Esame di Programmazione del 8/972021 Parte iterativa 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. Vogliamo definire un array M di booleani di dimT*dimP elementi che, per la chiarezza dell'esposizione, tratteremo nel seguito come se avesse 2 dimensioni benché abbia 1 sola dimensione. M va riempita di valori in modo che sia soddisfatta 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 colonna mostra che P[1] appare in T nelle posizioni 2,4,6 e 8. E così via per le successive 2 colonne (ovviamente la quarta colonna è identica alla seconda).

Usando l'array M è semplice costruire una funzione che calcoli il massimo match di P in T. Siamo interessati a trovare in T dei match contigui di sotto-array di P, cioè P[i..i+m] con i >=0, m >=0 e i+m<dimP, e vogliamo trovare il match più lungo. In caso di match di pari lunghezza, vogliamo quello che inizia prima in T e in caso di ulteriore parità, vogliamo quello che inizia prima in P. Nel seguente esempio spieghiamo come usare M per trovare il match più lungo.

Esempio 2: consideriamo nuovamente T, P ed M dell'Esempio 1. Da M è facile vedere che non esistono match che iniziano con 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], e, di nuovo, per sapere se si prolunga, basta controllare M[5][2] che è 1, quindi abbiamo trovato un match di P[1,2] in T[4,5]. Controlliamo ora M[6][3] che è true e quindi abbiamo trovato un match di lunghezza 3: P[1..3]=T[4..6]. A questo punto P è finito e quindi questo è il match di lunghezza massima (visto che P[0] non appare in T) e anche quello che inizia più a sinistra 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 tripla serve a rappresentare queste triple di 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 2 funzioni **iterative**:

1) void computeM(int*T, int*P, int dimT, int dimP, bool*M)

PRE=(T e P hanno dimT e dimP elementi definiti e ne ha M dimT*dimT)

POST=(M viene riempita con valori che soddisfano l'Esempio 1)

2) tripla match(bool*M, int dimT, int dimP)

PRE=(M ha dimT*dimP elementi definiti)

POST=(restituisce la tripla che definisce il massimo match di P in T, in caso di match di uguale lunghezza, restituisce quello che inizia prima in T e, in caso di parità, che inizia prima in P)

Attenzione: M va definita e trattata come un array a 1 dimensione

Correttezza: Specificare gli invarianti dei cicli delle 2 funzioni.