Esercizio 1 del 5/5/2015 da consegnare per l'11/5

Si tratta di un esercizio ricorsivo in cui si vuole fare pattern matching di un array int P[20] con dimP elementi definiti in un array int T[200] che contiene dim elementi definiti. A differenza di altri esercizi già visti di pattern matching, ora vogliamo considerare anche il caso di match di porzioni (contigue) di P.

Esempio: se dimP=5 e P=[1,0,1,2,1], allora oltre al match di tutto P, dobbiamo considerare anche quello di [0,1,2,1] (gli ultimi 4 elementi di P), e quello di [1,0,1] (i primi 3 elementi) e di [0,1,2] (i 3 elementi centrali di P) e di [2,1] (gli ultimi 2). Insomma dobbiamo considerare ogni possibile porzione contigua di P. Una porzione sarà caratterizzata da un inizio in P ed una lunghezza. Per esempio [1,0,1] ha inizio=0 e lunghezza=3, mentre [2,1] ha inizio= 3 e lunghezza=2 e [0,1,2] ha inizio=1 e lunghezza=3. In caso una stessa porzione appaia più volte nel pattern, si deve considerare l'inizio minimo.

Per una certa porzione di P, per esempio [0,1,2] con inizio=1 e lunghezza=3, un match in T di questa porzione è una porzione T[i,i+1,i+2] di T che sia identica a [0,1,2]. Un tale match ha inizio in T alla posizione i.

Quindi useremo la seguente struttura (con costruttore) per modellare un match di una porzione di P in T:

struct M {int lung,inizioP,inizioT; M(int a=0, int b=0, int c=0){lung=a; inizioP=b; inizioT=c;}};

L'idea è che il campo lung contenga la lunghezza della porzione di P che viene matchata, inizioP contenga la posizione in P in cui inizia la porzione matchata e inizioT indichi la posizione in T in cui inizia il match.

Si tratta di scrivere una funzione ricorsiva che determini il match di una porzione di P in T di lunghezza massima e che produca valore di tipo M che rappresenti questo match. In caso ci siano match diversi con la stessa lunghezza, si chiede quello con inizioP minimo ed in caso ce ne siano 2 con la stessa lunghezza e stesso inizioP, allora si deve scegliere quello con inizioT massimo.

Esempio: sia dimP=5, P=[1,0,1,2,1], dim=10 e T=[1,1,0,1,3,2,1,0,0,0]. Il match di lunghezza massima è quello della porzione [1,0,1] di P che è rappresentato dalla tripla [lung=3, inizioP=0, inizioT=1]. Questa tripla dovrebbe essere scritta su "output" dal programma richiesto. Se T=[1,1,0,1,3,2,1,0,1,0], allora [1,0,1] avrebbe 2 match caratterizzati da [lung=3, inizioP=0, inizioT=1] e da [lung=3, inizioP=0, inizioT=6] e la seconda è maggiore della prima tripla perché ha inizioT maggiore, a parità delle componenti lung e inizioP. Quindi in questo caso il programma dovrebbe scrivere su "output" la seconda tripla. Esiste anche il caso che nessuna porzione di P trovi un match su T (nessun valore di P compare in T). In questo caso il programma deve stampare [lung=0, inizioP=-1, inizioT=-1].

Cosa c'è da fare:

a) viene dato un main che esegue l'i/o e invoca la funzione ricorsiva match del punto (b) da fare;

b) va fatta una funzione ricorsiva match con il seguente prototipo e che deve essere corretta rispetto alle seguenti pre- e post-condizioni:

PRE=(dimP>0, dim>0, T[0..dim-1] è definita, P[0..dimP-1] è definita, 0<=indiceT<=dim)

M match(int*T, int sim, int*P, int dimP, int indiceT)

POST=(restituisce un valore M che rappresenta il massimo match in T[indiceT..dim-1]di una porzione di P (secondo l'ordine descritto prima: lunghezza massima, a parità di lunghezza, inizioP minimo e, a parità di

lunghezza e inizioP, inizioT massimo), qualora non ci siano match, la funzione deve restituire [lung=0, inizioP=-1, inizioT=-1])

La funzione match deve usare almeno un'altra funzione ricorsiva.

Correttezza: è richiesta la pre- e post-condizione delle funzioni ausiliarie. La correttezza di match va dimostrata induttivamente.

Consiglio: non preoccupatevi se, almeno in prima istanza, la vostra soluzione sembra piuttosto inefficiente. Cercate prima di tutto la semplicità.