Esame di Programmazione 19/20 del 15/1/2020

L'esercizio riguarda un pattern matching da fare sui cammini di un albero binario t. Il pattern è, come sempre, un array P di dimP interi. La maniera di affrontare il problema che dovete seguire è la seguente:

l'albero t viene attraversato in profondità seguendo l'ordine prefisso e mentre si percorre un cammino, si riempie un'array path (di interi) con i campi info dei nodi attraversati. Questa procedura si ferma sulle foglie di t e a quel punto l'array path contiene i campi info del cammino percorso dalla radice ad una foglia. A questo punto si cerca un match di P su path. Si cercano solamente match completi (tutto P deve essere trovato) e contigui (elementi immediatamente successivi di P sono matchati con elementi immediatamente successivi di path).

Esempio: sia t= 4(6(2(3(_,_),1(_,_)),4(_,_)),5(8(4(_,_),_),9(_,11(12(_,_),_)))), e P=[5,9,11]. Si deve assumere che path abbia un numero di elementi (inizialmente tutti indefiniti) che è certamente maggiore della profondità di t. Quindi nessun controllo sui limiti di path è mai necessario. La foglia più a destra di t è quella che corrisponde al cammino 000, il suo campo è info=3 e quando la funzione richiesta arriva a questa foglia, path dovrà contenere [4,6,2,3]. Su questo cammino non c'è alcun match di P. Quindi si passa alla foglia successiva che corrisponde al cammino 001, path dovrà diventare [4,6,2,1]. Di nuovo nessun match. La prossima foglia corrisponde al cammino 01, corrispondentemente path diventerà, [4,6,4]. Ancora nessun match. La prossima foglia corrisponde al cammino 100 e path diventa, [4,5,8,4]. Ancora niente. La prossima foglia corrisponde al cammino 1110 e path=[4,5,9,11,12]. Finalmente, su questo cammino si trova un match di P. Si osservi che il cammino 10 non viene considerato in quanto il nodo finale del cammino non è una foglia e quindi esiste un cammino più lungo che lo contiene e che quindi avrà maggiori chance di contenere un match. Lo stesso vale per il cammino 11.

E' importante capire che esiste un solo array path che viene usato per tutti i cammini. Insomma path[i] a turno servirà a contenere i campi info dei nodi di profondità i di t (da sinistra a destra, finché non si trova un match).

Si chiede di scrivere una funzione ricorsiva che soddisfi la seguente specifica:

PRE=(albero(t) è ben formato e non vuoto, P ha dimP>0 posizioni, depth >=0 e path ha più di depth + altezza(t) +1 posizioni)

bool prefix(nodo* t, int*P, int dimP, int* path, int depth)

POST=(se in t non esiste alcun cammino dalla radice ad una foglia che contiene un match di P, allora prefix restituisce false, se invece un tale cammino esiste, chiamiamo c quello più a sinistra, allora prefix restituisce true e a partire da path[depth], path contiene i campi info dei nodi attraversati dal cammino c)

Aiuto: depth in ogni momento sarà la profondità del nodo raggiunto. Invece path non cambia mai e punta sempre all'inizio dell'array. Osservate che la PRE richiede che t non sia vuoto.

Si chiede inoltre di definire una funzione iterativa che soddisfi la seguente specifica:
PRE=(P ha dimP >0 posizioni, path ha dim posizioni)

bool match(int* path, int dim, int*P, int dimP)

POST =(restituisce true sse path[0..dim-1] contiene un match contiguo e completo di P[0..dimP-1].

Correttezza:

-scrivere gli invarianti dei cicli di match e di eventuali funzioni ausiliarie (anch'esse iterative)

-dare la dimostrazione induttiva della funzione prefix rispetto alla specifica data.