Esame di Programmazione del 3/7/2014

Il problema riguarda il pattern matching di un pattern in un testo, dove entrambi sono array di interi. Il pattern matching viene però affrontato in una maniera piuttosto originale. Sia P il pattern e T il testo. P ha lunghezza dimP. Per ogni elemento P[i] del pattern si costruisce una lista L_i delle posizioni j di T tali che T[j]=P[i]. Queste liste hanno tutte le informazioni necessarie per trovare tutti i match che ci interessano. Infatti, se L_i contiene j, allora ci basterà controllare che L_(i+1) contiene j+1 e che L_(i+2) contiene j+2 e così via, per sapere che T[j]=P[i], T[j+1]=P[i+1], T[j+2]=P[i+2], eccetera. Insomma che a partire dalla posizione j, T contiene un match di P[i..].

Precisiamo: cerchiamo il match di lunghezza massima di una qualsiasi sotto-sequenza di P in T. Consideriamo solo match contigui.

Esempio1: Supponete che T=[1,2,0,1,3,2,1,0,0,0,3,0,1,2] e P=[1,0,1,3]. Le 4 liste che si devono costruire con i match in T di P[0], P[1], P[2], e P[3] in T, sono:
L_0=0, 3, 6, 12
L_1=2, 7, 8, 9, 11
L_2= 0, 3, 6, 12
L_3= 4, 10
Il match di lunghezza massima è quello che inizia nella posizione 1 di P ed ha lunghezza 3, precisamente è la sotto-sequenza [0,1,3] di P ed il match inizia in T nella posizione 2. Rappresenteremo un tale match con la seguente tripla di interi [ip=1, im=2, fm=4], dove ip è l'inizio della sottosequenza di P che viene matchata, im è l'indice di T in cui il match inizia e fm l'indice di T in cui il match finisce. Nel file dato trovate la struttura 'match' che ha i tre campi appena descritti e che dovrete usare per rappresentare i match trovati.

File dato: come sempre trovate un file col main che esegue tutto l'i/o richiesto . Il file contiene anche le dichiarazioni delle strutture che servono e la funzione FM (con Pre e Post). Dovete aggiungere a questo file le funzioni richieste nell'esercizio. Si raccomanda di non scrivere le asserzioni di correttezza all'interno del codice, ma dopo le funzioni cui si riferiscono. Lo stesso vale per le dimostrazioni.

Programmazione iterativa:
Si chiede di scrivere una funzione iterativa M che costruisce la lista di liste dei match descritte prima. I nodo della lista che M deve costruire sono del tipo 'nodoG' che trovate nel file dato. Ogni nodoG ha un campo N a cui è possibile appendere una lista L_i. Le liste L_i hanno nodi del tipo 'nodo' con campo info intero. Vedi file dato.

Esempio2: rispetto all'esempio precedente, M deve costruire una lista di 4 nodi 'nodoG' tale che al primo nodo sarà appesa L_0, al secondo L_1 e così via. Nel seguito chiameremo G la lista prodotta da M.

PRE e POST e segnatura di M sono queste:

PRE_M=(T è un array di dimT interi, P un array di dimP interi, dimT e dimP>=0)
nodoG* M(int* T, int dimT, int* P, int dimP)
POST_M=(M restituisce una lista di dimP liste tale che la lista L_i (i in [0..dimP-1]) contiene gli indici di tutte le occorrenze di P[i] in T, da sinistra a destra)

Attenzione: è consigliabile definire anche altre funzioni che permettono di semplificare M. Per ogni funzione va specificata PRE e POST.

Correttezza iterativa: specificare l'invariante e la post-condizione del ciclo principale di M. Mostrare la correttezza del ciclo stesso.

Programmazione ricorsiva: Nel file dato trovate una funzione FM che è iterativa ed invoca FM1_ric che dovete scrivere voi e deve essere ricorsiva. FM esamina le diverse liste L_0, L_1,... di G e cerca iterativamente il massimo match con inizio in P[0], e poi con inizio in P[1], e in P[2],... ricordandosi in ogni momento il match di lunghezza massima trovato fino a quel momento.
La funzione ricorsiva FM1_ric deve calcolare il massimo match a partire da P[i..] (per i in [0..dimP-1]). Lo fa usando L_i e la parte di G che inizia dal nodo i+1.

Nel seguito consideriamo la lista L_i e chiamiamo G' la parte di G che inizia al suo nodo di indice i+1, cioè la parte di G a cui sono appese le liste L_(i+1), L_(i+2), ecc. Chiamiamo S_(G',L_i) l'insieme che contiene tutte le sequenze S di nodi n_0, n_1, n_2, ..,n_m, tali che essi appartengano rispettivamente a L_i, L_(i+1), L_(i+2), ..L_(i+m) e tali che n_0->info=x, n_1->info=x+1, n_2->info=x+2,..., n_m->info=x+m. Ogni sequenza S di S_(G',L_i), come descritta prima, rappresenta un match di P[i..i+m] in T che inizia in posizione x e finisce in posizione x+m di T. Un tale match è descritto dall'apposita struttura 'match' che assume il valore, [ip=i, im=x, fm=x+m]. La lunghezza del match è fm-im+1, che in questo caso è m+1. Nel seguito identifichiamo ciascuna sequenza S con il corrispondente valore di tipo 'match'. Indichiamo con MAX_S_(G',L_i) il valore di tipo 'match' in S_(G',L_i) che ha massima lunghezza fm-im+1. Qualora esistessero vari elementi di S_(G',L_i) di lunghezza massima, MAX_S_(G',L_i) è quello col minimo valore del campo im.

PRE_FM1_ric=(G sia una lista corretta di liste tutte corrette e contenenti valori interi non negativi e crescenti, e sia L una lista corretta e possibilmente vuota di interi non negativi e i>=0)
match FM1_ric(nodoG*G, nodo* L, int i)
POST_FM1_ric=(la funzione restituisce il valore [i, x, y] che corrisponde a MAX_S_(G,L) . Se L è vuota, allora FM1_ric restituisce [0,0,-1] a indicare l'assenza di match)

Esempio3: usiamo la lista G dell'Esempio 2. Allora, se invochiamo FM1_ric, passando G->next come primo parametro, G->N come come secondo parametro, e 0 come terzo, FM1_ric ci restituirà il valore [0,6,7][1] che indica che solamente per il 6 in G->N=L_0 c'è 6+1=7 in L_1 e indica anche che L_2 non contiene 8. Quindi MAX_S_(G->next,L_0) è [0,6,7]. Si osservi che ci sono anche altre sequenze in S_(G->next,L_0), ma di lunghezza 1. Se invece invochiamo FM1_ric con i parametri (G->next->next, G->next->N, 1) esso ci ritorna il valore: [1,2,4], in accordo a quanto visto nell'esempio 1. Questo è il match massimo in assoluto e resterà quindi il risultato finale di FM. Per esercizio, consideriamo anche l'invocazione di FM1_ric con parametri (G->next->next->next, G->next->next->N, 2). Essa ci restituisce il valore [2, 3, 4].

Correttezza ricorsiva: dimostrare induttivamente la correttezza di FM1_ric. Assumere che le eventuali funzioni ausiliarie, siano corrette rispetto alla loro PRE e POST (che voi dovete specificare).

Attenzione :
i)conviene introdurre altre funzioni ausiliarie che verranno invocate da FM1_ric. Queste funzioni dovranno avere PRE e POST.
ii)si deve prestare attenzione alla definizione dei campi dei match che le funzioni devono restituire in particolare nei casi base.

Integrazione: al posto della correttezza, realizzare una funzione ricorsiva che faccia quanto richiesto alla funzione M.

[1] Osserva che [0,6,7] è un valore di tipo 'match' dove 0 è il valore del campo ip, mentre 6 e 7 sono i valori di im e fm, rispettivamente. I nomi dei campi non vengono ripetuti per semplicità.