Esame di Programmazione del 8/972021 Parte ricorsiva 1 ora di tempo

Affrontiamo il problema del pattern matching in un modo diverso dal solito. Il testo T e il pattern P sono array di interi di dimT e dimP elementi, rispettivamente. Immaginiamo di avere un array di booleani di dimT*dimP elementi che, per la chiarezza dell'esposizione, tratteremo nel seguito come se avesse 2 dimensioni benché ne abbia 1 sola. Assumiamo che M sia riempita di valori che soddisfano la seguente condizione: M[i][j] è true sse P[j]==T[i].

Esempio 1: supponiamo che T=[2, 2, 3, 2, 3, 1, 3, 1, 3, 2] e P=[0, 3, 1, 3], allora M è come segue:

0 0 0 0
0 0 0 0
0 1 0 1
0 0 0 0
0 1 0 1
0 0 1 0
0 1 0 1
0 0 1 0
0 1 0 1
0 0 0 0

La prima colonna che contiene solo false, mostra che P[0] non appare mai in T, mentre la seconda che P[1] appare in T in posizione 2,4,6 e 8. E così via per le successive 2 colonne.

Usando l'array M è possibile costruire una funzione che calcoli il massimo match di P in T. Siamo interessati a trovare in T il match di lunghezza massima dei sotto-array di P, cioè P[i..i+m], i>=0, m>=0 e i+m<dimP, e, in caso di diversi match massimi di pari lunghezza, vogliamo il match che inizia più a sinistra in T e in caso di ulteriore parità, vogliamo quello che inizia prima in P.

Esempio 2: consideriamo nuovamente T, P ed M dell'Esempio 1. Da M è facile vedere che non esistono match che iniziano in P[0]. Consideriamo quindi i match che iniziano con P[1], il suo primo match inizia in T[2], infatti M[2](1]=true, per sapere se il match continua basta controllare M[3][2]. Putroppo M[3][2] è 0 e quindi il match ha lunghezza 1. Però in T[4] c'è un altro match di P[1], per sapere se si prolunga, basta controllare M[5][2] che infatti è 1, quindi abbiamo trovato un match di P[1,2] in T[4,5]. Controlliamo ora M[6][3] che è anch'esso true e quindi abbiamo trovato un match di lunghezza 3: P[1..3]=T[4..6]. A questo punto il pattern è finito e questo è il match a lunghezza massima (visto che P[0] non appare in T). Un tale match è definito da 3 interi: l'inizio del match in T=4, l'inizio in P=1 e la lunghezza=3. La seguente struttura serve a rappresentare questi valori:

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

Si chiede di scrivere una funzione ricorsiva:

PRE=(M è la matrice descritta in esempio 1 per T e P, 0<=t<=dimT, 0<=p<=dimP)

tripla matchR(bool*M, int t, int p, int dimT, int dimP)

POST=(restituisce la tripla che definisce il massimo match di P in T, in caso match di uguale lunghezza, restituisce quello più a sinistra in T e in P)

Attenzione:

1) M è un array a 1 dimensione
2) è consigliabile definire funzioni **ausiliarie** che devono essere **ricorsive**.

Correttezza: Specificare PRE e POST delle funzioni ausiliarie. Dimostrare la correttezza di matchR.