GRAPH_path_check : metto controllo per evitare sforare limite vettore con ultima iterazione*
- ✓ *[r.247-250] GRAPH_path_best_R : aggiornamento soluzione massima*
- ✓ *[r.258] GRAPH_path_best_R : chiamata a funzione visited per vedere se vertice gia' stato percorso*
- ✓ *[r.268] GRAPH_path_best_R : chiamata ricorsiva*

Commenti

- ✓ *GRAPH_path_check : nell'interpretazione del testo che ho dato questa funzione ha il compito di controllare il rispetto dei vincoli M e PF, senza verificare la correttezza dei dati immessi come ad esempio l'effettiva presenza dei vertici nel grafo, cosa che avrei potuto fare con una ricerca in tabella di simboli con flag ritorno.*
- ✓ *Interruzione anomala respondus : quando mancavano circa 20 minuti la piattaforma di esame si e' chiusa forzatamente e non ho potuto terminare il mio compito. Per fortuna ero in leggero anticipo e al momento della chiusura avevo gia' impostato i blocchi principali della funzione ricorsiva, mi restava solo da scrivere la*

chiamata ricorsiva all'interno del ciclo (punto 3 delle modifiche: solamente riportare i parametri decrementati) e l'aggiornamento della soluzione all'interno dell if della condizione di terminazione (punto 3 delle modifiche : copia 2 vettori e calcolo ricchezza).

- ✓ *Non ho riportato tra le modifiche al codice sopra le chiamate della funzione ricorsiva nella funzione wrapper perche' si tratta banalmente di trascrivere i parametri della funzione.*