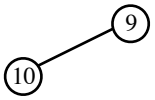
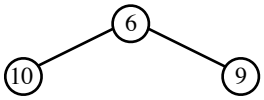
1. Si supponga di operare su di un Min-Heap inizialmente vuoto, inserendo le seguenti 13 chiavi, nell'ordine dato: $\langle 10, 9, 6, 8, 4, 11, 13, 12, 7, 5, 3, 1, 2\rangle$. Si fornisca la configurazione (disegnare l'albero) del Min-Heap dopo ciascuna delle 13 operazioni di inserimento. Indicare infine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

Inserimento della chiave 10

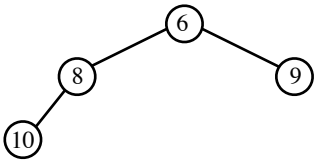
Inserimento della chiave 9



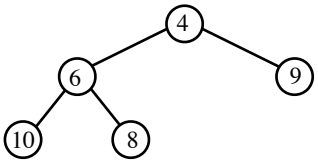
Inserimento della chiave 6



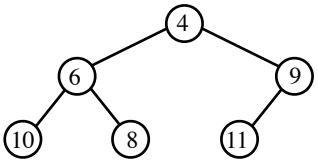
Inserimento della chiave 8



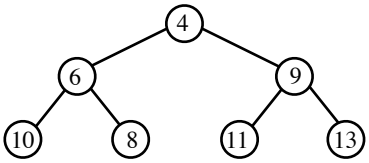
Inserimento della chiave 4



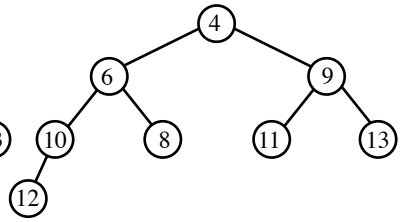
Inserimento della chiave 11



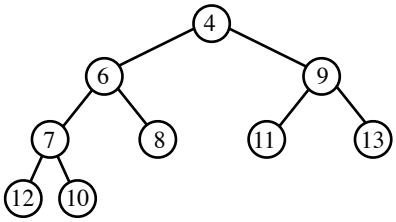
Inserimento della chiave 13



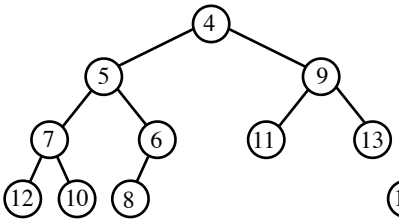
Inserimento della chiave 12



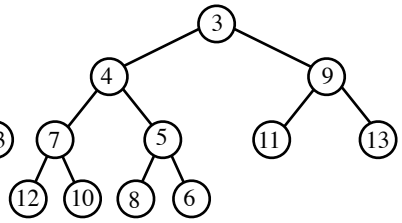
Inserimento della chiave 7



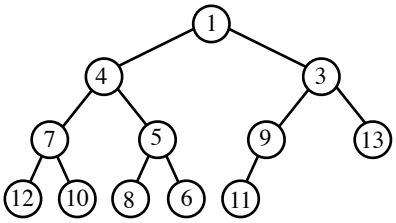
Inserimento della chiave 5



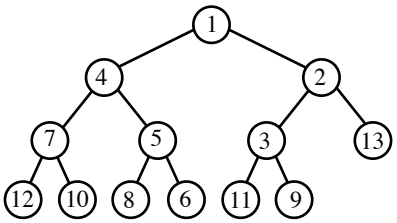
Inserimento della chiave 3



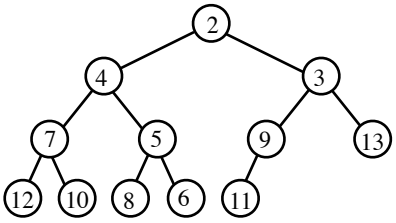
Inserimento della chiave 1



Inserimento della chiave 2

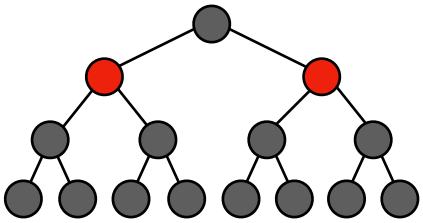


Estrazione del minimo

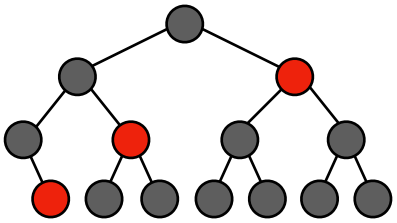


2. Si supponga di operare su di un albero Rosso-Nero completo, contenente 15 chiavi. I nodi dell'albero sono tutti nodi neri ad esclusione dei nodi del livello 1 il cui colore è rosso. Nello specifico si effettuino 6 operazioni di cancellazioni della chiave più piccola contenuta nell'albero e si fornisca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

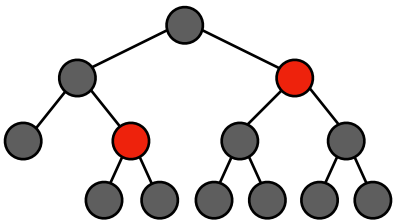
Configurazione iniziale



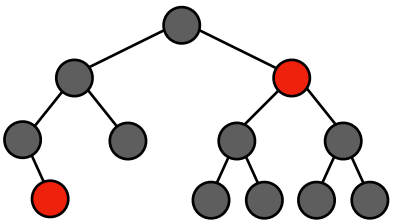
Prima cancellazione



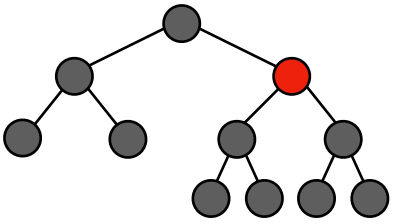
Seconda cancellazione



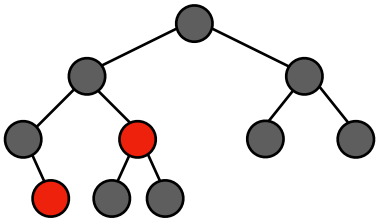
Terza cancellazione



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)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if v.d > u.d + w(u, v)
7            return FALSE
8  return TRUE
```

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = ∅
4  for each vertex u ∈ G.V
5       INSERT(Q, u)
6  while Q ≠ ∅
7       u = EXTRACT-MIN(Q)
8       S = S ∪ {u}
9       for each vertex v in G.Adj[u]
10           RELAX(u, v, w)
11           if the call of RELAX decreased v.d
12                DECREASE-KEY(Q, v, v.d)
```