Scritto di Programmazione del 18 luglio 2014

Dal file "input", il main legge nell'array C una sequenza di sottosequenze di interi. Ogni sottosequenza termina con -1, mentre l'intera sequenza di sottosequenze termina con -2.

Esempio 1: una possibile sequenza di sottosequenze è la seguente: 2 3 -1 0 2 3 -1 2 3 -1 0 2 -1 -1 2 3 0 -1 -2. Le sottosequenze contenute nella sequenza sono 6. Tra di esse c'è anche la sottosequenza vuota (che è rappresentata solamente da -1).

Programmazione iterativa: dato l'array C definito dal main, si tratta di scrivere una funzione IC iterativa che restituisce col return un array X di interi (allocato dinamicamente da IC) ed un intero lim che soddisfano le seguenti pre e post-condizioni:

PRE_IC=(C è costituita da n>=0 sottosequenze di interi, ciascuna terminante con -1 e la fine delle sotto-sequenze è segnata da -2)

int * IC(int*C, int & lim)

POST_IC=(alla fine di IC, il valore di lim è il numero di sottosequenze presenti in C, IC restituisce col return un array X allocato dinamicamente in IC tale che per ogni i in [0, lim-1] contiene in X[2*i] l'indice in cui inizia in C la sottosequenza di indice i e in X[2*i+1] contiene la lunghezza della sottosequenza di indice i)

Esempio 2: Considerando la sequenza di sottosequenze dell'Esempio 1, alla fine di IC, lim deve avere valore 6 e X deve contenere le seguenti 6 coppie di interi, [0,2,3,3,7,2,10,2,13,0,14,3]. Prendiamo la coppia (10,2) in X, essa è la quarta coppia in X, cioè corrisponde alla sottosequenza di indice 3 in C. I due valori indicano che in C, questa sottosequenza comincia nella posizione 10 ed ha lunghezza 2 (escludendo il -1). La coppia (13,0) indica che la sottosequenza vuota appare in C alla posizione 13 ed ha lunghezza 0 visto che il -1 non conta.

Ulteriore richiesta su IC: visto che non si sa a priori quanto grande diventerà lim, si richiede che IC inizialmente allochi un array X di 20 posizioni, ma che sia in grado, qualora lim diventi maggiore di 10, di "allungare" X di ulteriori 20 posizioni e, se lim supera anche 20, allunghi X di altre 20 posizioni e così via. Non ci devono essere limiti alla possibilità di X di crescere.

Correttezza: si specifichi l'invariante del ciclo principale della funzione IC. Si delinei la dimostrazione di correttezza del ciclo.

Consiglio: conviene introdurre funzioni (iterative) ausiliarie. Per esse vanno specificate pre e post-condizioni.

Programmazione ricorsiva: Usando C e X si vuole costruire un array Z di lim interi tale che per ogni i in [0,lim-1], Z[i] contenga il numero di sottosequenze presenti in C che sono contenute nella sottosequenza di indice i. Una sottosequenza non va mai confrontata con se stessa, ma solo con le altre.

Esempio 3: Definiamo con chiarezza cosa significa in questo esercizio che una sottosequenza ne contiene un'altra. Consideriamo questa sottosequenza S=[2,1,4] essa contiene la sottosequenza vuota [], la sottosequenza [2], e anche la [2,1], e anche la [2,1,4]. Insomma S contiene tutti i suoi prefissi, compresa

una sottosequenza identica ad S. Si osservi che per determinare se una sottosequenza ne contiene un'altra, conviene ignorare il -1 finale.

Esempio 4: usiamo C e lim=6 dell'Esempio 1. L'array Z richiesto deve avere i seguenti lim=6 elementi: Z=[2,2,2,1,0,3]. Cerchiamo di capire questi 6 numeri. Innanzitutto vediamo che la sottosequenza vuota, di indice 4, non contiene nessuna sottosequenza (potrebbe contenere solo un'altra sottosequenza vuota, ma è l'unica presente). Poi Z[0]=2 indica che la sottosequenza di indice 0 in C contiene 2 altre sottosequenze di C: si tratta della sottosequenza di indice 2 ([2, 3], che è identica a quella di indice 0) e di quella vuota. Si osservi che anche Z[2]=2, il che è naturale visto che le sottosequenze di indice 0 e 2 sono identiche. Questo significa che la sottosequenza di indice 0 contiene quella di indice 2, ma anche quella di indice 2 contiene quella di indice 0. Questo fatto va notato perché ha un'influenza sulla funzione M che deve calcolare Z. Invece Z[5]=3 significa che la sottosequenza [2,3,0] contiene 3 altre sottosequenze. Una di esse è certamente quella vuota. Le altre 2 sono quella di indice 0 ([2,3]) e quella di indice 2 identica alla precedente.

Si chiede di realizzare una funzione M ricorsiva che soddisfi alle seguenti specifiche:

PRE_M=(C lista di sottosequenze, X l'array prodotto da IC con input C, lim= n. di coppie in X, lim>=0, j in [0,lim-1], Z di lim elementi con valori >=0, con vZ indichiamo il valore iniziale di Z)

void M(int*C, int*X, int lim, int j, int *Z)

POST_M=(per ogni n in [j..lim-1], Z[n]=vZ[n]+q dove q è il numero di sottosequenze in [j..lim-1] (distinte dalla n) che sono contenute nella sottosequenza n)

Si osservi che, come succede spesso per le funzioni ricorsive, POST_M può sembrare strana, ma essa indica per una qualsiasi invocazione di M, il contributo che quella invocazione porterà a Z. Quindi per la prima invocazione ricorsiva, j sarà 0 e quindi POST_M dice che alla fine Z è quello che ci si aspetta (se Z è inizialmente tutta a 0, cosa che il main fa). Ma per un j>0, la POST_M indica che si devono considerare solo le sottosequenze da j a lim-1 e che per ciascuna coppia di queste sottosequenze , se una delle 2 contiene l'altra, allora si deve incrementare il corrispondente elemento di Z. Quando si confrontano 2 sottosequenze si deve considerare anche il caso in cui esse siano identiche (pur essendo sottosequenze distinte, come le sottosequenze di indice 0 e 2 dell'Esempio 4).

Correttezza: delineare la dimostrazione induttiva della correttezza di M.

Consigli: 0) Sebbene non sia usato nell'Esempio 4, X deve essere usato da M per costruire Z.

1) come sempre, conviene introdurre funzioni (ricorsive) ausiliarie. Per esse vanno specificate pre e post-condizioni.

2) se si devono confrontare 2 sottosequenze conviene considerare la loro lunghezza e fare ricorsione sulla lunghezza minore (o uguale).

Integrazione: al posto della correttezza scrivere una funzione con lo stesso compito di M (costruire Z), ma iterativa anziché ricorsiva.