

Laboratorio I

a.a. 2025/2026

Alberi binari di ricerca e grafi

Contenuti

- Alberi binari di ricerca
 - Definizione
 - Operazioni
- Grafi
 - Rappresentazione tramite liste di adiacenza
 - Rappresentazione tramite matrice di adiacenza
 - Esercizi
- Esercizi extra

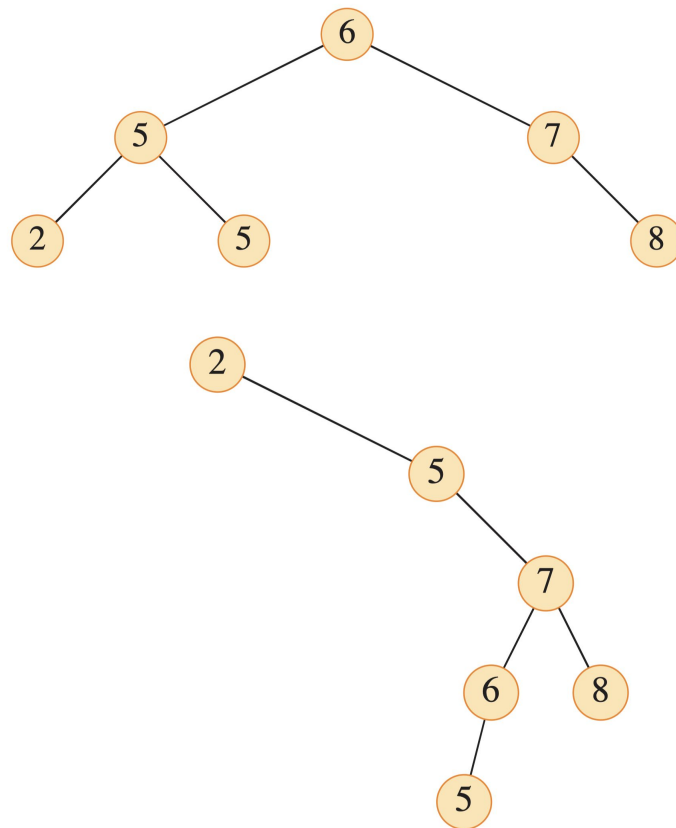
Alberi binari di ricerca (ABR)

Albero binario in cui ogni nodo x soddisfa le seguenti proprietà:

1. Ogni nodo nel sottoalbero **sinistro** di x ha un valore $\leq x.val$
2. Ogni nodo nel sottoalbero **destro** di x ha un valore $\geq x.val$

I valori nei nodi vengono anche detti **chiavi**

Esempi di ABR:



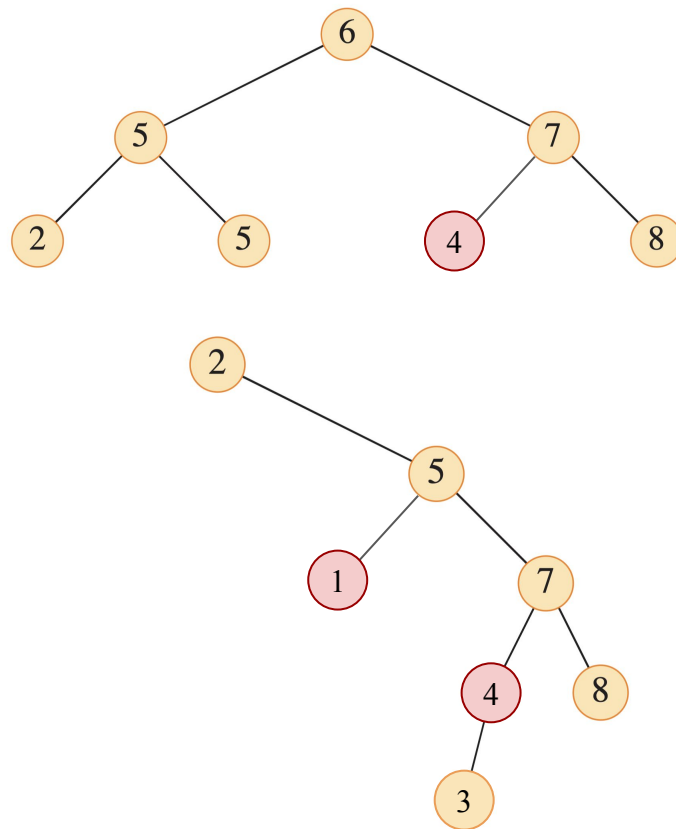
Alberi binari di ricerca (ABR)

Albero binario in cui ogni nodo x soddisfa le seguenti proprietà:

1. Ogni nodo nel sottoalbero **sinistro** di x ha un valore $\leq x.val$
2. Ogni nodo nel sottoalbero **destro** di x ha un valore $\geq x.val$

I valori nei nodi vengono anche detti **chiavi**

Esempi di non ABR:



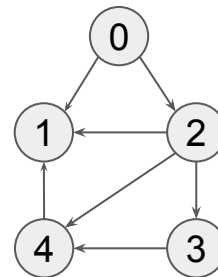
Operazioni su alberi binari di ricerca

- `abrStampaCrescente(t)`: Stampa i valori di `t` in ordine crescente
- `abrStampaDecrescente(t)`: Stampa i valori di `t` ordine decrescente
- `abrArray(t)`: Restituisce un array ordinato con i valori in `t`
- `abrMassimo(t)`: Restituisce il valore massimo in `t`
- `abrCerca(t, v)`: Restituisce un nodo di `t` che ha valore uguale a `v`,
oppure `null` se tale nodo non esiste
- `abrInserisci(t, v)`: Inserisce in `t` un nodo con un valore `v`
- `abrVerifica(t)`: Restituisce `true` se e solo se `t` è un ABR

Grafi

Un grafo G è una coppia (N, E) dove

- N è un insieme di nodi
- E è un insieme di archi, $E \subseteq N \times N$
 - Nei grafi *non* orientati vale $(u,v) \in E \Rightarrow (v,u) \in E$



Due rappresentazioni classiche:

Liste di adiacenza

$v \in \text{Lista}(u)$
se $(u,v) \in E$

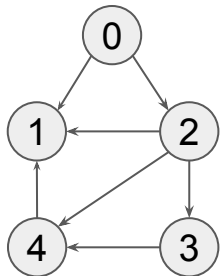
- $0 \rightarrow 1, 2$
- $1 \rightarrow /$
- $2 \rightarrow 1, 3, 4$
- $3 \rightarrow 4$
- $4 \rightarrow 1$

Matrice di adiacenza

$M[i,j]=1$ se $(i,j) \in E$,
0 altrimenti

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	0	0	0
2	0	1	0	1	1
3	0	1	0	0	1
4	0	1	0	0	0

Grafi in JavaScript



Liste di adiacenza

0 → 1, 2

1 → /

2 → 1, 3, 4

3 → 4

4 → 1

Matrice di adiacenza

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	0	0	0
2	0	1	0	1	1
3	0	0	0	0	1
4	0	1	0	0	0

```
let g1 = {  
  0: [1, 2],  
  1: [],  
  2: [1, 3, 4],  
  3: [4],  
  4: [1]  
}
```

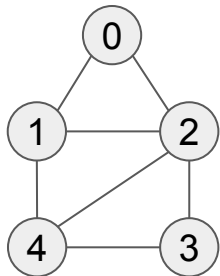
← Gli identificatori dei nodi possono anche essere stringhe

```
let gm = [  
  [0, 1, 1, 0, 0],  
  [0, 0, 0, 0, 0],  
  [0, 1, 0, 1, 1],  
  [0, 0, 0, 0, 1],  
  [0, 1, 0, 0, 0]  
]
```

Anche qui, definendo oggetti per mappare stringa a indice e viceversa:

```
let str2idx = { "A": 0, "B": 1, "C": 2,  
  "D": 3, "F": 4}  
let idx2str = { 0: "A", 1: "B", 2: "C",  
  3: "D", 4: "F"}  
// Esempio accesso:  
gm[str2idx["A"]][str2idx["D"]]  
// Equivale a:  
gm[0][3]
```

Grafi non orientati in JavaScript



Liste di adiacenza

0 → 1, 2
1 → 0, 2, 4
2 → 0, 1, 3, 4
3 → 2, 4
4 → 1, 2, 3

```
let g1 = {  
  0: [1, 2],  
  1: [0, 2, 4],  
  2: [0, 1, 3, 4],  
  3: [2, 4],  
  4: [1, 2, 3]  
}
```

← Per ogni arco (i, j) , il nodo j appare nella lista di i e viceversa

Matrice di adiacenza

	0	1	2	3	4
0	0	1	1	0	0
1	1	0	1	0	1
2	1	1	0	1	1
3	0	0	1	0	1
4	0	1	1	1	0

```
let gm = [  
  [0, 1, 1, 0, 0],  
  [1, 0, 1, 0, 1],  
  [1, 1, 0, 1, 1],  
  [0, 0, 1, 0, 1],  
  [0]  
]
```

← La matrice è simmetrica quindi possiamo risparmiare memoria eliminando gli elementi sotto la diagonale, ma negli accessi $gm[i][j]$ dobbiamo garantire $i \leq j$

Esercizi su grafi orientati

Per entrambi i tipi di rappresentazione, implementare:

- `bidirezionale(g, i, j)`: Restituisce true se e solo se il grafo `g` ha sia un arco `(i,j)` che un arco `(j,i)`
- `grado(g, n)`: Restituisce il grado del nodo `n`, cioè il numero di archi entranti e uscenti
- `trasponi(g)`: Restituisce un nuovo grafo ottenuto invertendo la direzione di ogni arco presente in `g`

Esercizi extra

ABR:

- `abrContaMinoriUguali(t, v)`: Conta quanti valori $\leq v$ contiene l'ABR `t`
- `abrDaArray(a)`: Costruisce un ABR a partire da un array *ordinato* `a`
- `abrSommaIntervallo(t, min, max)`: Somma i valori nell'ABR `t` che sono compresi nell'intervallo `[min, max]`

Per entrambi i tipi di rappresentazione di un grafo, implementare:

- `nodiSorgente(g)`: Restituisce un array di tutti i nodi con grado entrante 0
- `distribuzioneGradi(g)`: Restituisce un multinsieme che rappresenta la distribuzione dei gradi dei nodi in `g`, cioè un oggetto dove le chiavi rappresentano i gradi e i valori rappresentano il numero di nodi con quel grado