Compitino 2 di Programmazione del 10/4/2018

L'esercizio riguarda il problema del pattern matching. Viene dato un testo T con dimT valori interi ed un pattern P con dimP interi. Vanno ricercati match di P in T contigui, ma non necessariamente completi. In sostanza, si cerca di matchare P[0] e, se funziona, si cerca di matchare P[1] e così di seguito, finchè c'è match. Quando il match fallisce (oppure quando P finisce) ci si ferma anche se possono restare elementi alla fine di P non matchati. La lunghezza del match è uguale al numero di elementi di P matchati. Spieghiamo con un esempio.

Esempio. Sia T=[1,2,0,3,5,1,1,1,1,2,3,5] e P=[1,1,2,5]. Osserviamo i seguenti match di P in T:

- Nella posizione 0 c'è un match di lunghezza 1: solo P[0] = T[0], poi P[1] != T[1].
- Nella posizione 5 c'è un match di lunghezza 2: si matchano i primi 2 elementi di P.
- Nella posizione 6 succede lo stesso che nella 5.
- Nella posizione 7 c'è un match di lunghezza 3: si matchano i primi 3 elementi di P.
- Nella posizione 8 c'è un match di lunghezza 1.

In questo esempio non ci sono match di P completi. Vengono matchati al massimo i primi 3 elementi di P. In generale è comunque possibile che ci siano match completi.

Viene richiesto di scrivere un programma capace di determinare l'indice di inizio e la lunghezza del match più lungo, che nell'esempio precedente, sarebbe quello che inizia in posizione 7 ed ha lunghezza 3. Il main già contiene l'istruzione di stampa che produce l'output richiesto. Basta calcolare i valori giusti per le variabili da stampare.

In caso ci siano diversi match più lunghi degli altri e di lunghezza uguale tra loro, va considerato quello che inizia nell'indice minimo. Si osservi che è anche possibile che non ci siano match di lunghezza maggiore di 0. In questo caso nel main è previsto un output che segnala la cosa. A questo proposito, si osservi la POST_F data qui sotto.

Il programma da fare deve consistere di 2 funzioni, una iterativa (F) ed una ricorsiva (match) che soddisfano le seguenti specifiche:

PRE_F=(T ha dimT elementi definiti e P ne ha dimP)

void F(int*T, int*P, int dimT, int dimP, int & inizio, int & lung)

POST_F=(se c'è un match di P in T di lunghezza maggiore di 0, la funzione restituisce in inizio l'indice di inizio di del match più lungo e in lung la sua lunghezza, altrimenti restituisce inizio =-1)

La funzione F deve usare una funzione **ricorsiva** match che obbedisce alle seguenti specifiche:

PRE_m=(T ha almeno dimT elementi definiti e P ne ha dimP)

int match(int*T, int*P, int dimT, int dimP);

POST_m=(restituisce la lunghezza del più lungo match di P **che inizia** in T[0])

NOTA: è importante che la funzione match obbedisca alla POST_m data. In particolare deve calcolare la lunghezza del match che inizia in T[0] e non deve considerare altri match che inizino in posizioni successive di T.

Correttezza:

1) Dare un invariante significativo per il ciclo di F
2) Dimostrare per induzione la correttezza di match rispetto alla PRE_m e POST_m.