Algoritmi e Strutture Dati
24 settembre 2021

Note

1. La leggibilità è un prerequisito: parti difficili da leggere potranno essere ignorate.
2. Quando si presenta un algoritmo è fondamentale spiegare l'idea e motivarne la correttezza.
3. L'efficienza e l'aderenza alla traccia sono criteri di valutazione delle soluzioni proposte.
4. Si consegnano i fogli di bella copia.

Domande

Domanda A (6 punti) Data la ricorrenza $T(n) = 6T(n/3) + (n-1)^2$, trovare la soluzione asintotica.

Soluzione: Si usa il master theorem con $a = 6$, $b = 3$, $f(n) = (n-1)^2$. Si deve confrontare $f(n)$ con $n^{\log_b a}) = n^{\log_3 6}$ ed essendo che $\log_3 6 < 2$ per qualunque $0 < \epsilon < 2 - \log_2 6$ si vede facilmente che $f(n) = \Omega(n^{\log_2 3+\epsilon})$.

Per concludere che $T(n) = \Theta(f(n)) = \Theta(n^2)$ usando il caso 3 del Master Theorem occorre dimostrare la regolarità di $f(n)$, ovvero che $af(n/b) < kf(n)$ per $0 < k < 1$, asintoticamente. Si imposta

$$af(n/b) = 6(n/3-1)^2 < k(n-1)^2$$

e si osserva

$$6(n/3-2)^2 = 6((n-6)/3)^2 < 6((n-2)/3)^2 = 6/9(n-2)^2 = 2/3(n-2)^2$$

per cui si può scegliere $k = 2/3 < 1$ e n qualunque.

Domanda B (6 punti) Si calcoli la lunghezza della longest common subsequence (LCS) tra le stringhe *armo* e *toro*, calcolando tutta la tabella $L[i,j]$ delle lunghezze delle LCS sui prefissi usando l'algoritmo visto in classe.

Soluzione: Si ottiene

		t	*o*	*r*	*o*
	0	0	0	0	0
a	0	0	0	0	0
r	0	0	0	1	1
m	0	0	0	1	1
o	0	0	1	1	2

La lunghezza della longest common subsequence tra *armo* e *toro* è quindi 2.

Esercizi

Esercizio 1 (10 punti) Dare la definizione di albero binario di ricerca. Scrivere una funzione `conta(T,a,b)` che dato un albero binario di ricerca `T` e due valori `a` e `b`, ritorna il numero di nodi `x` tali che `x.key` $\in$ [`a`, `b`]. Mostrarne la correttezza e valutarne la complessità.

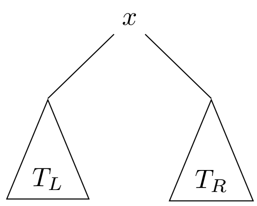
Soluzione: L'idea è quella di realizzare una funzione ricorsiva `contaRec(x,a,b)`, che dato un nodo x, restituisce il numero di nodi nel sottoalbero radicato in x con chiave nell'intervallo $[a, b]$. Quindi la si applica alla radice dell'albero.

```
conta(T,a,b)
        return contaRec(T,root, a,b)

contaRec(x, a, b)
    if x <> nil
       if b < x.key
                 return contaRec(x.left)
       else if x.key < a
          return contaRec(x.right)
       else
          return 1 + contaRec(x.right) + contaRec(x.left)
```

La correttezza della funzione ricorsiva può essere dimostrata per induzione sull'altezza h del sottoalbero radicato in x

- $(h = 0)$ In questo caso, $x = nil$, il sottoalbero è vuoto e quindi correttamente si ritorna 0.
- $(h > 0)$ In questo caso l'abero ha la forma



  Si distinuono tre casi:

  – se $b < x.key$, sicuramente, per le proprietà degli ABR, tutti i nodi nel sottoalbero T_R hanno chiave maggiore di $x.key$ e quindi di b, e pertanto non sono nell'intervallo. Dunque si ritorna il numero di nodi in $[a, b]$ nel sottoalbero T_L, che per ipotesi induttiva è fornito da $contaRec(x.left)$.
  – $x.key < a$ si ragiona in modo duale rispetto al caso precedente
  – altrimenti, $x.key \in [a, b]$ e i nodi con chiave in $[a, b]$ possono travarsi sia nel sottoalbero sinistro che nel sottoalbero destro. Si ritorna dunque $1 + contaRec(x.right) + contaRec(x.left)$, dove le chiamate $contaRec(x.right)$ e $contaRec(x.left)$ ritornano, per ipotesi induttiva, i nodi in $[a, b]$ nei sottoalberi T_R e T_L.

Nel caso peggiore, ad esempio quando tutti i nodi hanno chiave in $[a, b]$, la funzione esegue una visita completa dell'albero e quindi la complessità è $O(n)$ dove n è il numero dei nodi.

Esercizio 2 (9 punti) Si consideri un file definito sull'alfabeto $\Sigma = \{a, b, c\}$, con frequenze $f(a), f(b), f(c)$. Per ognuna delle seguenti codifiche si determini, se esiste, un opportuno assegnamento di valori alle 3 frequenze $f(a), f(b), f(c)$ per cui l'algoritmo di Huffman restituisce tale codifica, oppure si argomenti che tale codifica non è mai ottenibile.

1. $e(a) = 0,\ e(b) = 10,\ e(c) = 11$

2. $e(a) = 1,\ e(b) = 0,\ e(c) = 11$
3. $e(a) = 10,\ e(b) = 01,\ e(c) = 00$

Soluzione:

1. Questa codifica viene restituita dall'algoritmo di Huffman quando $f(b), f(c) < f(a)$: in questo caso i nodi associati a b e c vengono uniti creando un nuovo nodo interno, che poi viene unito al nodo associato ad a. Quindi, per esempio, $f(a) = 40, f(b) = 25, f(c) = 35$.
2. La codeword $e(a) = 1$ è un prefisso della codeword $e(c) = 11$, cioè questa codifica non è libera da prefissi, e quindi non è una codifica di Huffman.
3. L'albero associato a questa codifica non è pieno perché c'è un nodo interno con un solo figlio (quello nel cammino che porta alla foglia associata al carattere a). Quindi questa codifica non è ottima e quindi non è una codifica di Huffman.