03MNO ALGORITMI E PROGRAMMAZIONE CORSO DI LAUREA IN INGEGNERIA INFORMATICA A.A. 2018/19

Commento al Laboratorio n. 8

Esercizio n.1: Sequenza di attività

Come da specifiche, si crea una struct att con 2 campi interi s e f per memorizzare il tempo di inizio e quello di fine di un'attività e un vettore di attività a il cui numero di elementi n è letto sulla prima riga del file.

Il problema è di ottimizzazione e richiede di generare tutti i sottoinsiemi degli n intervalli con la condizione che siano compatibili e tra questi ritornare quello (o uno di quelli) la somma delle cui durate sia massima. Il modello per generare tutti i sottoinsiemi è quello del powerset, implementato con le disposizioni ripetute. Nella condizione di terminazione si confronta la durata della soluzione appena calcolata con quella stimata massima e si aggiornano durata massima e soluzione ottima se è il caso. Se un intervallo non viene preso si ricorre sull'intervallo successivo. Si subordina la selezione dell'intervallo e la discesa ricorsiva sul prossimo alla verifica di accettabilità, cioè che esso non intersechi nessuno degli intervalli già inclusi nella soluzione. Trattasi quindi di una forma di pruning in cui i vincoli condizionano la discesa ricorsiva (in questo caso quella in cui l'intervallo è preso).

Esercizio n.2: Tessere e scacchiere

Strutture dati: per le tessere si introduce una struct con 2 campi: col vettore di 2 caratteri per registrare i colori, che si suppongono codificati con una lettera singola, del tubo orizzontale e del tubo verticale e val, vettore di 2 interi con significato analogo. La singola mossa è memorizzata come una struct mossa con campi la tessera e un intero che indica se è ruotata o meno. La collezione di tessere è memorizzata in un vettore vTiles, quella di mosse è una scacchiera board di R righe e C colonne. Per impedire che la stessa tessera sia usata più volte si introduce un vettore mark.

Modello: si osservi la somiglianza con il Sudoku: anche in questo caso si tratta di leggere una configurazione iniziale della scacchiera e poi riempire le celle vuote con le tessere disponibili. Si osservi però che nel Sudoku le cifre erano ripetute, in questo caso le tessere non lo sono. Il modello è quello delle disposizioni semplici. Le mosse possibili sono:

- se la cella contiene già una tessera si ricorre sulla cella successiva
- se la cella è vuota e la tessera corrente non è ancora stata utilizzata
 - o la si marca come utilizzata, la si piazza nella cella corrente senza ruotarla e si ricorre sulla cella successiva
 - o la si marca come utilizzata, la si piazza nella cella corrente ruotandola e si ricorre sulla cella successiva
 - o si annulla la decisione precedente (backtrack) non piazzando la tessera nella cella corrente e marcando la tessera come non utilizzata.

Nella soluzione proposta non si introduce pruning, non considerando infatti come tale il controllo su cella già piena e/o tessera già utilizzata.

Il controllo di ottimalità viene effettuato nella condizione di terminazione. La funzione score itera sulle righe e sulle colonne, calcolando per ciascuna un valore parziale di punteggio e poi ritornando il punteggio globale. Il controllo parte identificando il colore della prima cella della riga o della colonna, scandendo le rimanenti e sommando il loro contributo solo se il colore è lo stesso, altrimenti il punteggio parziale è 0. Nell'identificare la tessera corrente si tiene conto della sua eventuale rotazione.

Esercizio n.3: Gioco di ruolo (multifile)



03MNO ALGORITMI E PROGRAMMAZIONECORSO DI LAUREA IN INGEGNERIA INFORMATICA A.A. 2018/19

Nel file inv.h sono state inserite le definizioni delle struct relative agli oggetti dell'inventario stat_t e inv_t e alla loro collezione tabInv_t e gli header delle funzioni che vi operano. Nel file inv.c compaiono le implementazioni delle funzioni i cui header sono in inv.h: leggiStat, stampaStat, leggiInv e stampaInv, nonché quelle relative alla loro collezione leggiTabInventario e stampaTabInv.

Nel file pg.h sono state inserite le definizioni delle struct relative ai personaggi pg_t, alla loro collezione tabPg_t e all'equipaggiamento di ciascun personaggio tabEquip_t e gli header delle funzioni che vi operano.

Nel file pg.c compaiono le implementazioni delle funzioni i cui header sono in pg.h: leggiPg, newNodoPg, inserisciInListaPg, leggiTabPg, stampaPg, stampaTabPg, ricercaCodice, ricercaCodiceRef, aggiungi, freeEquip, freePg, modificaTabEquip, aggiornaPgStatEquip, aggiornaPgEquip, elimina.

Per l'implementazione delle funzioni si faccia riferimento alle soluzioni dell'es. 3 del Lab. 7. Il main dichiara le strutture dati, ne legge i contenuti da file e gestisce il menu.