Commento al Laboratorio n. 7

Esercizio n. 1: **Collane e pietre preziose**

Trattasi di problema di ottimizzazione. Una volta letti i dati (numero di zaffiri, rubini, topazi e smeraldi) è calcolabile la lunghezza massima della collana `maxlun`. Il `main`, mediante un ciclo, esplora i problemi di lunghezza `k` crescente tra 1 e `maxlun` e registra in `bestlun` il massimo valore di `k` per cui si è trovata una soluzione accettabile. Questo soddisfa la richiesta di trovare una soluzione ottima, quindi collana a lunghezza massima. Il `main` opera iterativamente su `numtestset` problemi: letta da un file di ingresso la quaterna che rappresenta il problema corrente, calcola la lunghezza massima possibile della collana e poi per tutte le lunghezze `k` tra 1 e la massima risolve il problema. Si ipotizza per il file in ingresso un formato con la prima riga che contiene `numtestset` (numero di problemi, cioè di quaterne), seguita dalle quaterne che descrivono ciascun problema. Vengono proposti diversi file di prova di difficoltà variabile.

Il modello del Calcolo Combinatorio è quello delle disposizioni ripetute di `N` oggetti presi a `k` a `k`. Si presentano 3 soluzioni: da un file di ingresso

1. versione 0: la verifica dell'accettabilità di una soluzione di lunghezza `k` è fatta nella condizione di terminazione. La funzione `check`:
   - calcola in `usGemme` il numero di occorrenze di ciascuna gemma nella soluzione corrente. Se tale numero eccede la disponibilità registrata nel vettore `numGemme`, la soluzione è scartata
   - verifica le regole di composizione: scorrendo la soluzione `sol`, in base alla gemma scelta in posizione `i`-1 si verifica che quella in posizione `i` sia conforme alla regola, altrimenti si scarta la soluzione.

   Non essendo prevista alcuna forma di pruning, questa soluzione è accettabile solo per lunghezze massime di collane molto piccole
2. versione 1: si introduce una prima forma di pruning: nella condizione di terminazione si verificano solo le regole di composizione, mentre la discesa ricorsiva è subordinata alla verifica della disponibilità di gemme. Sperimentalmente si osserva un discreto miglioramento nella capacità di trattare in tempi ragionevoli lunghezze massime maggiori
3. versione 2: la condizione di terminazione non prevede verifica di accettabilità, in quanto anche la verifica di regole di composizione è usata per condizionare la ricerca ricorsiva. Sperimentalmente si verifica la capacità di trattare in tempi ragionevoli lunghezze massime notevoli.

La versione 2 viene modificata a livello di `main` nel ciclo che itera sulle catene:

- versione 3: il ciclo avviene per lunghezze decrescenti delle catene, nell'ipotesi di interromperlo non appena giunti ad una soluzione all'iterazione con lunghezza `k`, in quanto le iterazioni successive possono portare solo a lunghezze minori
- versione 4: si seleziona `k` in maniera dicotomica (a metà della catena). Se si trova una soluzione di lunghezza `k` si procede per lunghezze da `k`+1 a `N`, altrimenti per lunghezze da 1 a `k`-1.

Esercizio n. 2: Collane e pietre preziose (versione 2)

Trattasi di problema di ottimizzazione dove si chiede di massimizzare il valore della collana nel rispetto delle regole di composizione. Si segue la strategia dell'esercizio precedente con il `main` che opera iterativamente su `numtestset` problemi, acquisendo per ciascuno da un file di ingresso i dati sul numero di gemme, sul loro valore e sul numero massimo di ripetizioni

consecutive. Si ipotizza per il file in ingresso un formato con la prima riga che contiene `numtestset` (numero di problemi), seguita dalle n-uple di 9 dati che descrivono ciascun problema. Una volta letto il numero di zaffiri, rubini, topazi e smeraldi è calcolabile la lunghezza massima della collana `maxlun`.

La funzione wrapper `solve` alloca le strutture dati per la funzione ricorsiva di risoluzione: i vettori `sol` e `bestSol` di significato evidente, il vettore `usGemme` per tener traccia del numero di gemme di ogni tipo usate nella soluzione corrente, il vettore `ripGemme` per tener conto del numero di ripetizioni consecutive di una gemma nella soluzione corrente, gli interi passati per riferimento `bestval` e `bestlun` per tener traccia del valore e della lunghezza migliore stimati, l'intero `prec` per ricordare la gemma decisa al passo precedente di ricorsione.

Il modello del Calcolo Combinatorio è anche in questo esercizio quello delle disposizioni ripetute di `N` oggetti presi a `k` a `k`. La disponibilità di gemme, i valori consecutivi ripetuti e le regole di composizione sono utilizzate per condizionare la discesa ricorsiva. Il vincolo su zaffiri e smeraldi è invece verificato nella condizione di terminazione per non precludere l'esplorazione di tutto lo spazio utile. Il vettore `ripGemme` serve per registrare per ogni gemma il numero di occorrenze consecutive. Esso viene assegnato in fase di decisione su di una gemma e ripristinato nella configurazione precedente in fase di backtrack.

Esercizio n. 3: Gioco di ruolo
Strutture dati: si definiscono le seguenti `struct`:

- per le statistiche una `struct stat_t` con i 6 campi interi indicati dalle specifiche
- per gli oggetti che formano l'equipaggiamento una `struct inv_t` avente come campi `nome` e `tipo` (stringhe allocate dinamicamente) e le statistiche
- per l'equipaggiamento una `struct tabEquip_t` avente un campo intero `inUso` e un vettore `vettEquip` di puntatori a oggetti di tipo `inv_t`
- per il personaggio una `struct pg_t` avente come campi `codice`, `nome` e `classe` (stringhe allocate dinamicamente), le statistiche di base e quelle date dall'equipaggiamento e un puntatore `equip` a una `struct tabEquip_t`.

Gli item delle collezioni di dati sono i personaggi di tipo `pg_t` e gli oggetti di tipo `inv_t`:

- per i personaggi la collezione è una lista realizzata come una `struct` wrapper di tipo `tabPg_t` contenente il numero corrente di personaggi e i puntatori a testa e coda della lista. Il nodo della lista contiene un personaggio di tipo `pg_t` e un puntatore di tipo `linkPg` al nodo successivo
- per gli oggetti la collezione è una `struct` wrapper di tipo `tabInv_t` contenente il numero corrente di oggetti, il vettore degli oggetti di tipo `inv_t` e il numero di oggetti `n Inv`. Il campo `maxInv` riportato nella figura è un refuso.

Il menu nel `main` è basato su interi, nel quale è sufficiente un vettore di stringhe da visualizzare, chiedendo all'utente di specificare il numero corrispondente all'opzione scelta.

La gestione dei personaggi (lettura da file, inserimento in coda in lista, aggiunta, cancellazione, ricerca per codice con ritorno del personaggio, aggiornamento delle statistiche, etc) non presenta alcuna difficoltà concettuale, trattandosi di operazioni standard su liste.

La gestione degli oggetti dell'inventario (lettura da file, stampa) non presenta alcuna difficoltà concettuale.

L'aggiunta/rimozione un oggetto dall'equipaggiamento di un personaggio comporta la ricerca per codice dello stesso, di cui si ritorna il puntatore, nonché la modifica della `struct` di tipo `tabEquip_t` cui esso punta per rimuovere o aggiungere l'oggetto.