# PRIORITY QUEUE - MFSET

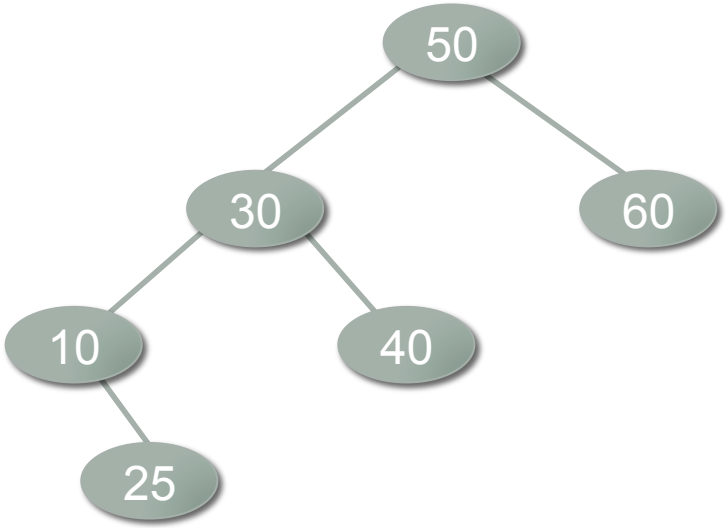
Angelo Di Iorio

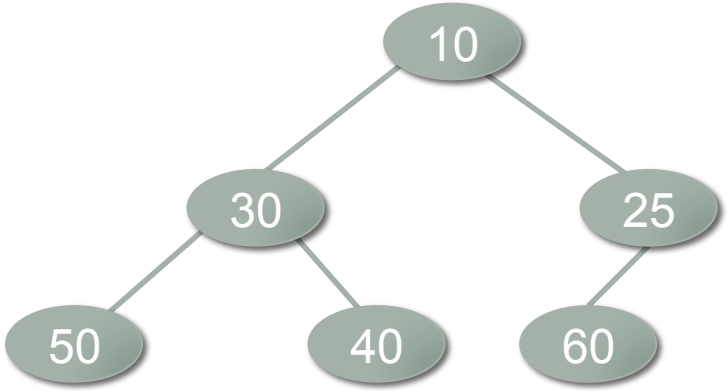
Università di Bologna

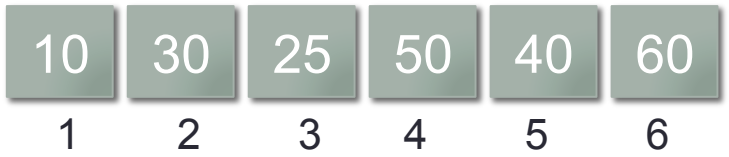
# Esercizio 1

- Implementare una classe Java per gestire un *(binary) heap* generico di elementi
- Metodi:
  - `public boolean insert(T element);`
  - `public T min();`
  - `public T deleteMin();`
  - `private void minHeapRestore(Integer i);`
- Aggiungere due metodi utility per stampare il contenuto della struttura dati come (1) vettore e (2) albero binario

# BST vs (min-)Heap



# Esercizio 2

- In uno *heap ternario* i nodi interni possono avere fino a tre figli.

- Domande:
  1. E' possibile rappresentare uno heap ternario in un vettore?
  2. Si scriva la funzione `deleteMin()` per gli heap ternari. Cosa cambia rispetto agli heap binari?
  3. Si scriva la funzione `insert()` per gli heap ternari. Cosa cambia rispetto agli heap binari?

# Esercizio 3

- Usare la classe `PriorityQueue` built-in Java per implementare un semplice task manager
- La classe `TaskManager` permette di:
  - aggiungere un task con una data priorità
  - estrarre(ed eliminare dalla coda) il task con priorità più alta
  - verificare se non ci sono più task da eseguire

- Aggiungere un metodo `toString()` per stampare i task in ordine crescente di priorità

- I task sono modellati con una classe `Task` che memorizza `nome` e `proritàà` del task

# Esercizio 3b

- Modificare la classe `TaskManager` in modo che sia possibile usare diversi criteri di priorità, selezionabili dal costruttore di `TaskManager`:
- Esempi
  - valore dell'attributo priorità
  - nome del task (inutile in pratica!)
  - durata prevista del task (da aggiungere alla classe `Task`)
  - …

# Esercizio 4 - MFSet

- Implementare una classe Java per gestire una struttura dati MFSet
- Per semplicità: la struttura MFSet è costruita a partire dall'insieme degli interi [`0, 1, 2, ..., n-1`]
- Interfaccia:
  - `public MFSet(Integer n); //costruttore`
  - `public Integer find(Integer n);`
  - `public void merge(Integer x, Integer y);`
  - `public void print();  //utility`

- Realizzazione basata su foresta, euristica del rango e compressione dei percorsi