Esercizio 2 del 24/4/2017

E' una variazione dell'esercizio del II compitino del 19/4/2017. Vogliamo considerare che il pattern possa contenere anche dei simboli *, con l'idea che uno * possa venire matchato con una qualsiasi stringa (anche vuota) del testo. Le restanti parti del pattern (cioè non *) devono però venire matchate in modo contiguo. Un esempio ci aiuta a chiarire l'idea.

Esempio 1. Sia T= ababcccdabeeaadcdca e P=a*ca, allora un match di P in T deve trovare una a, poi un qualsiasi numero di elementi di T (anche 0 elementi) può matchare *, dopo di che si deve trovare c e poi a di seguito. E' possibile soddisfare questa richiesta in T? La risposta è si, nel modo seguente: troviamo a in T[0], poi matchamo * con tutti i caratteri da T[1] fino al terzultimo e finalmente matchamo ca nelle ultime due posizioni di T. Ovviamente se P=aca, non ci sarebbe alcun match di P in T, in quanto, senza *, il match dovrebbe essere completamente contiguo. Se invece P=a*c*a, allora ci sarebbero molti altri match oltre a quello trovato prima in quanto basterebbe trovare una a, poi una c e ancora una a in questo ordine, ma in posizioni anche non contigue. Nell'esercizio è richiesto il primo match cioè quello nel quale ogni pezzo del pattern è matchato appena possibile. Quindi per P=a*c*a, il primo match è, P[0] con T[0], P[1] con T[4] e P[2] con T[8].

Parte 1: scrivere una funzione con il seguente prototipo:

bool match(char* T, int dimT, char* P, int dimP, int* X)

che rispetti le seguenti pre e postcondizioni:

PRE=(T contiene dimT caratteri, P ne contiene dimP e X ha 2*dimP elementi, dimT e dimP>0)

POST=(la funzione restituisce true sse esiste un match di P in T, secondo le modalità di match illustrate nell'Esempio 1) &&(se restituisce true allora X contiene la rappresentazione del primo match trovato secondo quanto spiegato nell'esempio che segue)

Esempio 2. Consideriamo i casi dell'Esempio 1. Per P=a*ca, la funzione match dovrà restituire true e X dovrà contenere la seguente rappresentazione del primo match: 0, 1,15,2, che va letto nel modo seguente: dopo aver saltato 0 elementi di T, troviamo P[0] , quindi 1 elemento del pattern, poi saltiamo 15 elementi di T e poi troviamo 2 elementi del pattern.

Per P=aca, match deve restituire semplicemente false. Per P=a*c*a, match deve restituire true e in X, la rappresentazione del primo match illustrato anche in Esempio 1: 0, 1,3,1,3,1.

Consideriamo, anche P=ba*bc*dc, match deve restituire true e in X, i valori: 1,4,9,2. In questo caso il primo * viene matchato con la sequenza vuota di caratteri in T. Anche se non l'abbiamo considerato esplicitamente, ci potrebbero essere anche più * contigui nel pattern. Per esempio P potrebbe essere *ba**bc**dc*. Ovviamente 2 * hanno lo stesso effetto di 1 * e un * all'inizio e alla fine non ha alcun effetto sul match. E' anche possibile che P sia composto solo da *. In questo il match ci sarebbe in maniera triviale e quindi X dovrebbe restare vuoto.

Parte 2: scrivere una funzione stampa col seguente prototipo: void stampa(char*P, int dimP, int*X,int num). L'ultimo parametro num indica il numero di coppie presenti in X. Stampa ha il compito di stampare su cout i valori contenuti in X nel modo seguente. Consideriamo l'ultimo caso dell'Esempio 2, la stampa da eseguire è la seguente:

dopo 1 elementi

troviamo babc

dopo 9 elementi

troviamo dc

Se match restituisce false, il main deve semplicemente stampare su cout “match fallito”. Se invece match restituisce true, il main deve invocare stampa.

Suggerimento: conviene definire funzioni ausiliarie. Per ognuna va specificata la sua pre e postcondizione.

Attenzione: la stampa deve essere capace di ignorare gli * eventualmente presenti nel pattern P. La cosa non è così facile visto che gli * non sono contati in X.