Corso di laurea in Informatica
Introduzione agli Algoritmi

Strutture dati fondamentali: Alberi

Angelo Monti

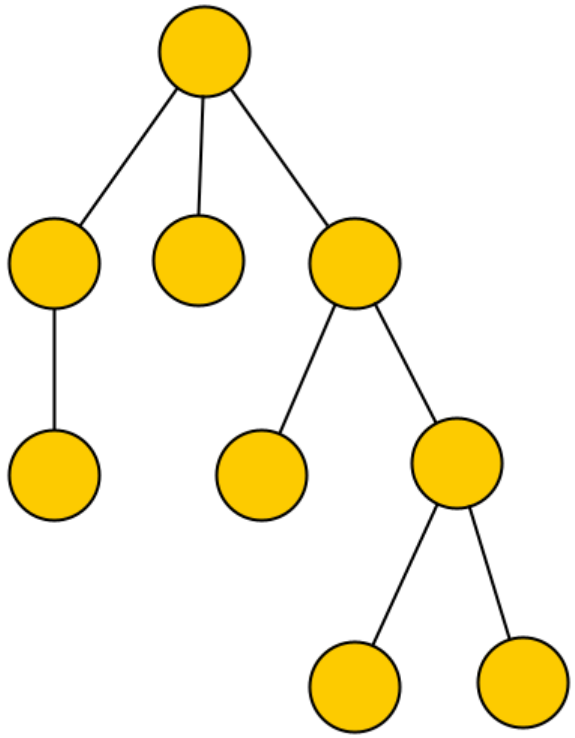
SAPIENZA
UNIVERSITÀ DI ROMA

Alberi

- L'***albero*** è una struttura dati estremamente versatile, utile per modellare una grande quantità di situazioni reali e progettare le relative soluzioni algoritmiche.
- Abbiamo già incontrato la struttura ad albero (in particolare ad albero binario) varie volte, ma l'abbiamo sempre considerata in modo intuitivo.

Alberi radicati

Un **albero radicato** è una struttura dati composta da nodi organizzati in modo gerarchico. Ha un nodo speciale, chiamato **radice**, che rappresenta il punto di partenza dell'albero e non ha genitori. Ogni altro nodo dell'albero ha esattamente un nodo genitore e può avere zero o più nodi figli. Un nodo che non ha figli è detto **foglia** e rappresenta un punto terminale dell'albero.



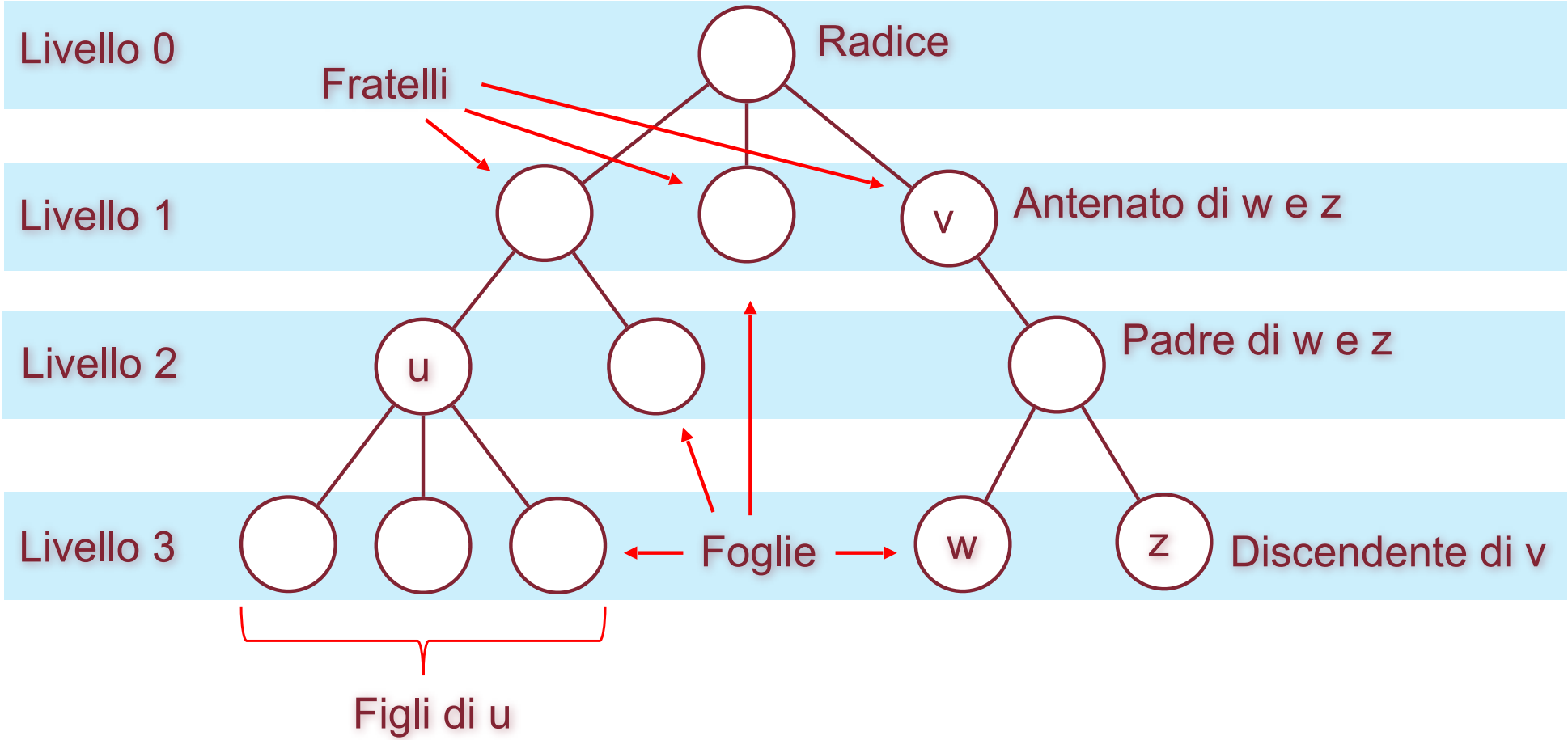
Terminologia

Consideriamo un albero radicato e sia x un suo nodo. Osserviamo ora l'unico cammino che parte dalla radice e raggiunge x. Possiamo introdurre i seguenti termini:

- Ogni nodo che si trova sul cammino è un **antenato** di x.
- Il nodo che precede x sul cammino è detto **padre** di x (l'unico nodo senza padre è la radice).
- I nodi che hanno lo stesso padre di x sono detti **fratelli** di x.
- Tutti i nodi per cui x è un antenato sono detti **discendenti** di x.

I nodi dell'albero sono organizzati in **livelli**, numerati in ordine crescente a partire dalla radice (che si trova al livello zero). L'**altezza** di un albero è la lunghezza del cammino più lungo dalla radice a una foglia. Pertanto, un albero di altezza h avrà $h + 1$ livelli.
Per convenzione all'albero vuoto, ovvero privo di nodi, si assegna altezza -1.

Alberi radicati (esempio con albero di altezza $h = 3$)

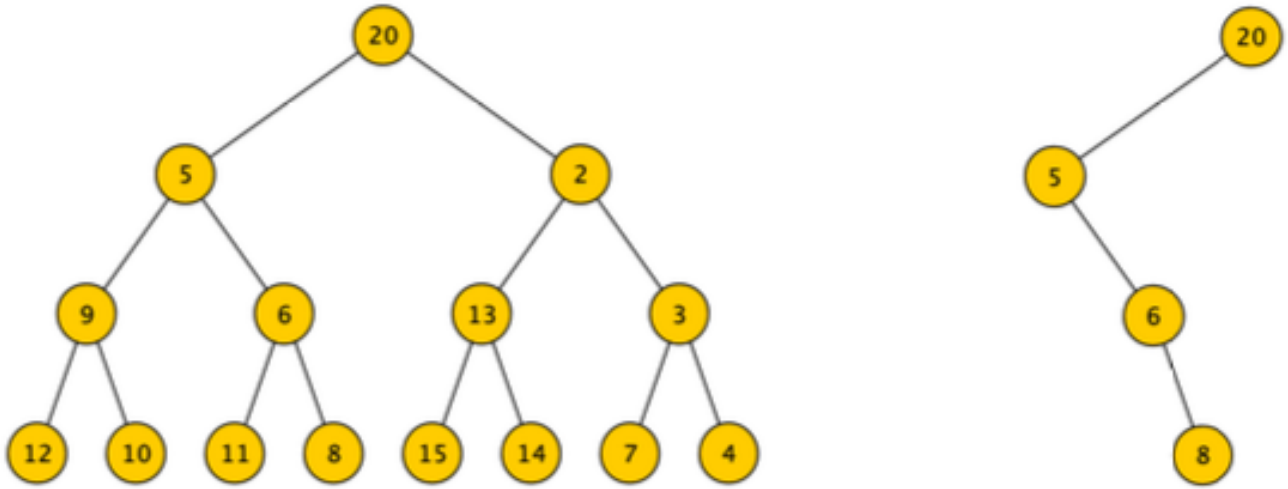


Alberi Binari

Un **albero binario** è un albero radicato in cui ogni nodo può avere al massimo due figli, denominati **figlio sinistro** e **figlio destro**. Nel seguito di questa dispensa, ci concentreremo esclusivamente sugli alberi binari.

- un albero binario nel quale tutti i livelli contengono il massimo numero possibile di nodi è chiamato **albero binario completo**
- un albero binario nel quale ogni livello contiene il minimo numero di nodi (vale a dire un nodo) è chiamato **albero binario degenere**

La Figura riporta esempi delle due tipologie di alberi:



Analizziamo ora il legame tra il numero di nodi n di un albero binario e la sua altezza h:

- **Albero di altezza h con più nodi**. Un albero binario di altezza h contiene il numero massimo possibile di nodi quando tutti i livelli dell'albero sono completamente pieni, ovvero quando l'albero è completo, In questo caso il numero di nodi è dato da

$$n = \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

  Da questa relazione, con semplici passaggi algebrici si ottiene

$$h = \log_2 \left(\frac{n+1}{2} \right)$$

- **Albero di altezza h con meno nodi**: Un albero binario di altezza h contiene il numero minimo possibile di nodi quando ciascun livello dell'albero contiene un unico nodo, ovvero quando l' albero è degenere. In questo caso, l'altezza coincide sostanzialmente con il numero di nodi:

$$h = n - 1$$

Combinando i due risultati, otteniamo che l'altezza di un albero binario soddisfa la relazione: $\log_2 \left(\frac{n+1}{2} \right) \leq h \leq n - 1$.
Da cui

$$\Omega(\log n) \leq h \leq O(n)$$

L'altezza rappresenta il numero massimo di passi necessari per raggiungere un qualunque nodo a partire dalla radice. Per ottimizzare questa quantità, una buona proprietà dell'albero binario è avere $h = \Theta(\log n)$. Un albero binario con questa caratteristica prende il nome di **albero bilanciato**.
Esempio di albero bilanciato è *l'albero binario completo o quasi completo* dove tutti i livelli tranne l'ultimo contengono il massimo numero possibile di nodi mentre l'ultimo livello è riempito completamente da sinistra verso destra solo fino ad un certo punto.

Rappresentazione di alberi binari

Gli alberi binari possono essere rappresentati in vari modi, ciascuno con vantaggi e svantaggi in termini di efficienza e semplicità. Le tre principali rappresentazioni sono:

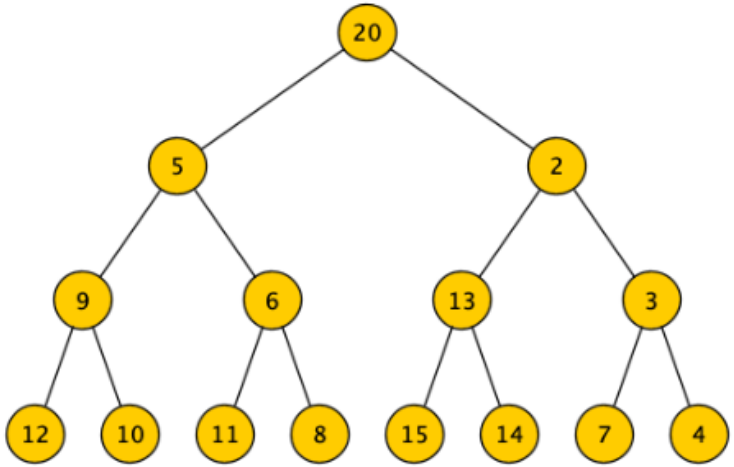
- tramite array,
- tramite il vettore dei padri,
- tramite puntatori.

Di seguito vengono esplorate queste tre modalità.

Rappresentazione tramite array

La principale caratteristica di questa rappresentazione è che, dato un nodo in posizione i, i suoi figli e i padri possono essere trovati facilmente tramite la seguente relazione:

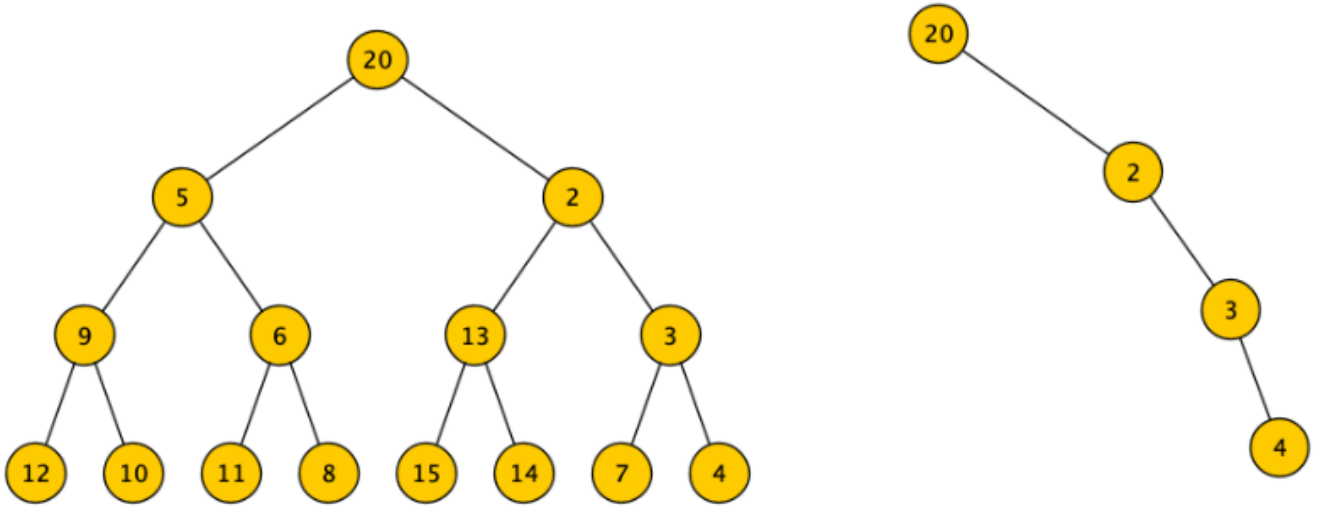
- Il **figlio sinistro**, se esiste, si trova alla posizione $2i + 1$.
- Il **figlio destro**, se esiste, si trova alla posizione $2i + 2$.
- Il **genitore**, se esiste, si trova alla posizione $\left\lfloor \frac{i-1}{2} \right\rfloor$.



La rappresentazione tramite array dell'albero è
$[20, 5, 2, 9, 6, 13, 3, 12, 10, 11, 8, 15, 14, 7, 4]$.

Svantaggi:

- Non adatta per alberi sbilanciati, poiché potrebbe esserci un ampio sp di memoria. Per rappresentare un albero con n nodi potrebbe essere essario un array di dimensioni $\Theta(2^n)$.



La rappresentazione tramite array dell'albero bilanciato a sinistra è

$$[20, 5, 2, 9, 6, 13, 3, 12, 10, 11, 8, 15, 14, 7, 4].$$

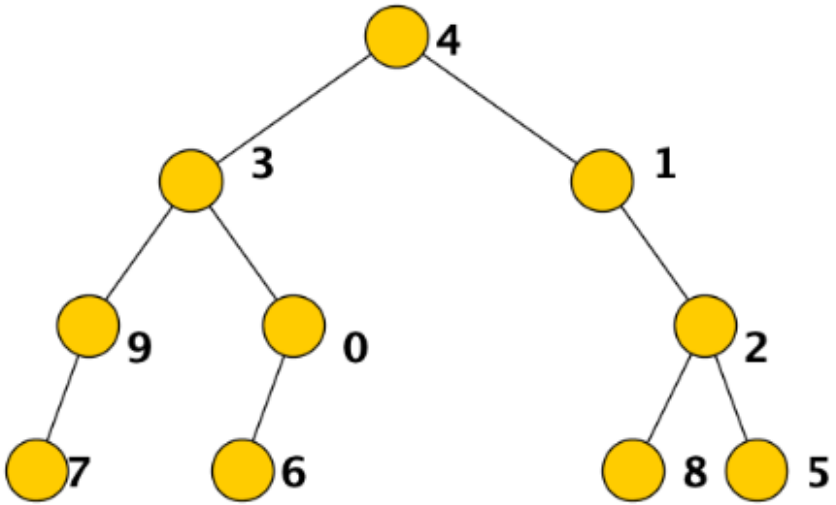
Quella dell'albero degenere a destra è

$$[20, *, 2, *, *, *, 3, *, *, *, *, *, *, *, 4].$$

Nonostante la notevole differenza nel numero di nodi, lo spazio di memoria occupato dai due alberi è lo stesso.

Rappresentazione tramite vettore dei padri

In questa rappresentazione ad ogni nodo dell'albero è associato l' indice di un vettore, e ogni elemento del vettore memorizza l'indice del nodo padre corrispondente.
Immaginiamo di avere un albero binario con n nodi. Per rappresentare l'albero tramite vettore dei padri, si utilizza un array di dimensione n in cui ogni posizione i contiene l'indice del nodo padre del nodo. La radice dell'albero è l'unico nodo che non ha un padre, l'assenza del padre viene rappresentata con un valore speciale (ad esempio, -1).



In figura è mostrato un albero binario con 10 nodi e accanto ad ogni nodo è indicato l'indice (da 0 a 9) assegnato al nodo. Il vettore dei padri di quest'albero è $[3, 4, 1, 4, -1, 2, 0, 9, 2, 3]$.

Vantaggi:

- La rappresentazione è semplice da implementare e intuitiva. Ogni nodo è identificato dal suo indice nel vettore, e il padre di ciascun nodo è facilmente accessibile.

- A differenza della rappresentazione tramite array, in questo caso la memoria utilizzata è sempre $\Theta(n)$, quindi proporzionale al numero di nodi.

Svantaggi:

- Un limite della rappresentazione è che operazioni come la ricerca dei nodi figli o l'analisi della struttura dell'albero (ad esempio il controllo del bilanciamento o il calcolo dell'altezza) risultano meno efficienti rispetto alle rappresentazioni alternative, come quella basata su puntatori.

- Questa rappresentazione non consente di distinguere direttamente tra figlio sinistro e figlio destro. Ogni nodo memorizza esclusivamente l'indice del proprio nodo padre, senza fornire informazioni sulla posizione relativa del nodo (se sinistra o destra) rispetto al genitore.

Rappresentazione tramite puntatori

E' la rappresentazione dell'albero in cui i nodi sono strutture di dati che contengono riferimenti (puntatori) ai figli. Un nodo tipico di un albero albero binario avrà almeno tre campi:

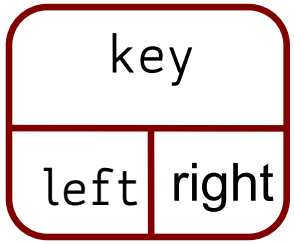
- Un campo per il **dato** del nodo.
- Un campo per il **puntatore al figlio sinistro**.
- Un campo per il **puntatore al figlio destro**.

Ogni singolo nodo è costituito da un record contenente:

key: le opportune informazioni pertinenti al nodo stesso;

left: il puntatore al figlio sinistro (oppure *None* se il nodo non ha figlio sinistro);

right: il puntatore al figlio destro (oppure *None* se il nodo non ha figlio destro);



All'albero si accede grazie al puntatore alla radice.

Vantaggi:

- Maggiore flessibilità, poiché non è necessario conoscere in anticipo la dimensione dell'albero.
- Inserimenti ed eliminazioni di nodi sono più efficienti rispetto alla rappresentazione con array.

Svantaggi:

- La gestione della memoria è più complessa (richiede allocazione dinamica).
- Ogni nodo occupa più memoria a causa dei puntatori, rispetto alle rappresentazioni tramite array.

Alberi radicati con record e puntatori in python

Ogni nodo dell'albero viene definito come una classe Python con riferimenti ai figli sinistro e destro.
Si consideri ad esempio la classe NodoAB che rappresenta un nodo dell'albero binario:

```
class NodoAB:
    def __init__(self,   key = None,   left = None,  right =
        None):
         self.key = valore
         self.left = left
         self.right = right
```

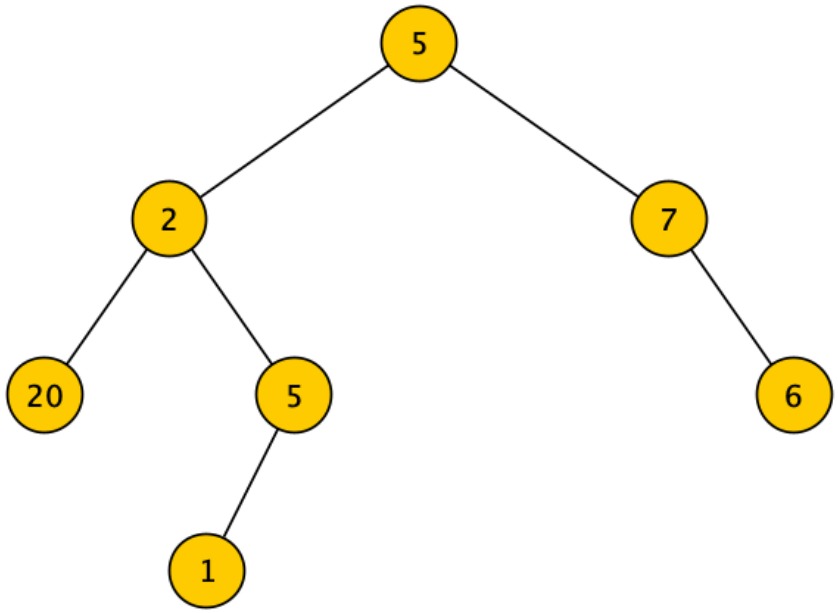
Ogni nodo contiene un valore e due riferimenti, uno al figlio sinistro e uno al figlio destro:

Per costruire un albero binario, possiamo creare nodi e collegarli tra loro come segue:

Per costruire un albero binario, possiamo creare nodi e collegarli tra loro come segue:

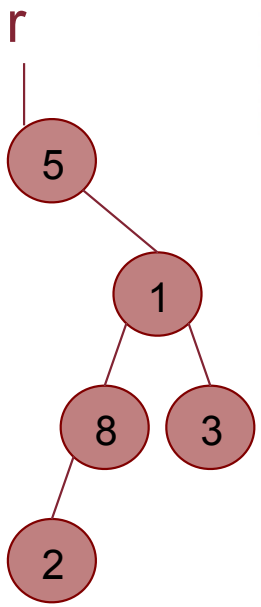
```
# Creazione dei nodi
radice = NodoAB(5)
radice.left = NodoAB(2)
radice.right = NodoAB(7)
radice.left.right = NodoAB(5)
radice.right.right = NodoAB(6)
radice.left.right.left = NodoAB(1)
radice.left.left = NodoAB(20)
```

L'albero risultante avrà la seguente struttura:



ESERCIZIO: scrivere una funzione *stampa*(*r*) che stampa le chiavi dei nodi dell'albero r. Stampa ricorsivamente in base a questa regole: Prima la chiave del nodo e poi indentate le chiavi del sottoalbero di sinistra e quelle del sottoalbero di destra.

```
def stampa(p, h = 0):
    if p == None:
        print('| '* h, '-')
    else:
        print('| '* h, p.key)
        stampa(p.left, h+1)
        stampa(p.right, h+1)
```




```
>>> stampa(r)
5
| -
| 1
| | 8
| | | 2
| | | | -
| | | | -
| | | -
| | 3
| | | -
| | | -
```

ESERCIZIO: scrivere una funzione $generaAlbero(n, m)$ che genera un albero a caso con n nodi aventi chiavi casuali nell'intervallo $[1, \ldots, m]$

```
def generaAlbero(n, m):
    ''' restituisce il puntatore alla testa di un albero binario
    di n nodi le cui chiavi sono interi tra 0 e m'''
    if n == 0:
        return None
    p = NodoAB(random.randint(1, m))
    s = random.randint(0, n-1)
    p.left = generaAlbero(s, m)
    p.right = generaAlbero(n - 1 - s, m)
    return p
```

```
>>> r = generaAlbero(10, 100)
```

Corso di laurea in Informatica
Introduzione agli Algoritmi

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

Esercizi

- Dire quant'è la massima lunghezza che può richiedere una lista per poter memorizzare un albero binario di n nodi con la rappresentazione posizionale.

- Progettare un algoritmo che, dato un albero binario memorizzato tramite vettore dei padri, restituisca il vettore relativo alla rappresentazione posizionale dello stesso albero. Calcolare il costo computazionale dell'algoritmo.

- Progettare un algoritmo che, dato un albero binario memorizzato tramite rappresentazione posizionale, restituisca il vettore dei padri dello stesso albero. Calcolare il costo computazionale dell'algoritmo.