Scritto di Programmazione del 15/9/2015

Dato l'array A che contiene lim1*lim2*lim3 interi, si richiede di vedere A come un array X[lim1][lim2][lim3] completamente definito[1]. Si legge da "input" un intero HF (0<=HF<lim2) e si devono usare i valori della H-fetta[2] HF di X, ordinati per tasselli, per costruire una lista concatenata L i cui nodi contengono i valori della h-fetta nei loro campi info. Vediamo tutto questo con un esempio.

Esempio 1. Sia lim1=3, lim2=4 e lim3=5 e siano questi i lim1*lim2*lim3=60 valori letti in A:

strato 0	strato 1	strato 2
0 1 2 3 4	0 1 2 3 4	0 1 2 3 4
5 6 7 8 9	5 6 7 8 9	5 6 7 8 9
10 11 12 13 14	10 11 12 13 14	10 11 12 13 14
15 16 17 18 19	15 16 17 18 19	15 16 17 18 19

Per facilitare l'esempio i 3 strati sono identici. Con questi valori in X, la H-fetta 0 ordinata per tasselli sarà composta dai seguenti valori: 0 0 0 1 1 1 2 2 2 3 3 3 4 4 4, in cui abbiamo separato i 5 tasselli con uno spazio aggiuntivo. La H-fetta di indice 2 sarà composta da: 10 10 10 11 11 11 12 12 12 13 13 13 14 14 14. Supponiamo di voler usare la H-fetta 2 per costruire una lista L. Allora L sarà composta dai seguenti 15 nodi: 10->10->10->11->11->11->12->12->12->13->13->13->14->14->14

Dopo aver costruito la lista concatenata L nel modo appena illustrato, va letto da input un array S di interi che terminano con la sentinella 1000. I valori di S indicano spostamenti da compiere sulla lista L: valori positivi indicano spostamenti verso destra, valori negativi indicano spostamenti verso sinistra. zero indica di restare fermi. Il seguente esempio spiega questa operazione:

Esempio 2. Se S=2 -1 8 -3 20 allora, se partiamo dall'inizio di L, eseguire gli spostamenti richiesti da S significa muoversi sui seguenti nodi: andare al nodo di indice 2 (info=10), poi sul nodo di indice 1 (info=10), poi sul nodo di indice 9 (che sarebbe il primo con info=13) poi muoversi sul nodo di indice 6 (primo con info=12), l'ultimo elemento di S che è 20 in realtà è troppo grande perché 20+6 è maggiore di 14 che è l'indice dell'ultimo elemento della H-fetta. In questo caso ci dovremmo fermare sull'ultimo elemento della H-fetta, cioè quello di indice 14. Lo stesso succede se i valori di S ci chiedessero di andare verso sinistra al di là del primo nodo di L, per esempio, se l'ultimo elemento di S fosse -20 anziché 20, partendo da 6, dovremmo fermarci sul primo nodo di L, cioè il primo con info=10.

Esempio 3. C'è una maniera efficiente per compiere le operazioni appena descritte. La spieghiamo usando gli esempi precedenti: si parte dal primo nodo di L=10->10->10->11->11->11->12->12->12->13->13->13->14->14->14 e, visto che S[0]=2, ci si deve spostare al nodo di indice 2. Questa situazione la possiamo rappresentare con 2 liste nel modo seguente:
10<-10 10->11->11->11->12->12->12->13->13->13->14->14->14, cioè con 2 liste dove la prima rappresenta i nodi saltati e la seconda contiene i nodi che vanno dal nodo su cui siamo arrivati, fino alla fine. Si osservi che la prima lista è orientata al contrario, cioè dal nodo di indice 1 verso quello di indice 0. In questo modo è facile gestire successivi spostamenti sia a sinistra che a destra, come mostreremo continuando l'esempio: S[1]=-1, spostarci di 1 posto verso sinistra ci porta alle seguenti 2 liste:
10 10->10->11->11->11->12->12->12->13->13->13->14->14->14, mentre il successivo spostamento causato da S[2]=8 ci porta nella seguente situazione:
10<-10<-10<-11<-11<-11<-12<-12<-12 13->13->13->14->14->14, da cui, con S[3]=-3 andremo a:

[1] Deve essere chiaro che l'array X non deve essere definito in alcun modo. Esiste solo l'array A che va visto come X[lim1][lim2][lim3].

[2] La H-fetta i di un array a 3 dimensioni come X[lim1][lim2][lim3], con 0<=i<lim2, è costituita dalle righe di indice i dei lim1 strati di X. Nel presente esercizio gli elementi delle H-fette sono considerati in ordine per tasselli, dove il tassello (i,j), 0<=i<lim2 e 0<=j<lim3, è costituito dagli elementi X[0][i][j], X[1][i][j],...X[lim1-1][i][j].

10<-10<-10<-11<-11<-11 12->12->12->13->13->13->14->14->14 (chiamiamo questa situazione $), da ($) con S[4]=20 andremo a:
10<-10<-10<-11<-11<-11<-12<-12<-12<-13<-13<-13<-14<-14 14 che rappresenta il fatto di essere sull'ultimo nodo della lista. L'esercizio richiede si stampare su OUT il campo info dei nodi raggiunti dopo ciascuno spostamento. Nell'esempio appena esaminato, si deve stampare: 10 10 13 12 14.
Se da ($) applicassimo lo spostamento -20 (anziché 20), allora ci troveremmo in :
lista vuota 10->10->10->11->11->11->12->12->12->13->13->13->14->14->14
cioè sul primo nodo della lista. Questa è in realtà la situazione iniziale. In questo caso dovremmo stampare 10 10 13 12 10. Questo esempio mostra che la prima lista può svuotarsi mentre la seconda deve contenere sempre almeno un nodo visto che il nodo su cui ci si trova è sempre il primo nodo della seconda lista. Inoltre sottolineiamo che la prima lista è orientata in modo opposto alla seconda. Questo permette di spostare i nodi da una lista all'altra sottraendoli dall'inizio di una lista ed aggiungendoli all'inizio dell'altra. Si osservi che tutte le coppie di liste, elencate in questo esempio, **rappresentano** la lista iniziale. Infatti, per recuperare da esse la lista iniziale è sufficiente spostare tutti i nodi di L1 in L2.

Esercizio iterativo. Questa parte riguarda la lettura di S da IN e la costruzione di L. La funzione leggiS che legge S da IN è data, ma essa fa uso della funzione allunga il cui compito è quello di **allungare l'array S di 3 posizioni** rispetto a prima. La funzione allunga è da fare. La funzione iterativa buildL che costruisce L è da fare. Essa è specificata come segue:

PRE=(A contiene almeno lim1*lim2*lim3 valori, 0<=HF<lim2)
nodo* buildL(int*A,int lim1, int lim2, int lim3, int HF)
POST=(restituisce una lista di lim1*lim3 nodi i cui campi info sono i valori della H-fetta HF considerati in ordine per tasselli)

Attenzione: buildL deve usare una funzione calc_coord che calcola le coordinate (strato, riga, colonna) in X di un qualsiasi elemento di una H-fetta (considerata in ordine per tasselli). Anche la struttura FIFO può facilitare la realizzazione di buildL. Le strutture coord e FIFO sono definite nel programma dato.

Esercizio ricorsivo. Si tratta di realizzare la funzione ricorsiva F, che ricorre su S, ed esegue gli spostamenti su L richiesti da S, stampando i campi info dei nodi raggiunti. F deve soddisfare le seguenti specifiche:

PRE=(L1 ed L2 sono liste corrette, con L2 non vuota, S ha dimS elementi, vL1=L1 e vL2=L2)
void F(nodo*&L1, nodo*& L2, int*S, int dimS, ofstream & OUT)
POST=(L1=0 e L2 è la lista rappresentata da vL1 e vL2, inoltre sono stati eseguiti gli spostamenti richiesti da S[0..dimS-1] (partendo dalla situazione rappresentata da vL1 e vL2) ed è stato stampato il campo info dei nodi raggiunti dopo ciascuno spostamento)

Attenzione: F deve seguire la tecnica delle 2 liste illustrata nell'esempio precedente. Per farlo non c'è bisogno di usare la struttura FIFO. Conviene che F usi (almeno) 2 funzioni ausiliarie anch'esse ricorsive. Una per gli spostamenti verso destra ed una per quelli verso sinistra. Si osservi che la POST richiede che, al ritorno di F, L1 sia vuota e L2 sia l'intera lista. Cioè, quando S finisce, i nodi di L1 vanno trasferiti in L2 ricomponendo la lista originale (quella costruita da buildL).
IMPORTANTE: la funzione F non crea né distrugge alcun nodo, si limita a spostare opportunamente quelli di L costruita da buildL.

Correttezza:
1) delineare la prova di correttezza induttiva della correttezza di F rispetto a PRE e POST date.
2) descrivere l'invariante del ciclo principale di buildL.
3) si cerchi di dare una pre ed una postcondizione alle funzioni ausiliarie d F.