Lezione 7

Un albero rosso-nero è un albero binario con un extra-bit di memoria: il colore, che può essere ROSSO o NERO.

Il loro impiego garantisce che nessun cammino (radice-foglia) sia più lungo del doppio di qualunque altro. Gli alberi rosso-neri sono (approssimativamente) bilanciati.

Ogni nodo contiene gli attributi: COLOUR, KEY, LEFT, RIGHT, PARENT.

PROPRIETÀ DEGLI ALBERI ROSSO-NERI

1. Ogni nodo è rosso oppure nero;
2. La radice è nera;
3. Le foglie (nil) sono nere;
4. Se un nodo è rosso, allora i suoi figli sono neri;
5. Per ogni nodo, ogni cammino (semplice) dal nodo alle foglie contiene lo stesso numero di nodi neri.

L'altezza nera di un nodo x, $bh(x)$ è il numero di nodi neri lungo un cammino da x a una delle foglie (escludendo x stesso). È ben definita (prop. 5).

L'altezza nera di un albero rosso-nero è l'altezza nera della radice.

Lemma: Un albero rosso-nero con n nodi interni ha altezza al più $2\log(n+1)$, cioè $O(\log n)$.

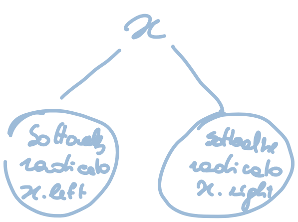
Proof: Iniziamo dimostrando il seguente claim

Ⓒ: Il sottoalbero radicato a un nodo x contiene almeno $2^{bh(x)}-1$ nodi interni.

Dimostriamo il claim Ⓒ per induzione sull'altezza $h(x)$ di x.

- base dell'induzione: $h(x)=0 \Rightarrow x$ è una foglia $\Rightarrow$ il sottoalbero radicato a x ha 0 nodi interni
  $\searrow bh(x)=0$
  $\#\text{nodi interni} = 0 \geq 2^0-1 = 1-1$. ✓
- ipotesi induttiva: Assumiamo il claim vero per qualunque nodo di altezza $< h(x)$; $h(x)>0$.
- passo induttivo: Siccome $h(x)>0 \Rightarrow x$ ha due figli.
  Se $x.child$ è nero $\Rightarrow bh(x) = bh(x.child)+1$
  Se $x.child$ è rosso $\Rightarrow bh(x) = bh(x.child)$.
  $\Rightarrow bh(x.child) \geq bh(x) - 1$.

Sottoalbero radicato a x



Inoltre, $h(x.child) < h(x)$, quindi possiamo applicare l'ipotesi induttiva a $x.child$.
Il sottoalbero radicato a x contiene i sottoalberi radicati ai suoi figli e quindi il numero di nodi interni è

$$\#\text{nodi interni all'albero radicato a } x.left + \#\text{nodi interni all'albero radicato a } x.right + 1$$

$$\geq \left(2^{bh(x.left)}-1\right)+\left(2^{bh(x.right)}-1\right)+1 \geq$$

uso l'ipotesi induttiva sui figli di x

$$\geq 2\left(2^{bh(x.child)}-1\right)+1 \geq 2\left(2^{bh(x)-1}-1\right)+1 =$$

$$= 2^{bh(x)-1+1}-2+1 = 2^{bh(x)}-1. \blacksquare$$

Adesso dimostriamo il lemma usando il claim ©:

Sia h^* l'altezza dell'intero albero.

$\boxed{\text{Goal: } h^* \leq 2\log(n+1), \text{ con } n \text{ numero di nodi interni.}}$

$h^* = h(root)$ quindi usiamo il claim © con $x = root$.

1. $n \geq 2^{bh(root)}-1$
2. inoltre, $bh(root) \geq \frac{h^*}{2}$ (almeno metà dei nodi lungo un cammino root-leaf sono neri, propr. 4)

$\Rightarrow n \geq 2^{\frac{h^*}{2}}-1 \Rightarrow 2^{\frac{h^*}{2}} \leq n+1 \Leftrightarrow \frac{h^*}{2} \leq \log(n+1) \Rightarrow h^* \leq 2\log(n+1).$ $\square$

Una conseguenza immediata di questo lemma è che le operazioni SEARCH, MINIMUM, MAXIMUM, SUCCESSOR e PREDECESSOR impiegano $O(\log n)$-TIME.

7.2. Problemi di ottimizzazione

Un problema computazionale sulle cui variabili è definita una o più funzioni di costo e tale che l'obiettivo sia trovare una soluzione che:

- soddisfa i requisiti del problema

- minimizza / massimizza una certa funzione obiettivo che è data da una combinazione delle funzioni di costo.

Una soluzione che minimizza / massimizza questa funzione obiettivo si dice soluzione ottima.

7.3 PROGRAMMAZIONE DINAMICA

CARATTERISTICHE che deve avere un problema per essere risolto correttamente con la Programmazione Dinamica

- OVERLAPPING SUBPROBLEMS: NUMERO POLINOMIALE DI SOTTOPROBLEMI CHE SI RIPETONO
- PROPRIETÀ DI SOTTOSTRUTTURA OTTIMA: Data una soluzione ottima le sue restrizioni ai sottoproblemi sono ottime per i sottoproblemi.

Proprietà del problema

7.2.1 ROD-CUTTING

Abbiamo una barra di lunghezza n e una tabella di prezzi p_i, $i \in \{1, \dots, n\}$ dei pezzi di lunghezza i.

Dobbiamo massimizzare il guadagno della vendita, ovvero il prezzo.

$$r_n = \max_{\substack{i_1, \dots, i_k \\ i_1 + \dots + i_k = n}} \left(p_{i_1} + p_{i_2} + \dots + p_{i_k} \right)$$

Possiamo pure scrivere ricorsivamente $r_n = \max\{p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \dots, r_{\lceil \frac{n}{2} \rceil} + r_{\lfloor \frac{n}{2} \rfloor}\}$

Dopo il primo taglio otteniamo due istanze che sono indipendenti:

Riformulando $\quad r_n = \max\{p_i + r_{n-i} \mid 1 \le i \le n\}$, $r_0 = 0$

Ogni istanza ha un solo sottoproblema.

Un algoritmo ricorsivo impiegherebbe $O(2^n)$-TIME

Una soluzione basata sulla programmazione dinamica impiega $O(n^2)$-TIME.

Il numero di sottoproblemi distinti è $r_1, r_2, \dots, r_{n-1}$ → polinomiale

Quindi abbiamo $O(n)$ sottoproblemi + n scelte per ognuno di essi

$$\Downarrow$$

ROD CUTTING CON PROG DINAMICA $\quad O(n^2)$-TIME

Vale la proprietà di sottostruttura ottima:

$$r_n = \max\{p_i + r_{n-i} \mid 1 \le i \le n\}$$

Fissiamo una certa i e sia r_n^* ottima

$r_n^* \ge r_n \ \forall$ soluzione r_n

Sia $r_n^* = p_i + r_{n-i}^*$ ← costruzione di r_n^*

Supponiamo che r_{n-i}^* non sia ottima ⇒ ne esiste una migliore $r'_{n-i} > r_{n-i}^*$ una allora posso definire $r'_n = p_i + r'_{n-i} > p_i + r^*_{n-i} = r^*_n$ ASSURDO

r^*_{n-i} è ottimo $\Rightarrow$ vale la proprietà di sottostruttura ottima.