Esame di programmazione del 12/7/2021

Parte Ricorsiva, l'esame dura 1 ora con 10 minuti in più per la consegna

Si tratta di un problema di pattern matching dove il testo e il patterno sono array di interi. Come sempre il testo è l'array T di dimT elementi e il pattern è l'array P di dimP elementi.

Vogliamo considerare tutti i sotto-array di P (cioè P[i..i+m] con 0<=i e i+m <dimP) e vogliamo determinare il sotto-array di lunghezza massima per cui ci sia un match in T. Il match che consideriamo deve essere contiguo, quindi per il sotto-array P[i..i+m] cerchiamo T[j...j+m]=P[i..i+m]. Vogliamo trovare il sotto-array di P più lungo per cui ci sia un tale match in T.

Esempio. Sia T=[3,1,4,5,2,1,1,2,3,2,1] e P=[1,3,2,4] il match più lungo è quello del sotto-array [3,2] di P. Questo match viene caratterizzato da una tripla (2,8,1) che specifica che la lunghezza del match è 2, che inizia in T[8] e riguarda il sotto-array P[1..2]. Se P fosse [3,3,3,1] il massimo match sarebbe quello del sotto-array [3,1] di P a partire dalla posizione 0 di T. La corrispondente tripla sarebbe (2,0,2). Per rappresentare queste triple dovete usare la seguente struttura:

struct triple{int L,TT, PP; triple(int a=0, int b=0, int c=0){L=a;TT=b;PP=c;}}; in cui L deve contenere la lunghezza del match, TT la posizione in T in cui inizia il match e PP la posizione in P in cui inizia il sotto-array matchato.

Si chiede di scrivere una funzione **ricorsiva** triple matchR(int*T, int dimT, int iT int*P, int dimP, int iP) che sia corretta rispetto alle seguenti PRE e POST:

PRE=(T ha dimT elementi definiti e P ne ha dimP, dimT e dimP > 0, 0<=iT<=dimT, 0<=iP<=dimP)

POST=(restituisce la tripla che individua il match di lunghezza massima dei sotto-array di P[iP..dimP-1] in T[iT..dimT-1])

In caso ci fossero diversi match di lunghezza uguale e massima, si chiede quello che inizia più a sinistra in T e anche in P.

Attenzione:

a) i valori dei parametri iT e iP della prima invocazione di matchR sono entrambi 0. Questi 2 parametri servono a scorrere T e P e infatti T,P, dimT e dimP non dovrebbero cambiare;
b) si consiglia di definire funzioni ausiliarie che devono essere anch'esse **ricorsive** e avere PRE e POST definite da voi;
c) cercate una soluzione semplice. La ricorsione può essere di tipo 1. Non servono ottimizzazioni astute.

Correttezza:

Dare la dimostrazione di correttezza di matchR. Nella dimostrazione potete dare per scontato che le invocazioni delle eventuali funzioni ausiliarie siano corrette rispetto alle PRE e POST che avrete specificato per esse.