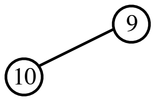
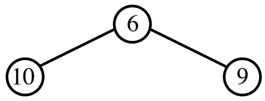
1. Si supponga di operare su di un Min-Heap inizialmente vuoto, inserendo le seguenti 13 chiavi, nell'ordine dato: $\langle 10, 9, 6, 8, 4, 11, 13, 12, 7, 5, 3, 1, 2 \rangle$. Si fornisca la configurazione (disegnare l'albero) del Min-Heap dopo ciascuna delle 13 operazioni di inserimento. Indicare infine quale sarebbe la configurazione della struttura dati dopo un'operazione di estrazione del minimo.

Inserimento della chiave 10

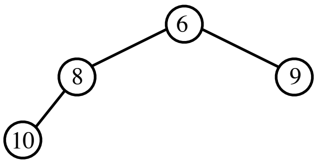
Inserimento della chiave 9



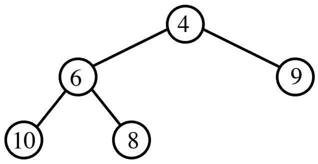
Inserimento della chiave 6



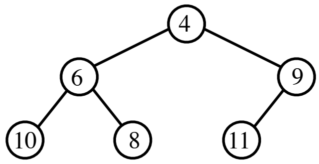
Inserimento della chiave 8



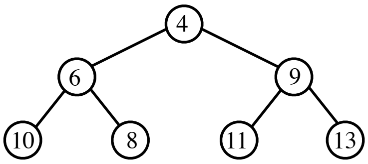
Inserimento della chiave 4



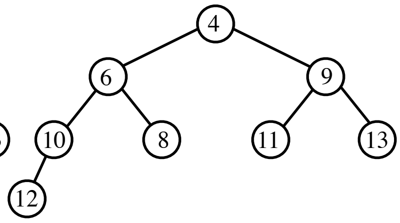
Inserimento della chiave 11



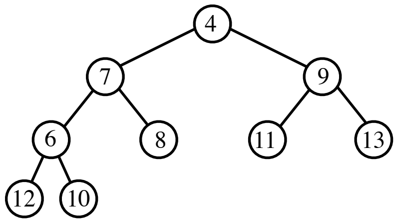
Inserimento della chiave 13



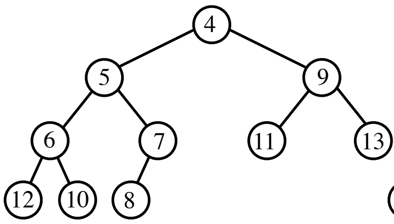
Inserimento della chiave 12



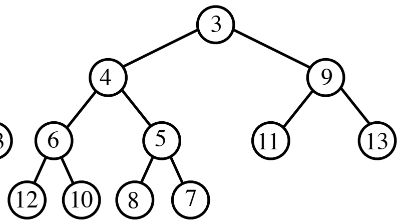
Inserimento della chiave 7



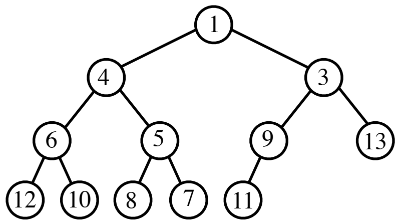
Inserimento della chiave 5



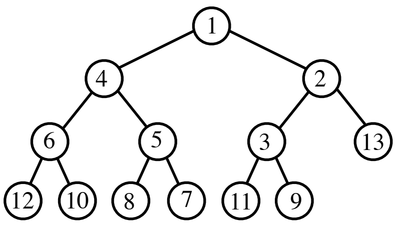
Inserimento della chiave 3



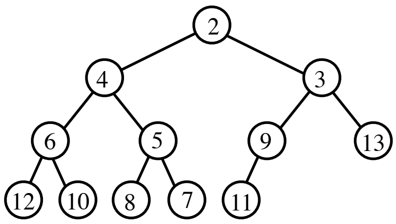
Inserimento della chiave 1



Inserimento della chiave 2

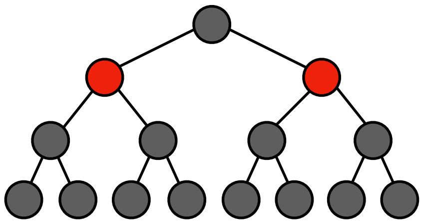


Estrazione del minimo

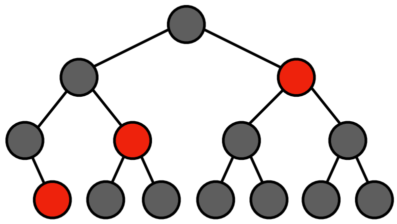


2. Si supponga di operare su di un albero Rosso-Nero completo, contenente 15 chiavi. I nodi dell'albero sono tutti nodi neri ad esclusione dei nodi del livello 1 il cui colore è rosso. Nello specifico si effettuino 6 operazioni di cancellazioni della chiave più piccola contenuta nell'albero e si fornisca la configurazione dell'albero dopo ciascuna delle 6 cancellazioni.

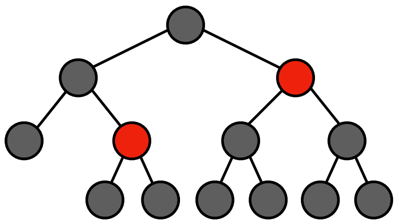
Configurazione iniziale



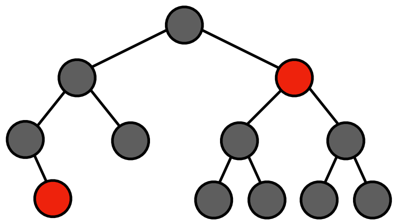
Prima cancellazione



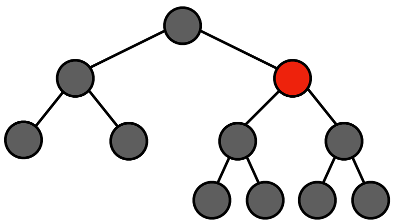
Seconda cancellazione



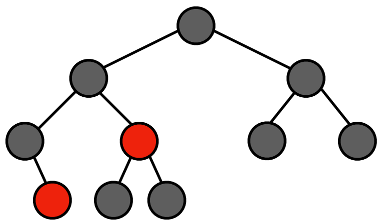
Terza cancellazione



Quarta cancellazione



Quinta cancellazione



3. Si forniscano le funzioni ricorsive utilizzate per il calcolo del costo di una soluzione ottima utilizzate dagli algoritmi di Programmazione Dinamica per i problemi della Longest Common Subsequence e della Distanza di Editing.

Funzione per il problema della longest common subsequence

$$LCS(i,j) = \begin{cases} 0 & \text{se } i = 0 \text{ o } j = 0 \\ LCS(i-1, j-1) + 1 & \text{se } i,j > 0 \text{ e } x[i] = y[j] \\ \max(LCS(i, j-1), LCS(i-1, j) & \text{se } i,j > 0 \text{ e } x[i] \neq y[j] \end{cases}$$

Funzione per il problema della distanza di editing

$$EDT(i,j) = \begin{cases} i & \text{se } j = 0 \\ j & \text{se } i = 0 \\ EDT(i-1, j-1) & \text{se } i,j > 0 \text{ e } x[i] = y[j] \\ \min(EDT(i, j-1), EDT(i-1, j), EDT(i-1, j-1)) + 1 & \text{se } i,j > 0 \text{ e } x[i] \neq y[j] \end{cases}$$

4. Si forniscano gli pseudo-codici (o i codici in linguaggio C/C++) degli algoritmi di Bellman-Ford e Dijkstra per la risoluzione dei problemi di cammino minimo da sorgente singola. Indicare anche la complessità computazionale delle procedure fornite, motivandone la risposta.

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

ESAME DI ALGORITMI (SIMULAZIONE) – RISOLUZIONE FOGLIO B

5. $$T(n) = 16\,T\left(\frac{n}{b}\right) + \Theta\left(n^2 \log^2(n)\right), \qquad b > 1.$$

(driving function) $\quad f(n) = \Theta(n^2 \log^2 n)$

(watershed function) $\quad w(n) = n^{\log_b 16}$

Il confronto tra $f(n)$ e $w(n)$ dipende dal confronto tra $n^{\log_b 16}$ e n^2. Abbiamo che
$$n^{\log_b 16} \gtrless n^2 \iff \log_b 16 \gtrless 2 \iff b \lessgtr 4.$$
Allora abbiamo che:

- $1 < b < 4 \Rightarrow \log_b 16 > 2 \Rightarrow \exists\, \varepsilon > 0$, più precisamente basta scegliere $0 < \varepsilon < \log_b 16 - 2$, tale che $f(n) = \Theta(n^2 \log^2 n) = O(n^{\log_b 16 - \varepsilon})$. In tal caso ci troviamo nel caso I del Thm Master e pertanto $T(n) = \Theta(n^{\log_b 16})$.

- $b = 4 \Rightarrow \log_b 16 = \log_4 16 = 2 \Rightarrow \exists\, k \geq 0$, in particolare $k = 2$, tale che $f(n) = \Theta(n^2 \log^2 n)$. Pertanto, ci troviamo nel caso II del Thm Master e perciò $T(n) = \Theta(n^2 \log^3 n)$.

- $b > 4 \Rightarrow \log_b 16 < 2 \Rightarrow \exists\, \varepsilon > 0$, in particolare basta scegliere $0 < \varepsilon < 2 - \log_b 16$ tale che $f(n) = \Theta(n^2 \log^2 n) = \Omega(n^{\log_b 16 + \varepsilon})$. Per rientrare nelle ipotesi del caso III del Thm Master dobbiamo verificare che valga

pure la condizione di regolarità, ovvero che $\exists\, c<1$ tale che $16 f\left(\frac{n}{b}\right) < c f(n) \iff 16\left(\frac{n}{b}\right)^2 \log^2\left(\frac{n}{b}\right) \leq c\, n^2 \log^2 n$
$\iff 16 \frac{n^2}{b^2} (\log n - \log b)^2 \leq c\, n^2 \log^2 n \iff$

$$\frac{16}{b^2} n^2 (\log^2 n - 2\log b \log n + \log^2 b) \leq c\, n^2 \log^2 n$$

$$\iff \left(\frac{16}{b^2} - c\right) n^2 \log^2 n \leq n^2 \frac{16}{b^2} \log b\, (2\log n - \log b)$$

vero asintoticamente se $\frac{16}{b^2} - c \leq 0 \Rightarrow c \geq \frac{16}{b^2}$
basta scegliere $\frac{16}{b^2} \leq c < 1$ (che esiste perché $\frac{16}{b^2} = \left(\frac{4}{b}\right)^2 < 1$)
Possiamo allora concludere che, per $b>4$,
$T(n) = \Theta(n^2 \log^2 n)$.

Ricapitolando
$1<b<4 \Rightarrow T(n) = \Theta(n^{\log_b 16})$,
$b=4 \Rightarrow T(n) = \Theta(n^2 \log^3 n)$,
$b>4 \Rightarrow T(n) = \Theta(n^2 \log^2 n)$.

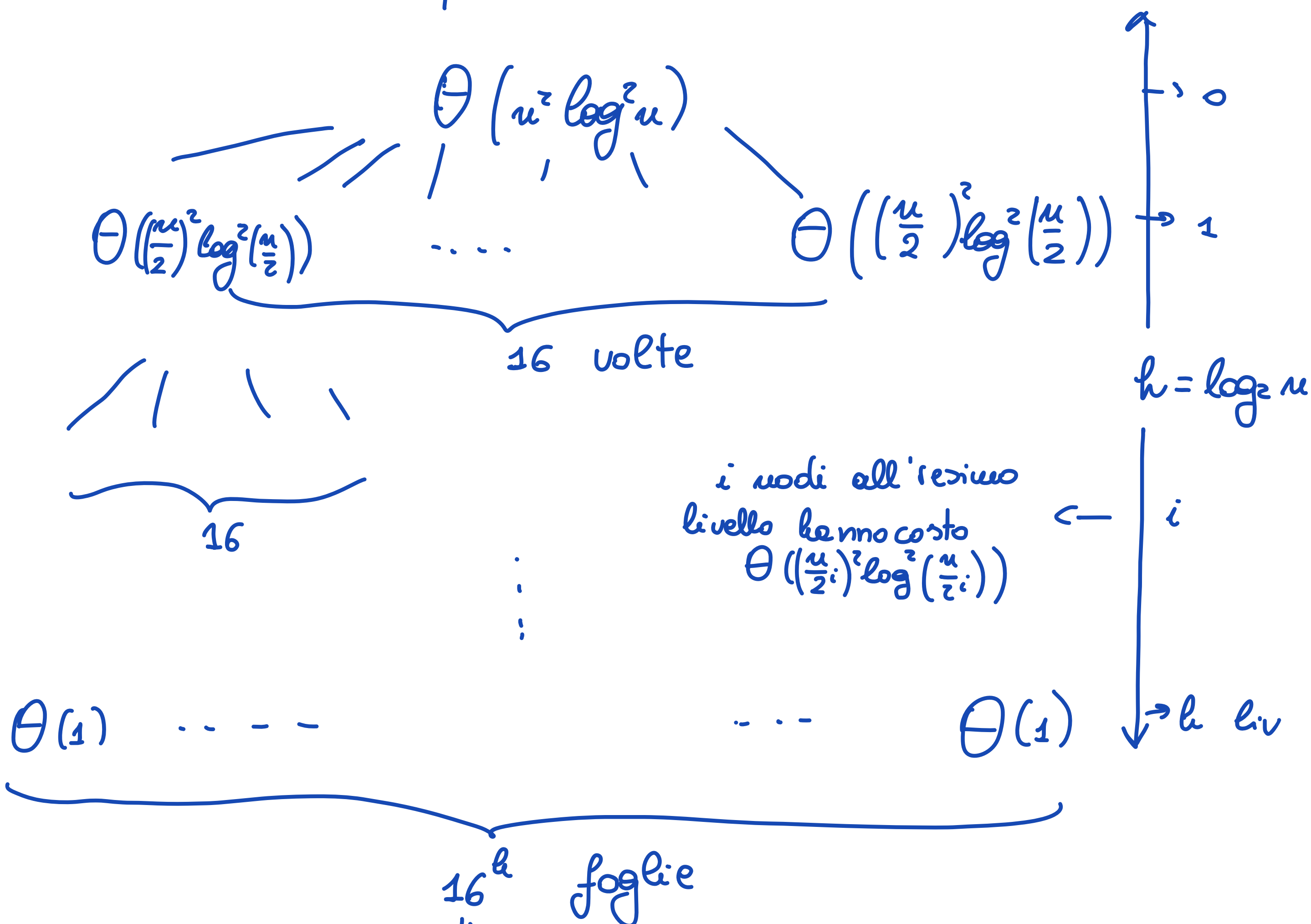
(i) $T(n) = O(n^2 \log^2 n)$ non può capitare per $1<b<4$ perché in questo caso $n^{\log_b 16}$ ha un ordine di grandezza maggiore di $n^2 \log^2 n$. Analogamente, $\Theta(n^2 \log^3 n) \neq O(n^2 \log^2 n)$. Per definizione, $T(n) = \Theta(n^2 \log^2 n) \Rightarrow T(n) = O(n^2 \log^2 n)$. Quindi (i) è vera per $b>4$.

(ii) $T(n) = o(n^2 \log^3(n))$. Per $1<b<4$, $n^{\log_b 16}$ ha un ordine di grandezza maggiore di $n^2 \log^3(n)$, quindi $\Theta(n^{\log_b 16}) \neq o(n^2 \log^3 n)$. Per $b=4$, $\Theta(n^2 \log^3 n) \neq o(n^2 \log^3 n)$ perché il bound è preciso.

Infine, per $b>4$, $T(n) = \Theta(n^2 \log^2 n) = o(n^2 \log^3 n)$, quindi (ii) è vera per $b>4$.

(iii) $T(n) = \Theta(n^4)$. Per $1<b<4$, $\Theta(n^{\log_b 16}) = \Theta(n^4)$ per $b=2$. Per $b=4$, $\Theta(n^2 \log^3 n) \neq \Theta(n^4)$ e, analogamente, per $b>4$, $\Theta(n^2 \log^2 n) \neq \Theta(n^4)$. Quindi (iii) è vera per $b=2$.

Albero di ricorsione per $b=2$.



$$16^h = 16^{\log_2 n} = 2^{4 \cdot \log_2 n} = \left(2^{\log_2 n}\right)^4 = n^4$$

6. (0-1)-KNAPSACK ha la proprietà di sottostrutture ottime.
Una soluzione è una tupla $(x_1, \dots, x_n)$ con $x_i \in \{0,1\}$, dove $x_i = 1$ se metto nello zaino i e $x_i = 0$ altrimenti.
Sia $(x_1^*, \dots, x_n^*)$ una soluzione ottima, allora $\sum_{i=1}^{n} x_i^* w_i \leq W$ e $\sum_{i=1}^{n} x_i^* v_i$ è massimo. Consideriamo il sottoproblema in cui bisogna trovare un assegnamento ottimo per $\{x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n\}$ per qualche $j \in \{1, \dots, n\}$.
Osserviamo che $\sum_{\substack{i=1 \\ i\neq j}}^{n} x_i^* w_i \leq W - x_j^* w_j =: W'$. Se $(x_1^*, \dots, x_{j-1}^*, x_{j+1}^*, \dots, x_n^*)$ non fosse ottima allora dovrebbe esistere $(x_1', \dots, x_{j-1}', x_{j+1}', \dots, x_n')$ tale che

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_i' w_i \leq W - x_j^* w_j \quad (a) \quad \text{e}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_i' v_i > \sum_{\substack{i=1 \\ i\neq j}}^{n} x_i^* v_i \quad (b).$$

Consideriamo allora $(x_1', \dots, x_{j-1}', x_j^*, x_{j+1}', \dots, x_n)$, abbiamo che

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_i' w_i + x_j^* w_j \leq W \quad (\text{per } (a)) \quad \text{e} \quad \sum_{\substack{i=1 \\ i\neq j}}^{n} x_i' v_i + x_j^* v_j > \sum_{i=1}^{n} x_i^* v_i \ (\text{per } (b))$$

e quindi esisterebbe una soluzione globale migliore di quella ottima, che è una contraddizione.

0-1-KNAPSACK non gode della proprietà di scelta greedy.
Supponiamo di avere 3 oggetti. Il primo oggetto ha valore 60 € e peso 10 kg, il secondo oggetto ha valore 100 € e peso 20 kg e il terzo oggetto ha valore 120 € e peso 30 kg. Il peso limite è 50 kg.
Le possibilità sono:

- item 1 + item 2 che ha valore 160 € e peso 30 kg
- item 1 + item 3 che ha valore 180 € e peso 40 kg
- item 2 + item 3 che ha valore 220 € e peso 50 kg.

Chiaramente la soluzione ottima è item 2 + item 3, ovvero $x_1 = 0$, $x_2 = x_3 = 1$.

I valori per unità di peso sono 6 €/kg per item 1, 5 €/kg per item 2 e 4 €/kg per item 3.

Una strategia greedy selezionerebbe l'item 1 alla prima iterazione, e l'item 2 alla seconda, producendo una soluzione che non è quella ottima. Questo mostra che 0-1 KNAPSACK non ha la proprietà di scelta greedy.