Terzo allenamento

Olimpiadi Italiane di Informatica - Selezione territoriale

Luca Chiodini
luca@chiodini.org

30 marzo 2019

Programma

1. Lettura e analisi di un problema
2. Spiegazione teorica
3. Soluzione

Lettura e analisi di un problema

“Ponti e isole” (seconda gara OIS 2015)

“Ponti e isole” (seconda gara OIS 2015)

A seguito di un violento maremoto alcuni dei ponti che collegano le N isole dell’arcipelago Nowhere sono stati distrutti e il governo deve correre ai ripari per non lasciare che alcune isolette rimangano isolate e irraggiungibili.
Il governo dell’arcipelago Nowhere ha quindi assunto Giorgio per determinare quale è il minimo numero di ponti che è necessario costruire in aggiunta agli M rimasti affinché l’arcipelago sia di nuovo connesso, ovvero sia possibile da ogni isola raggiungere tutte le altre isole. Aiuta Giorgio a svolgere il suo compito!

“Ponti e isole” (seconda gara OIS 2015)

Il file `input.txt` è composto da $M + 1$ righe. La prima riga contiene i due interi N e M. Le successive M righe contengono due interi ciascuna, gli indici `da[`i`]`, `a[`i`]` delle isole collegate dall’i-esimo ponte.

`input.txt`	`output.txt`

`4 2`
`1 3`
`3 2`
	`1`

“Ponti e isole” (seconda gara OIS 2015)

Il file `input.txt` è composto da $M + 1$ righe. La prima riga contiene i due interi N e M. Le successive M righe contengono due interi ciascuna, gli indici `da[`i`]`, `a[`i`]` delle isole collegate dall’i-esimo ponte.

input.txt	output.txt
4 2 1 3 3 2	1

input.txt	output.txt
2 0	1

“Ponti e isole” (seconda gara OIS 2015)

Caso limite

È possibile che la risposta al problema sia 0?

“Ponti e isole” (seconda gara OIS 2015)

Caso limite

È possibile che la risposta al problema sia 0?

Assunzioni

- $1 \leq N \leq 10\,000$.
- $0 \leq M \leq 100\,000$.
- $0 \leq \mathbf{da}[i], \mathbf{a}[i] < N$ per ogni $i = 0 \ldots M - 1$.
- Per ogni coppia di isole esiste al più un ponte che le collega, e i ponti non vengono ripetuti nell’input.
- Nessun ponte collega un’isola a se stessa.
- Le isole sono numerate a partire da 0.
- Se l’arcipelago è già connesso, rispondere il valore 0.

Spiegazione teorica

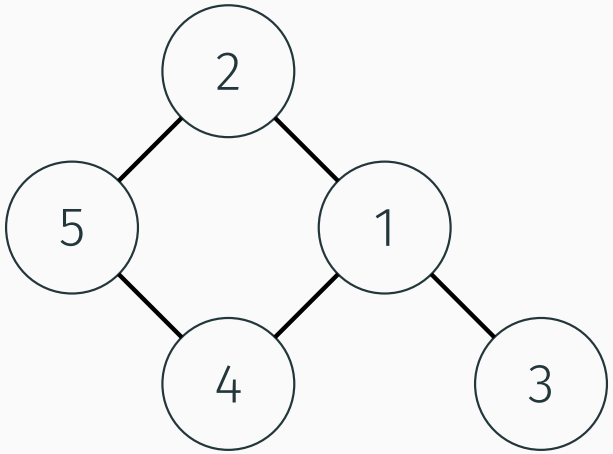
Grafi

Definizione

Un grafo è un insieme di elementi detti nodi (o vertici) che possono essere collegati fra loro da linee chiamate archi.

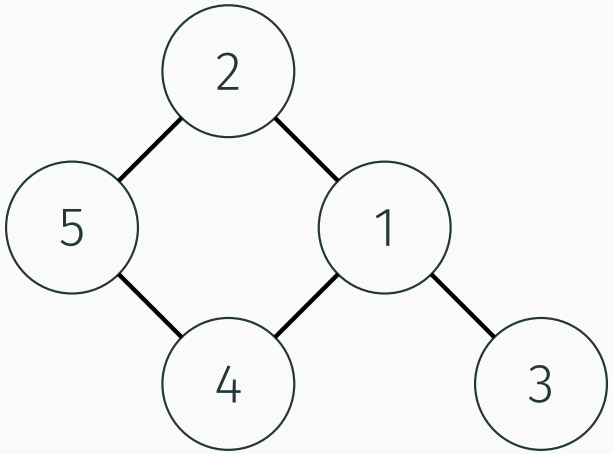
Formalmente, chiamiamo *grafo* una coppia ordinata $G = (V, E)$ dove V è l'insieme dei nodi ed E è l'insieme degli archi, tali che $E \subseteq V \times V$.

Grafi



$V =$

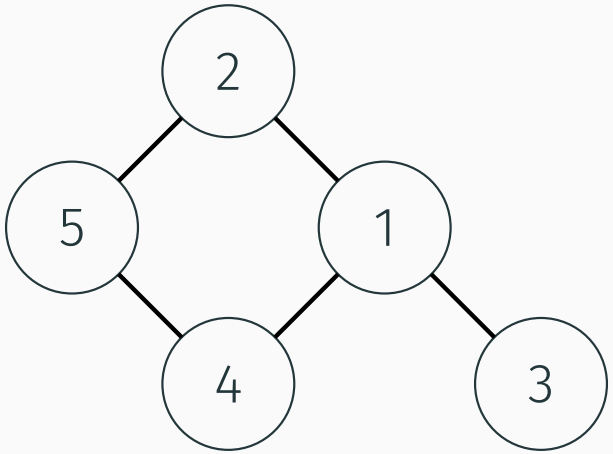
Grafi



$V = \{1, 2, 3, 4, 5\}$

$E =$

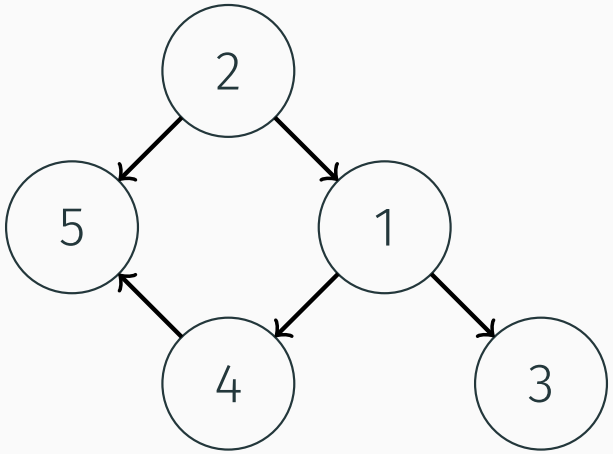
Grafi



$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{(2, 1), (1, 4), (4, 5), (1, 3), (2, 5)\}$$

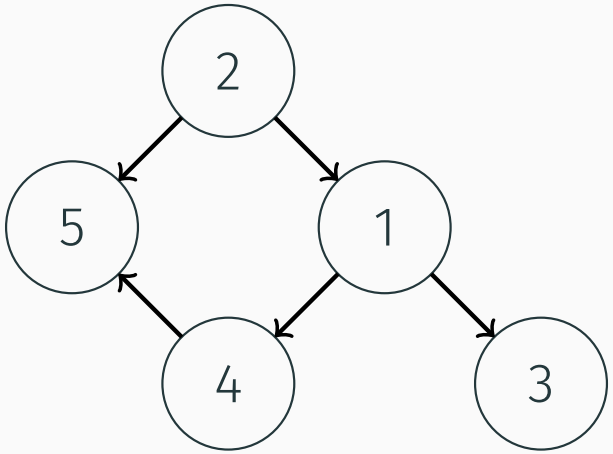
Grafi orientati



$V = \{1, 2, 3, 4, 5\}$

$E =$

Grafi orientati



$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{(2, 1), (1, 4), (4, 5), (1, 3), (2, 5)\}$$

Questa tipologia di grafo è detta grafo orientato (o “diretto”): gli archi hanno una direzione e le coppie in E sono ordinate.

Caratteristiche dei grafi

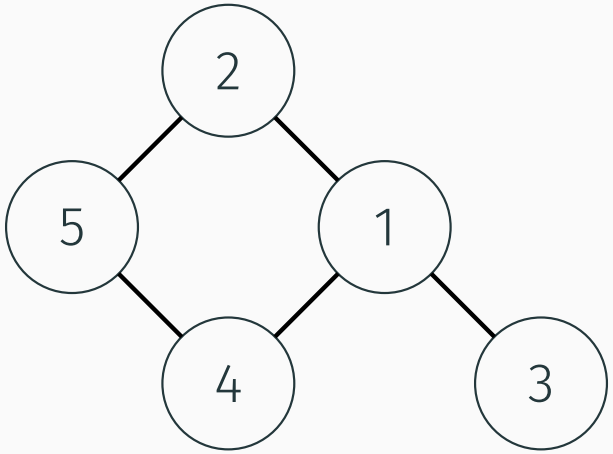
Cammino

Un cammino dal nodo v_1 al nodo v_n è una sequenza ordinata di vertici $(v_1, v_2, \ldots, v_n)$ tali che esistono gli archi $(v_1, v_2), \ldots, (v_{n-1}, v_n)$.

Caratteristiche dei grafi

Cammino

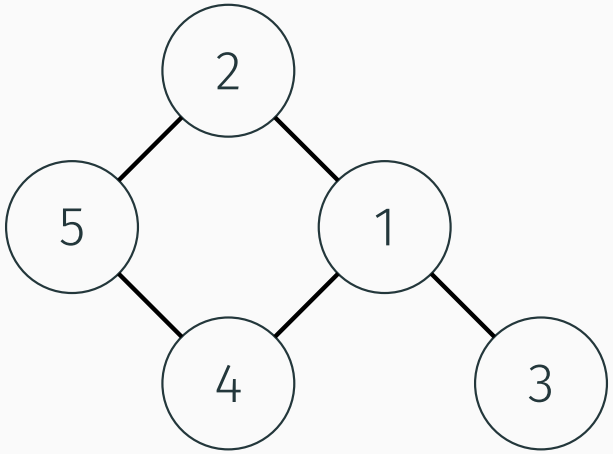
Un cammino dal nodo v_1 al nodo v_n è una sequenza ordinata di vertici $(v_1, v_2, \ldots, v_n)$ tali che esistono gli archi $(v_1, v_2), \ldots, (v_{n-1}, v_n)$.



Caratteristiche dei grafi

Cammino

Un cammino dal nodo v_1 al nodo v_n è una sequenza ordinata di vertici $(v_1, v_2, \ldots, v_n)$ tali che esistono gli archi $(v_1, v_2), \ldots, (v_{n-1}, v_n)$.



Ciclo

Un ciclo è un cammino chiuso, ovvero con $v_1 = v_n$.

Caratteristiche dei grafi

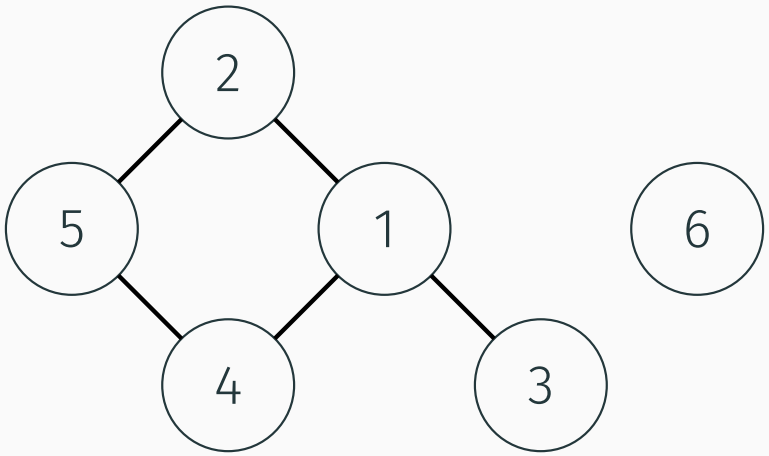
Raggiungibilità

Un nodo v è raggiungibile dal nodo u sse esiste un cammino tra u e v.

Caratteristiche dei grafi

Raggiungibilità

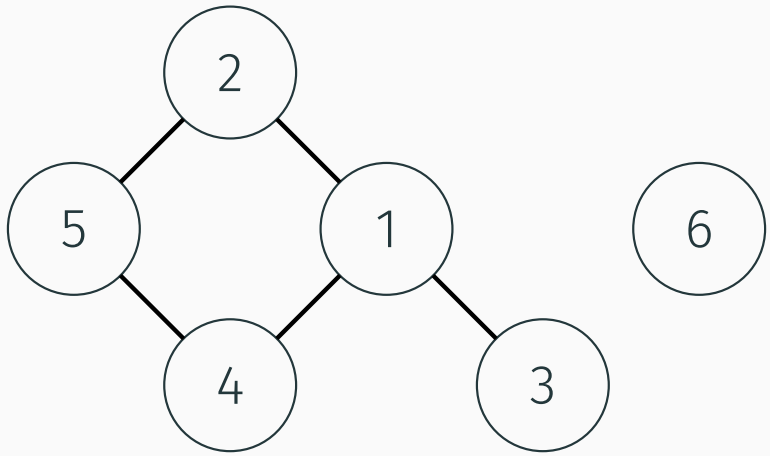
Un nodo *v* è raggiungibile dal nodo *u* sse esiste un cammino tra *u* e *v*.



Caratteristiche dei grafi

Raggiungibilità

Un nodo *v* è raggiungibile dal nodo *u* sse esiste un cammino tra *u* e *v*.

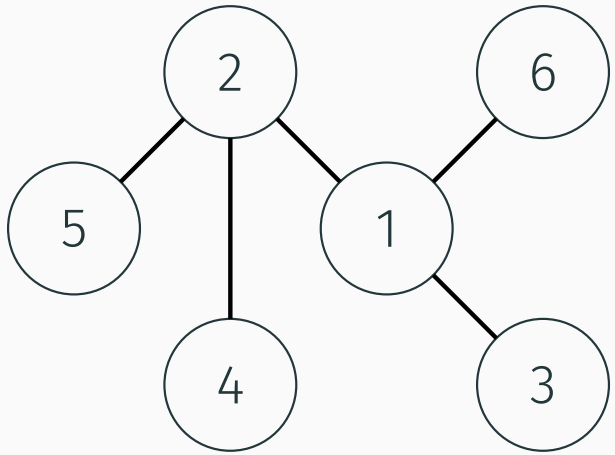


Qualora tutti i nodi siano raggiungibili da tutti gli altri, il grafo è detto *connesso*.

Caratteristiche dei grafi

Alberi

Un grafo G non orientato è detto *albero* se è connesso e aciclico.

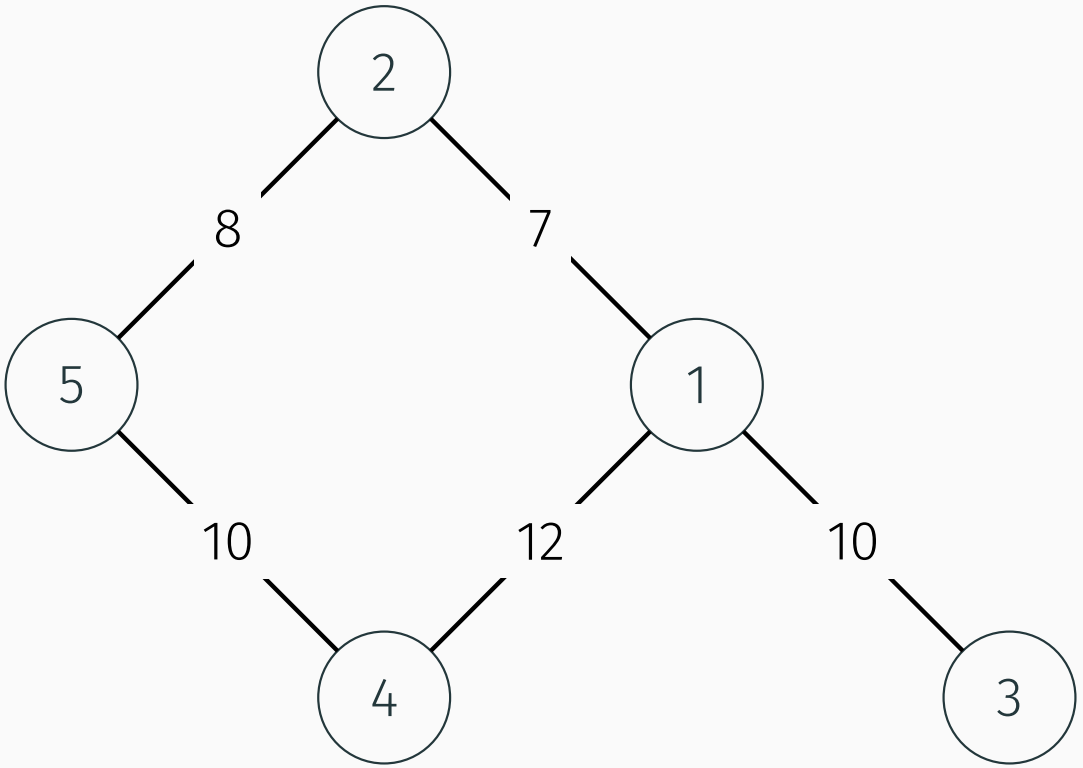


Caratteristiche dei grafi

Grafo pesato

Un grafo $G(V, E)$ si dice *pesato* se è dotato di una funzione di peso $w : E \to \mathbb{R}$.

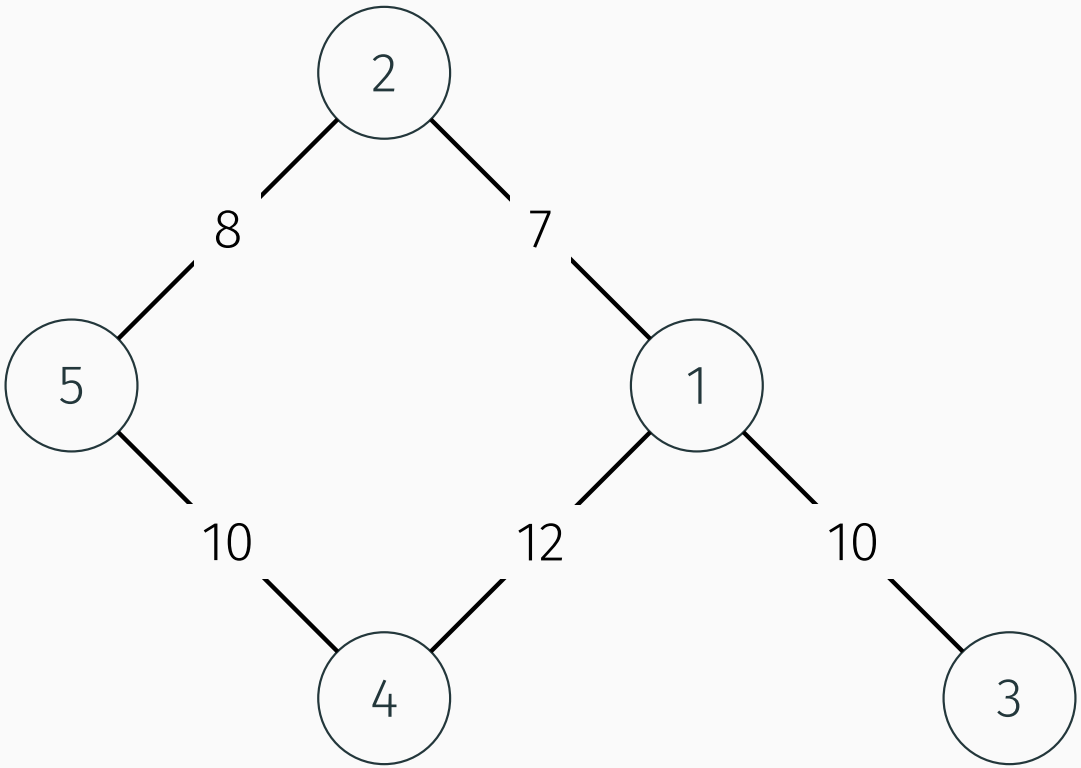
Tale funzione associa ad ogni arco un valore detto *peso* (spesso chiamato anche *distanza* o *costo* in alcuni contesti).



Caratteristiche dei grafi

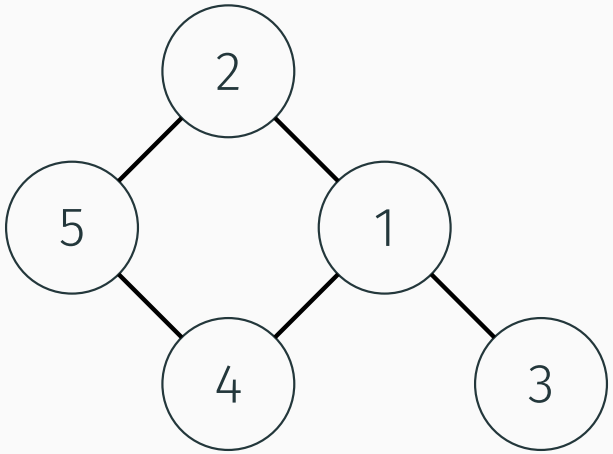
Cammino minimo

Tra tutti i cammini esistenti tra *u* e *v*, chiamiamo *cammino minimo* quello che minimizza il costo di attraversamento degli archi che lo compongono.



Rappresentazione di grafi

Rappresentazione di grafi



Matrice di adiacenza

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$m_{ij} = \begin{cases} 1 & \text{sse } (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Rappresentazione di grafi - Matrice di adiacenza

Occupazione di memoria

Quanto occupa una matrice di adiacenza per rappresentare un grafo (ricordiamo $G = (V, E)$)?

Rappresentazione di grafi - Matrice di adiacenza

Occupazione di memoria

Quanto occupa una matrice di adiacenza per rappresentare un grafo (ricordiamo $G = (V, E)$)?

$$\mathcal{O}(|V|^2)$$

Intuitivamente, dov'è lo "spreco" di spazio?

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Rappresentazione di grafi - Matrice di adiacenza

Occupazione di memoria

Quanto occupa una matrice di adiacenza per rappresentare un grafo (ricordiamo $G = (V, E)$)?

$$\mathcal{O}(|V|^2)$$

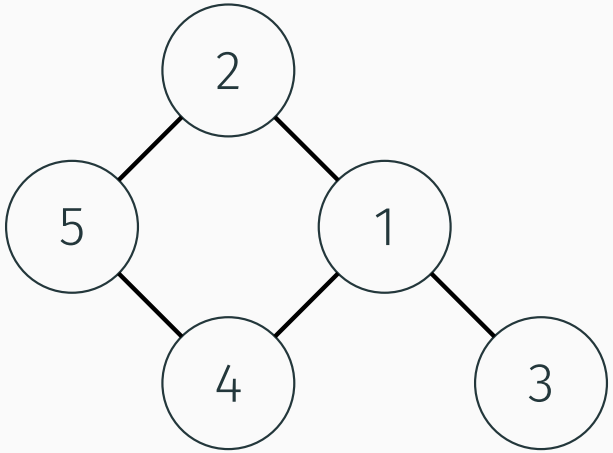
Intuitivamente, dov'è lo "spreco" di spazio?

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Stiamo memorizzando tantissimi "zeri": la matrice di adiacenza consuma troppo spazio soprattutto per grafi *sparsi*.

Rappresentazione di grafi - Liste di adiacenza

Rappresentazione di grafi - Liste di adiacenza



1 $\Rightarrow$ 2, 4, 3

2 $\Rightarrow$ 1, 5

3 $\Rightarrow$ 1

4 $\Rightarrow$ 1, 5

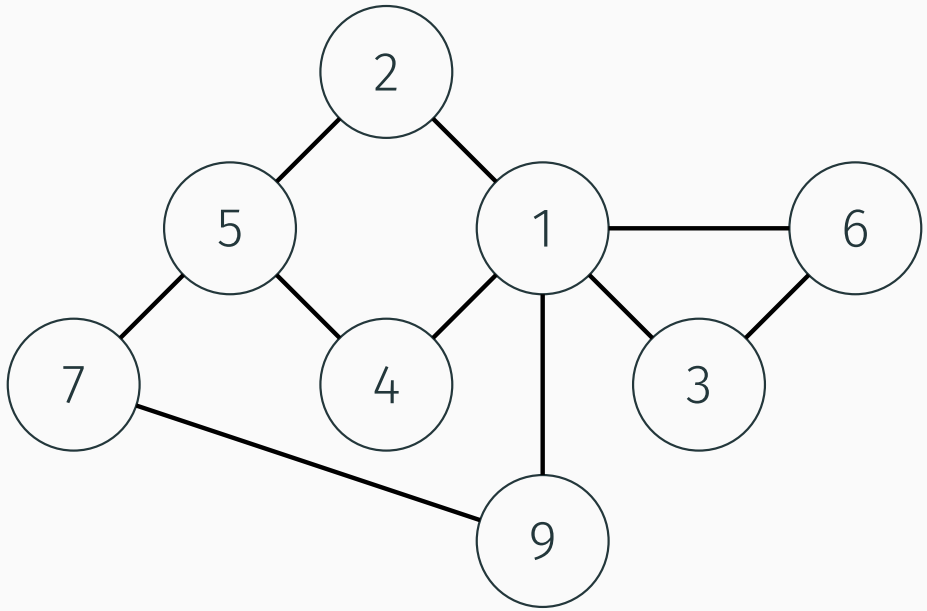
5 $\Rightarrow$ 2, 4

Rappresentazione di grafi - Liste di adiacenza in C++

```cpp
#include <list>
using namespace std;

#define MAXN 10000

list<int> grafo[MAXN];

for (int i = 0; i < M; i++)
{
    int u = da[i], v = a[i];
    grafo[u].push_back(v);
    grafo[v].push_back(u);
}
```

Algoritmi di visita di un grafo

Dato un grafo vorremmo essere in grado, ovviamente con un algoritmo, di “visitare” (ovvero scoprire, enumerare, cercare) tutti i suoi vertici.



Algoritmi di visita di un grafo

Dato un grafo vorremmo essere in grado, ovviamente con un algoritmo, di “visitare” (ovvero scoprire, enumerare, cercare) tutti i suoi vertici.

Due semplici strategie:

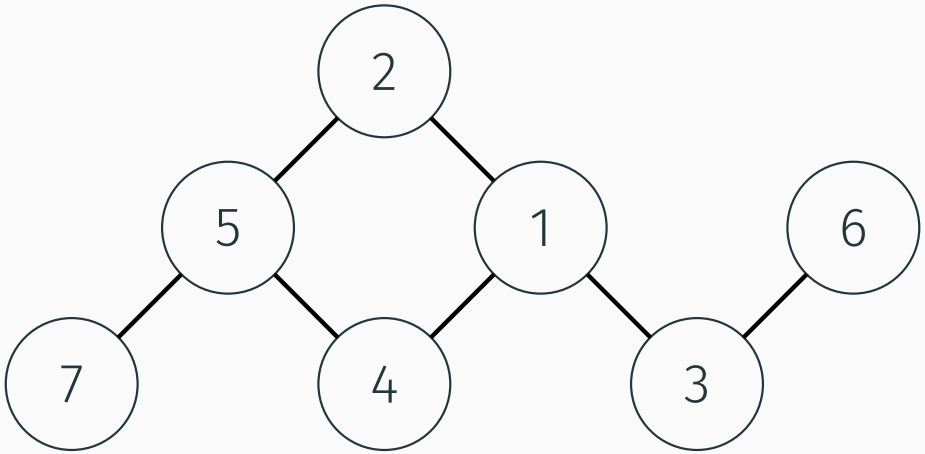
Visita in profondità (DFS)

Per ogni nodo, visita fin quando possibile uno dei nodi ad esso adiacente.

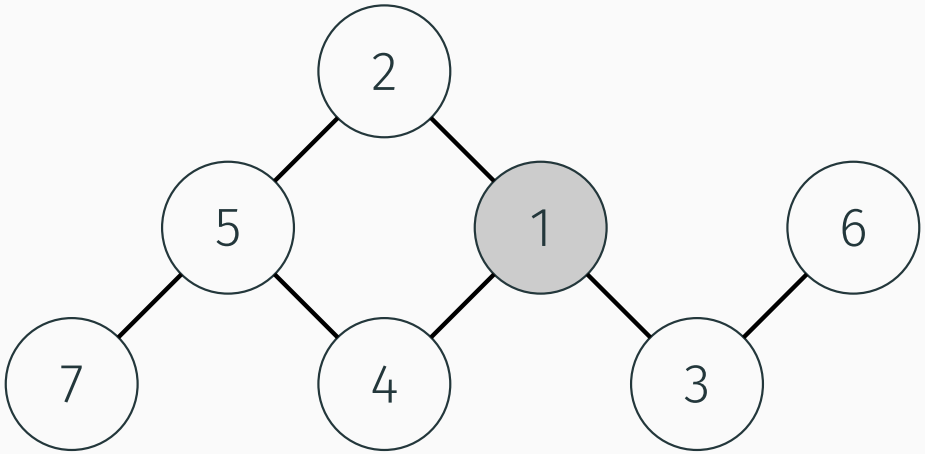
Visita in ampiezza (BFS)

Per ogni nodo, visita tutti i nodi ad esso adiacenti prima di spostarsi agli adiacenti degli adiacenti.

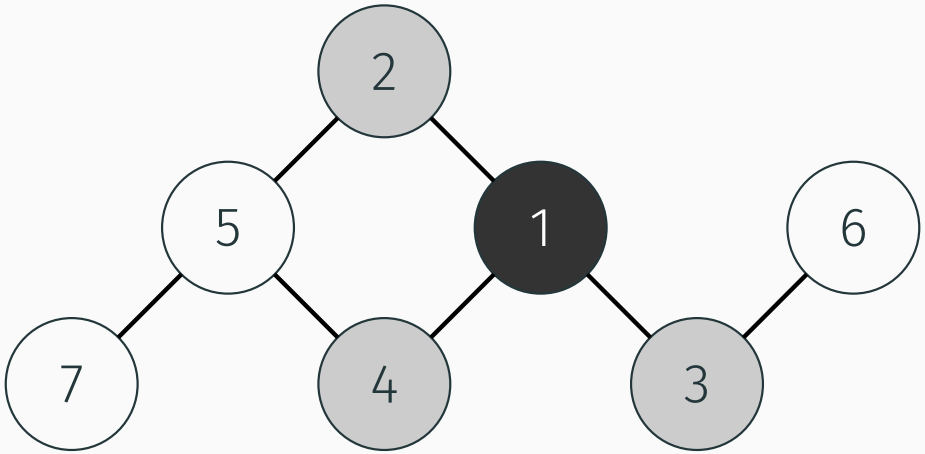
Visita BFS



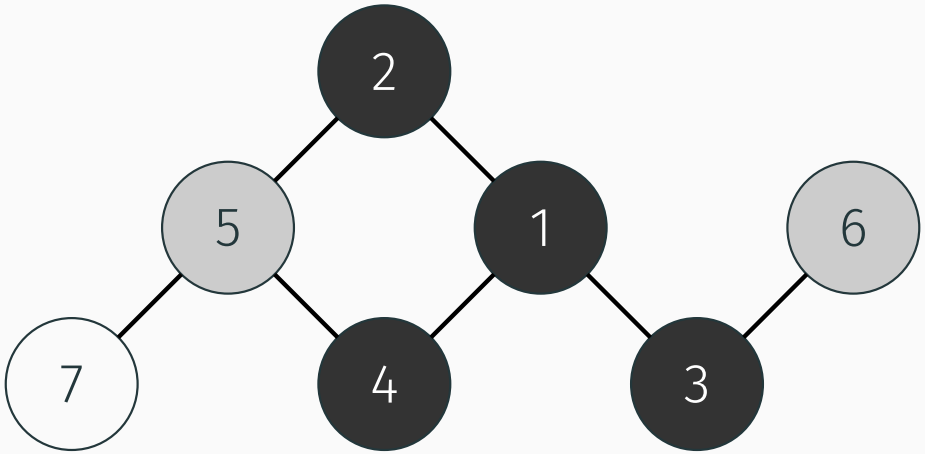
Visita BFS - Frontiera 0



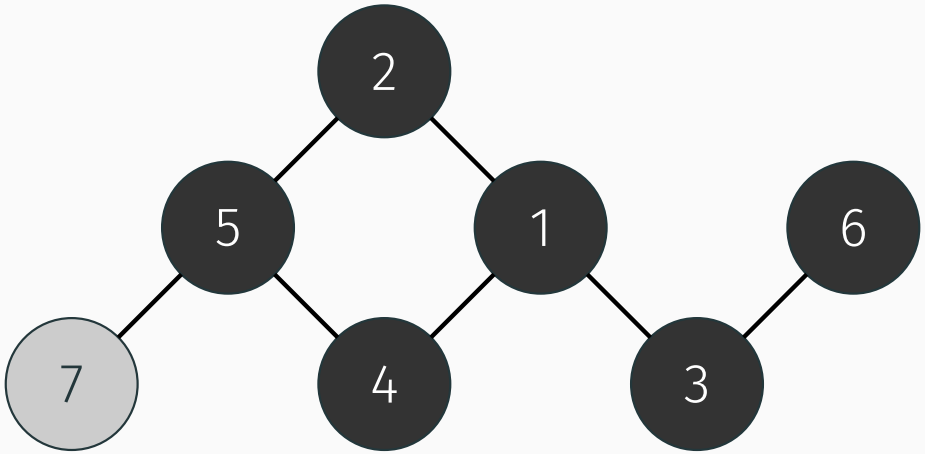
Visita BFS - Frontiera 1



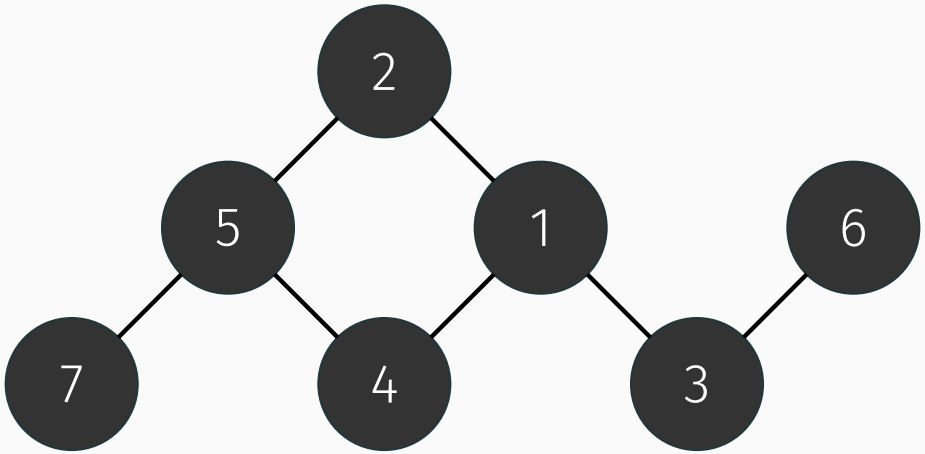
Visita BFS - Frontiera 2



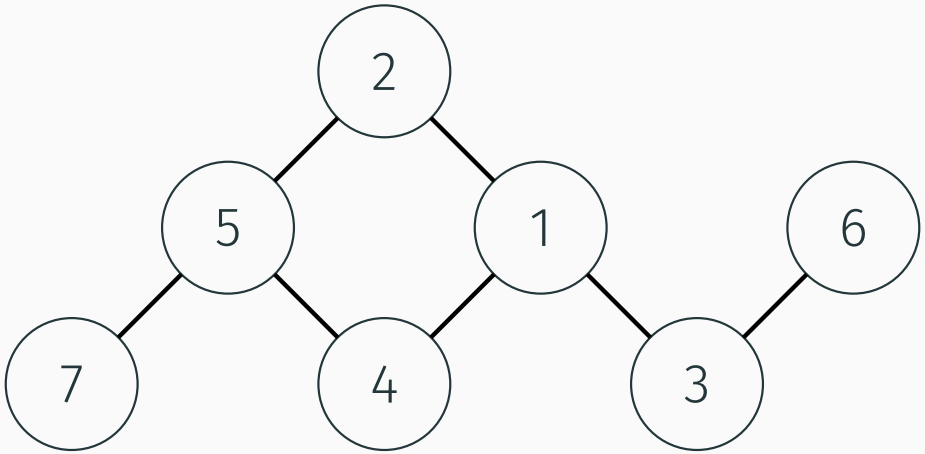
Visita BFS - Frontiera 3



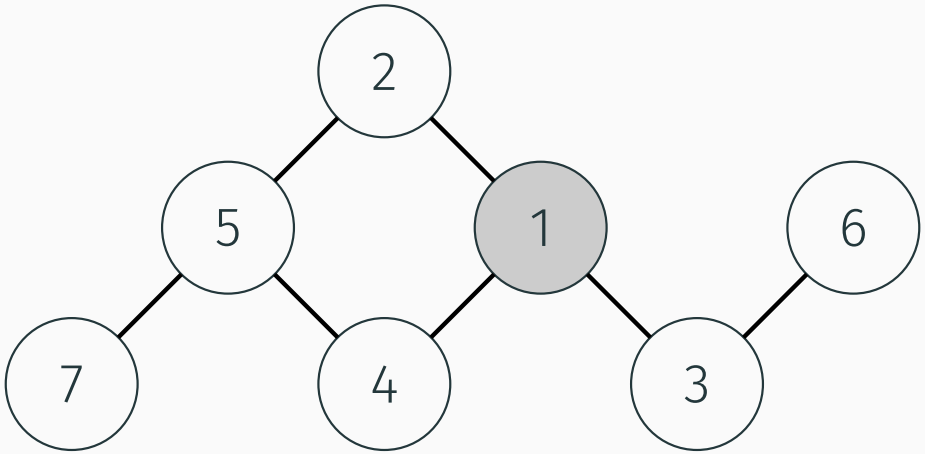
Visita BFS



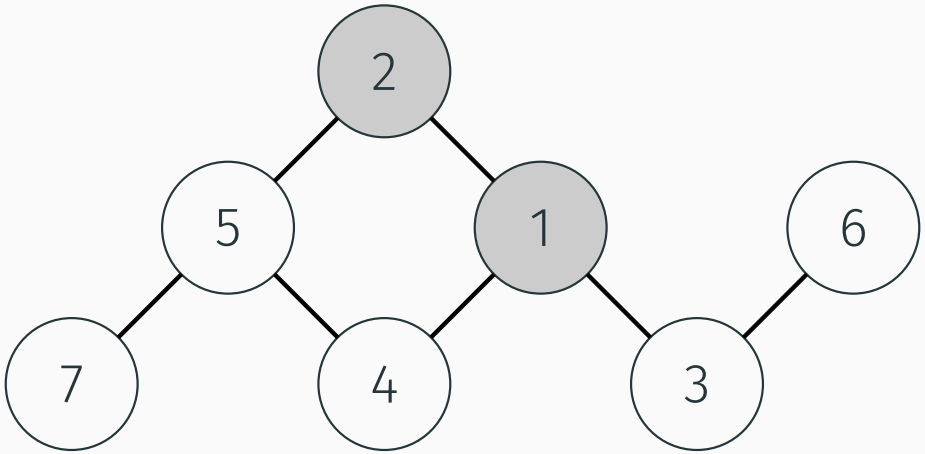
Visita DFS



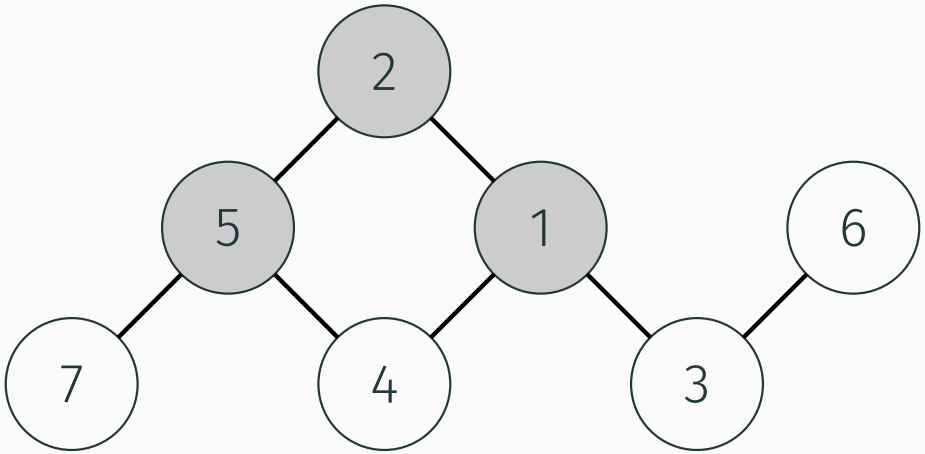
Visita DFS - Passo 0



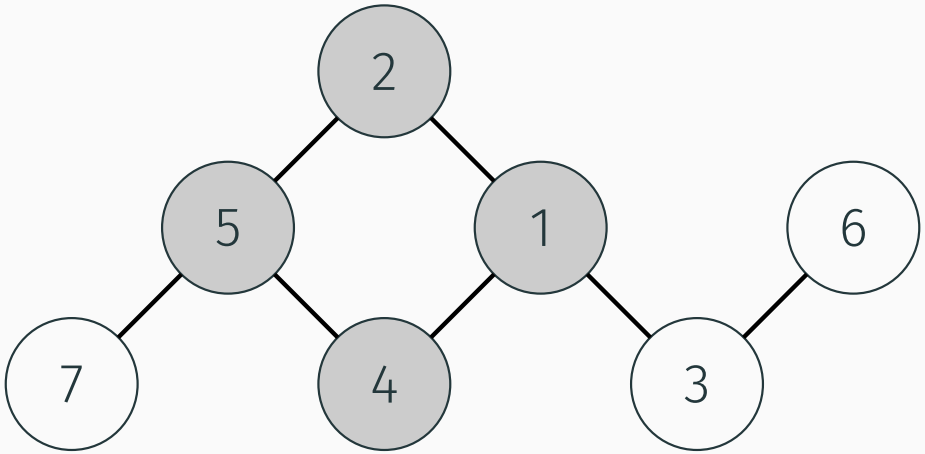
Visita DFS - Passo 1



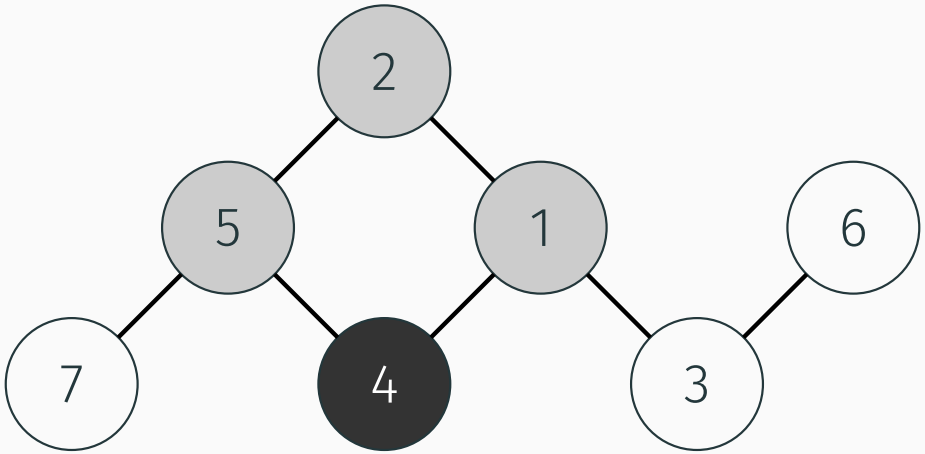
Visita DFS - Passo 2



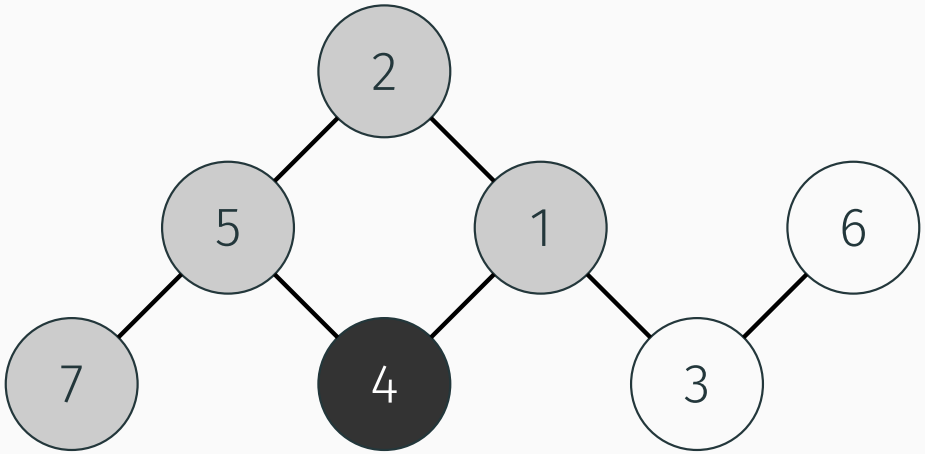
Visita DFS - Passo 3



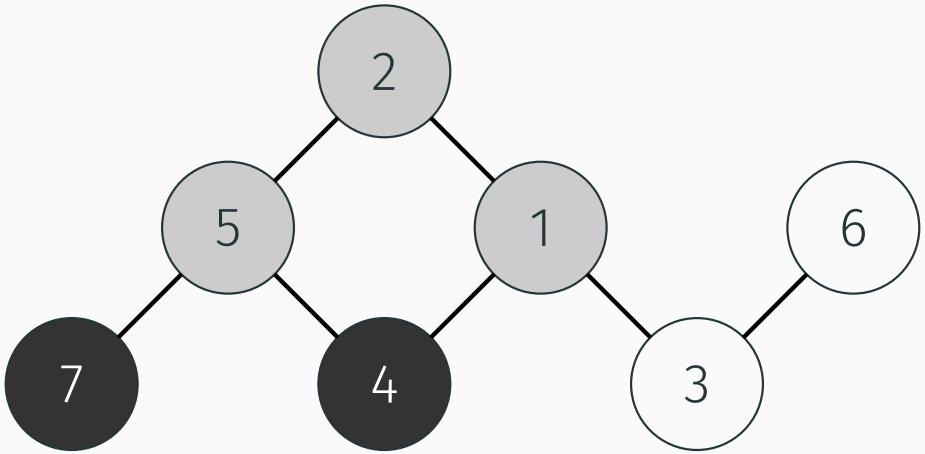
Visita DFS - Passo 4



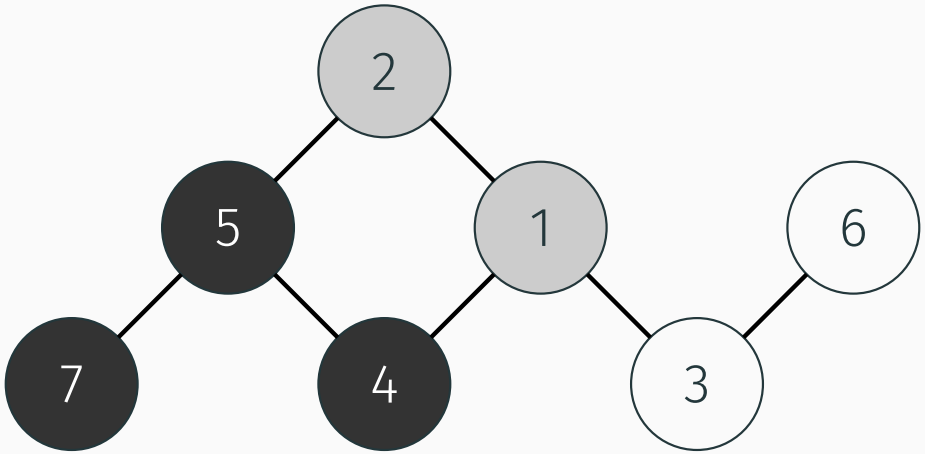
Visita DFS - Passo 5



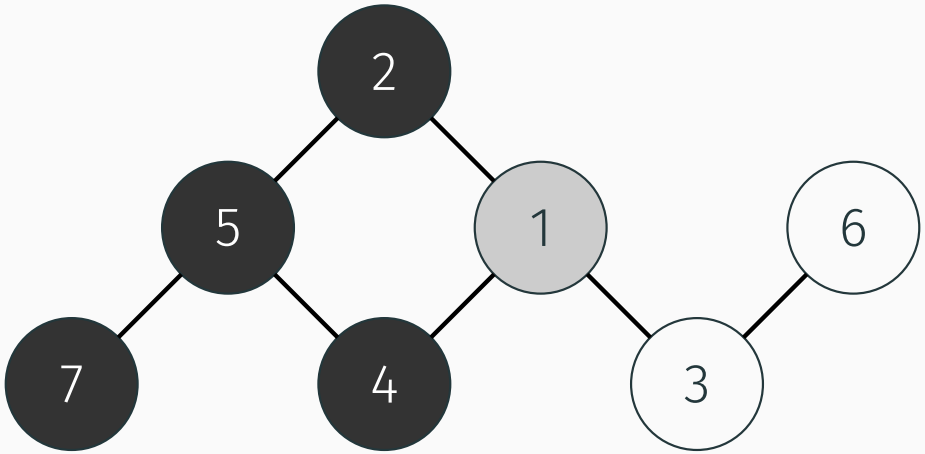
Visita DFS - Passo 6



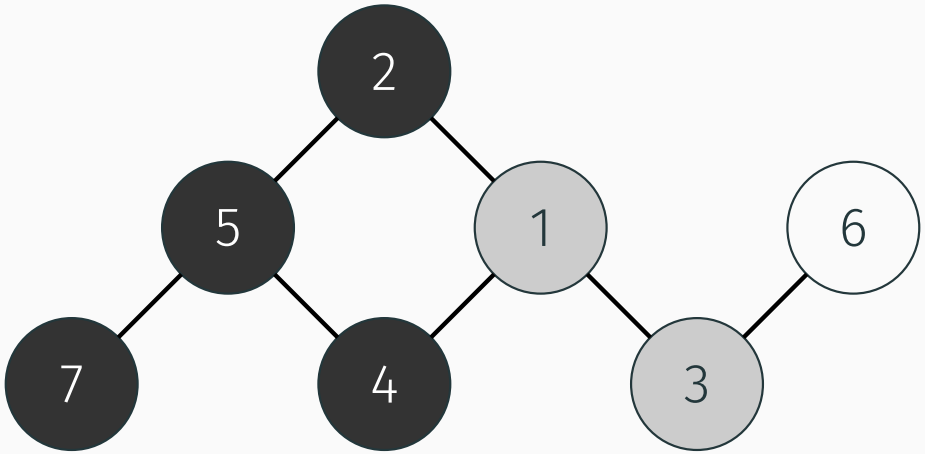
Visita DFS - Passo 7



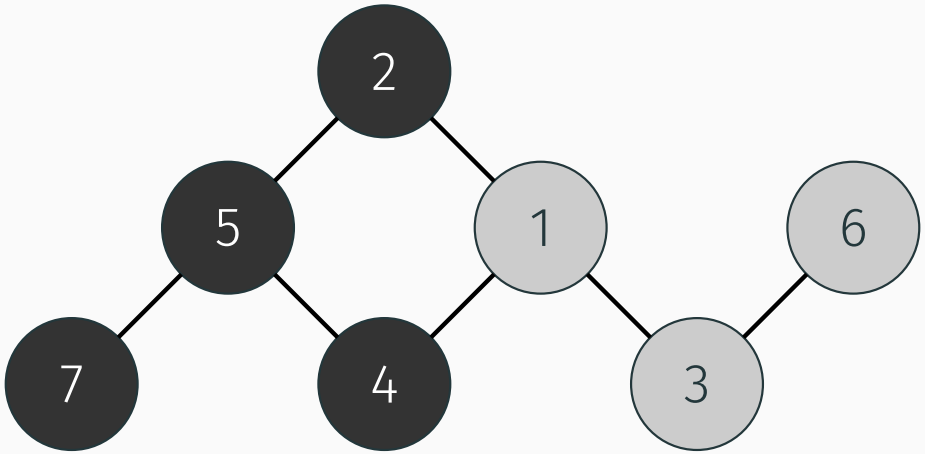
Visita DFS - Passo 8



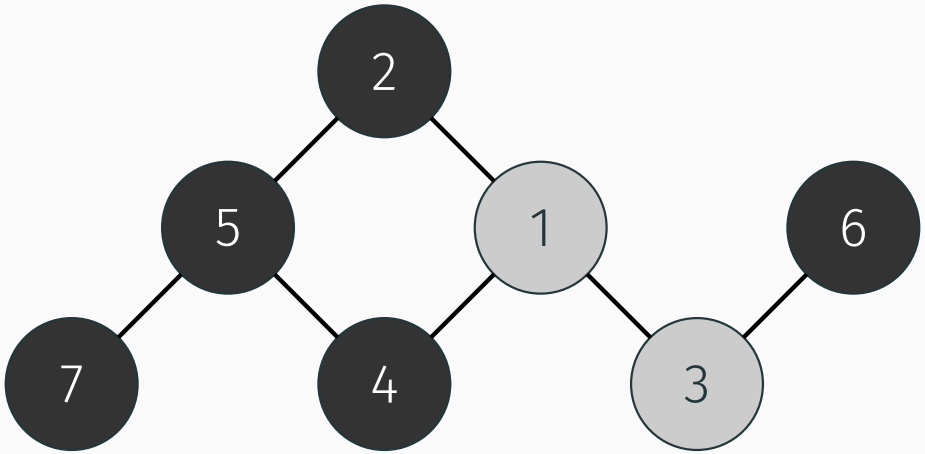
Visita DFS - Passo 9



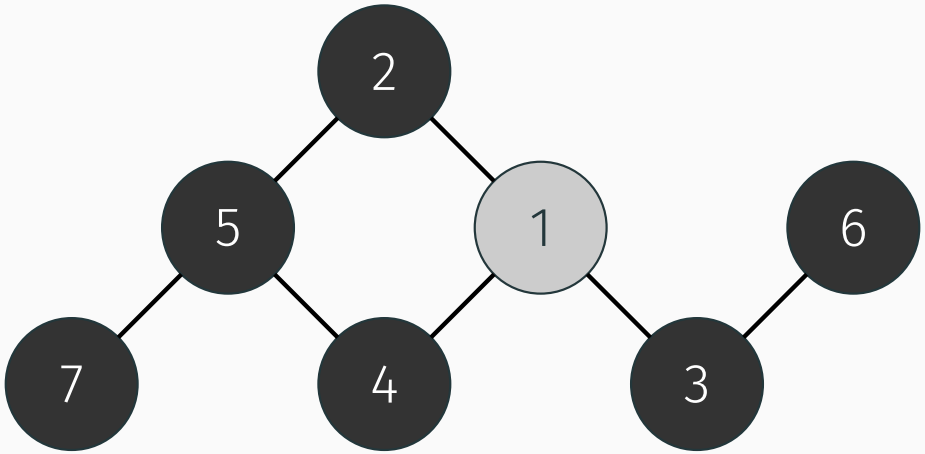
Visita DFS - Passo 10



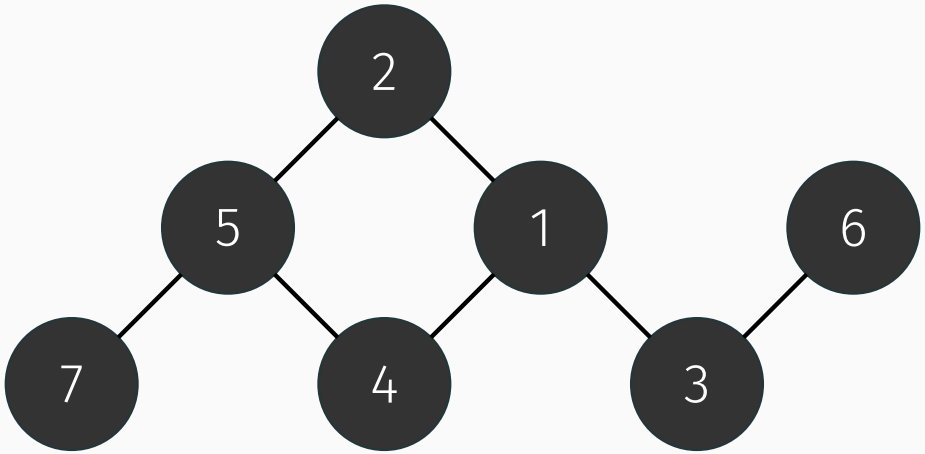
Visita DFS - Passo 11



Visita DFS - Passo 12



Visita DFS - Passo 13



BFS/DFS: complessità computazionale

Qual è la complessità di BFS e di DFS?

Ricordiamo che un grafo è espresso formalmente come $G = (V, E)$.

BFS/DFS: complessità computazionale

Qual è la complessità di BFS e di DFS?

Ricordiamo che un grafo è espresso formalmente come $G = (V, E)$.

Complessità computazionale di BFS e DFS

In entrambe le visite analizziamo al più una volta ciascun vertice e ciascun arco, quindi la complessità risulta $\mathcal{O}(|V| + |E|)$.

In sintesi: una visita richiede tempo lineare sia rispetto al numero di nodi che al numero di archi (attenzione: potenzialmente $|E| = |V|^2$).

BFS in C++

```
void bfs(int nodo_partenza)
{
    queue<int> coda;
    coda.push(nodo_partenza);

    while (!coda.empty()) {
        int nodo = coda.front();
        coda.pop();
        visitato[nodo] = true;
        for (int adiacente: grafo[nodo])
            if (!visitato[adiacente])
                coda.push(adiacente);
    }
}
```

DFS in C++

```
void dfs(int nodo)
{
    visitato[nodo] = true;

    // Scorre la lista di adiacenza di 'nodo'
    // e procede ricorsivamente.
    for (int adiacente: grafo[nodo])
        if (!visitato[adiacente])
            dfs(adiacente);
}
```

Soluzione

“Ponti e isole” (seconda gara OIS 2015)

Ripropongo il testo. È più facile da leggere ora?

“Ponti e isole” (seconda gara OIS 2015)

Ripropongo il testo. È più facile da leggere ora?

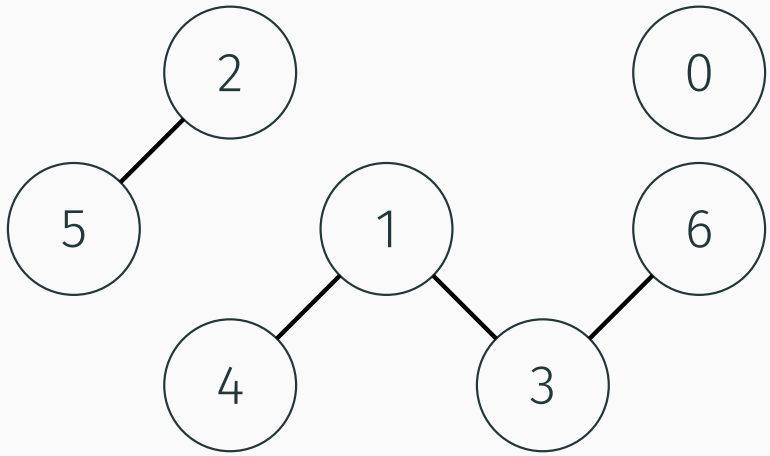
A seguito di un violento maremoto alcuni dei ponti che collegano le N isole dell’arcipelago Nowhere sono stati distrutti e il governo deve correre ai ripari per non lasciare che alcune isolette rimangano isolate e irraggiungibili.
Il governo dell’arcipelago Nowhere ha quindi assunto Giorgio per determinare quale è il minimo numero di ponti che è necessario costruire in aggiunta agli M rimasti affinché l’arcipelago sia di nuovo connesso, ovvero sia possibile da ogni isola raggiungere tutte le altre isole. Aiuta Giorgio a svolgere il suo compito!

Modellazione tramite grafo

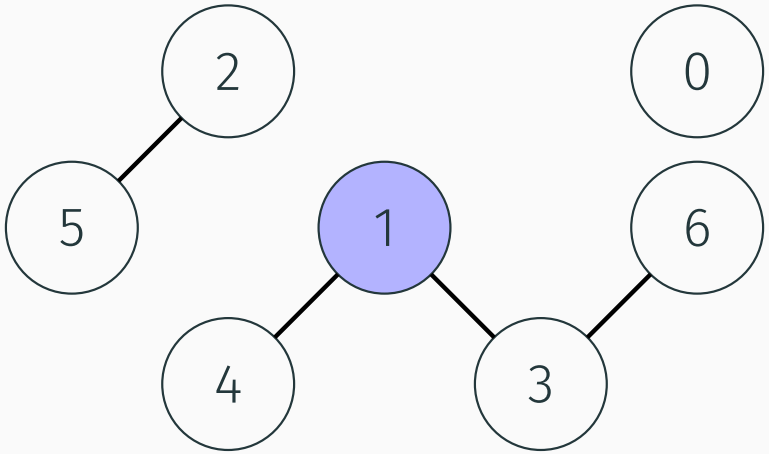
input.txt	output.txt
7 4 1 4 2 5 1 3 3 6	2

Modellazione tramite grafo

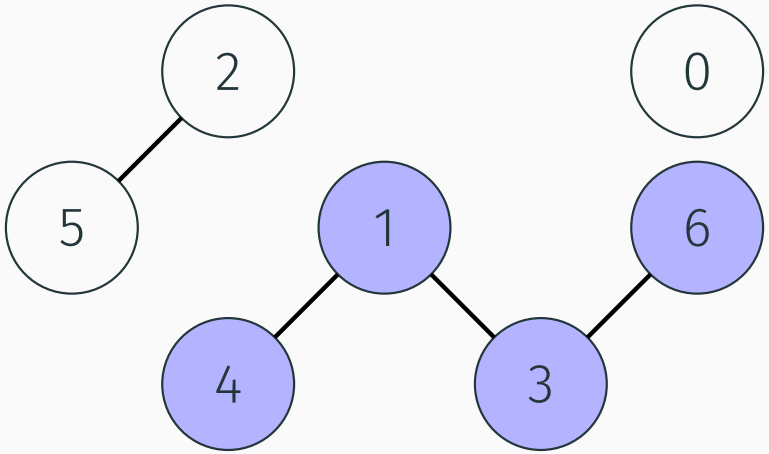
input.txt	output.txt
7 4 1 4 2 5 1 3 3 6	2



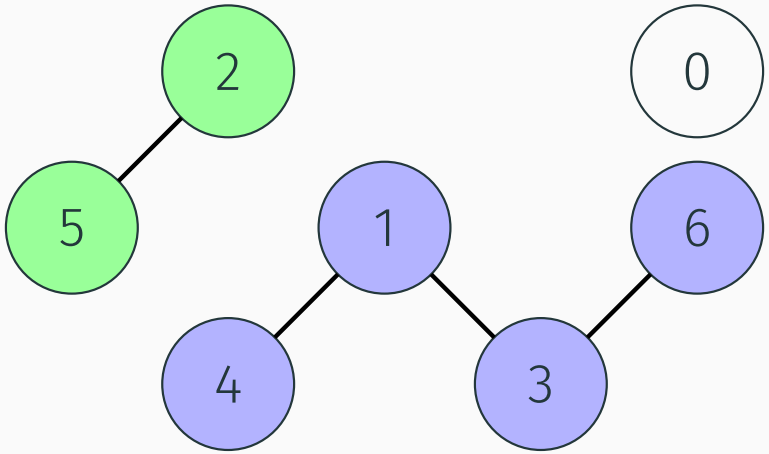
Idea di soluzione



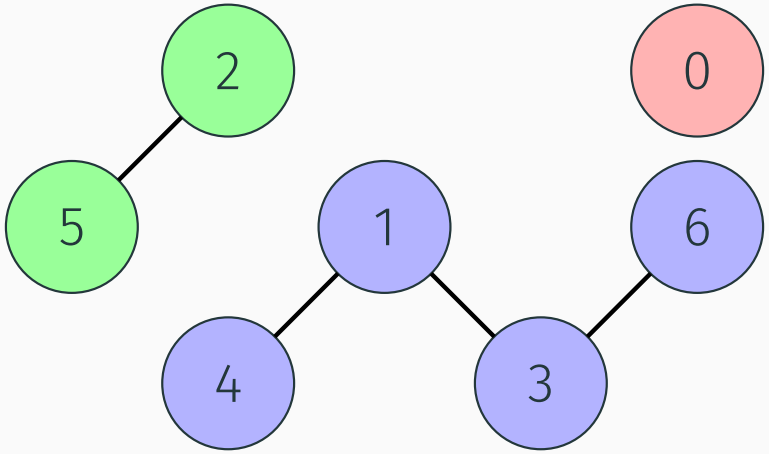
Idea di soluzione



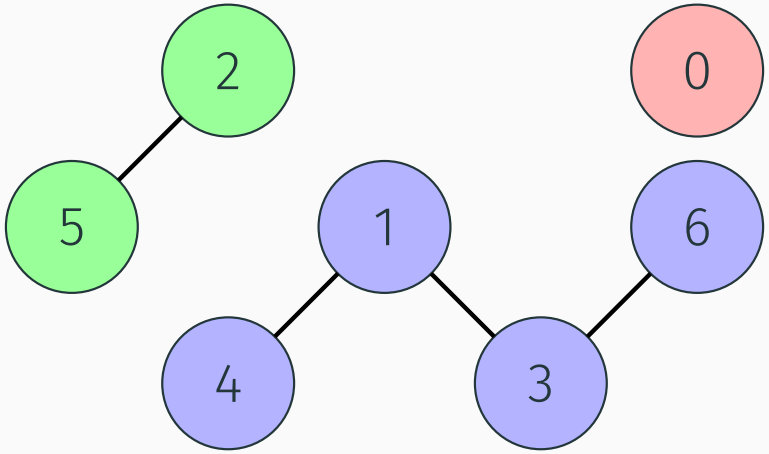
Idea di soluzione



Idea di soluzione



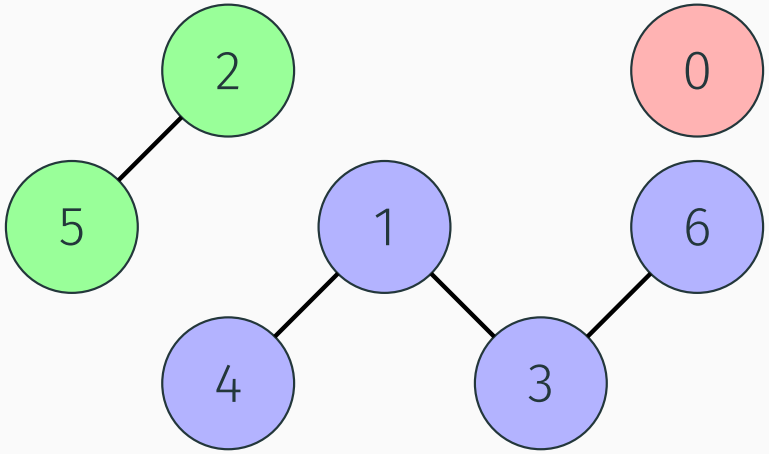
Idea di soluzione



Soluzione

Qual è la risposta al problema?

Idea di soluzione



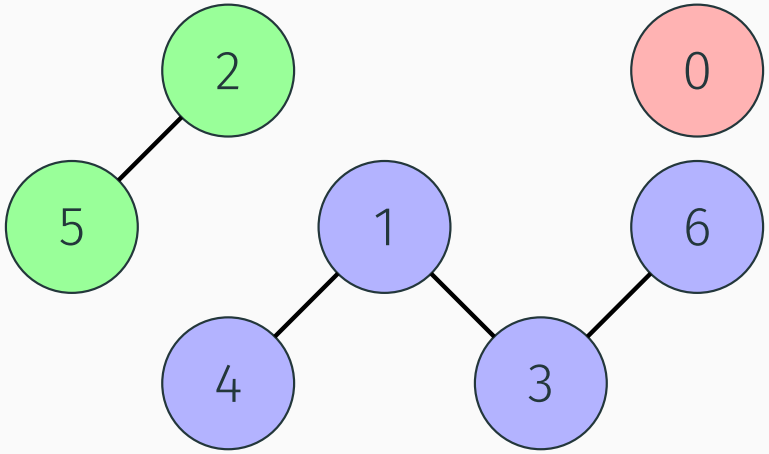
Soluzione

Qual è la risposta al problema? “Numero di colori” - 1

Complessità computazionale

Qual è la complessità computazionale della soluzione?

Idea di soluzione



Soluzione

Qual è la risposta al problema? “Numero di colori” - 1

Complessità computazionale

Qual è la complessità computazionale della soluzione?

Ciascuna visita impiega $\mathcal{O}(N)$, quindi sembrerebbe $\mathcal{O}(N^2)$. Tuttavia è semplice osservare che complessivamente non visitiamo mai un nodo più di una volta, il che implica $\mathcal{O}(N)$.

Altri problemi su grafi

Problemi su grafi... non troppo complessi

Altri problemi su grafi

Problemi su grafi... non troppo complessi

- `depura`
  - Territoriali 2009
  - Grafo delle precedenze
  - Visita del grafo

Altri problemi su grafi

Problemi su grafi... non troppo complessi

- `depura`
  - Territoriali 2009
  - Grafo delle precedenze
  - Visita del grafo
- `fulcro`
  - GATOR 2014
  - Visita del grafo

Altri problemi su grafi

Problemi su grafi... non troppo complessi

- `depura`
  - Territoriali 2009
  - Grafo delle precedenze
  - Visita del grafo
- `fulcro`
  - GATOR 2014
  - Visita del grafo
- `sentieri`
  - Territoriali 2016
  - 01-BFS oppure Dijkstra

Algoritmo di Dijkstra

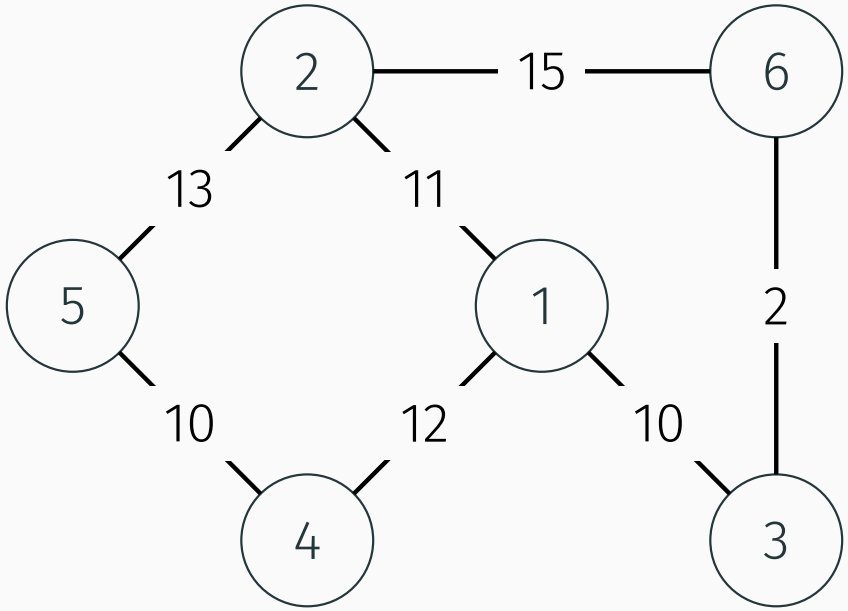
Algoritmo di Dijkstra

Dato un grafo pesato $G = (V, E, w)$, l'algoritmo di Dijkstra risolve il problema del *cammino minimo*.

Dato un vertice sorgente, l'algoritmo determina il cammino di costo minimo verso tutti gli altri nodi.

Algoritmo di Dijkstra

Dato un vertice sorgente, l'algoritmo determina il cammino di costo minimo verso tutti gli altri nodi.

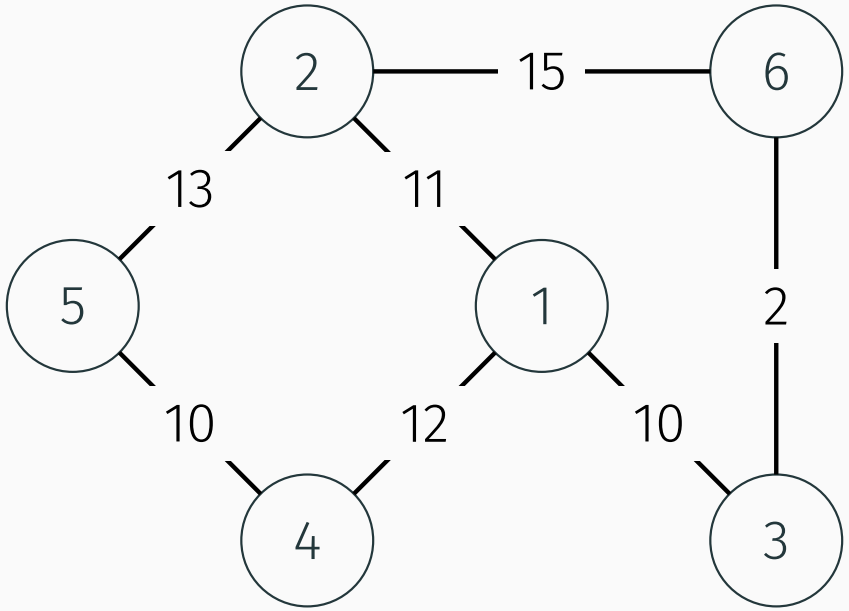


Vogliamo ottenere (dato il vertice 1 come sorgente):

`dist`

0	11	10	12	22	12

Algoritmo di Dijkstra

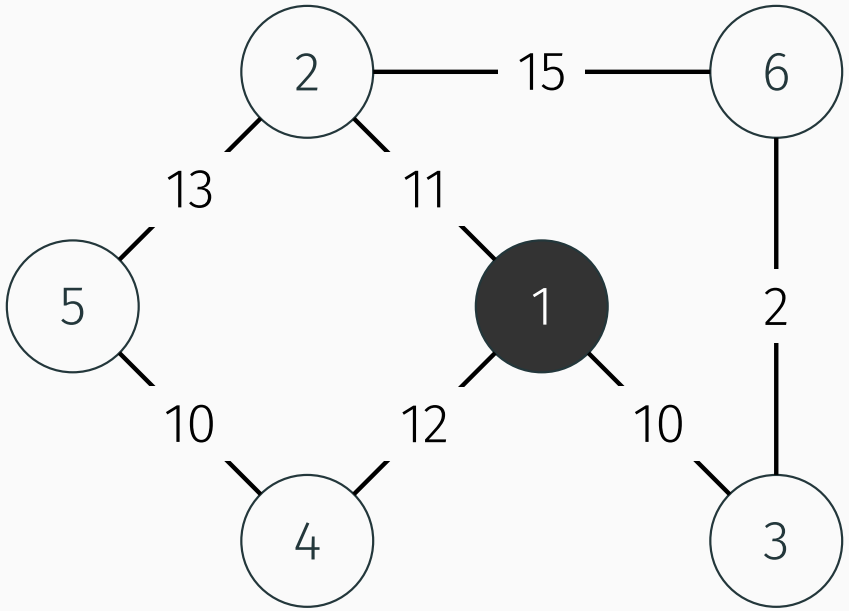


Inizializzazione:

`dist`

0	∞	∞	∞	∞	∞

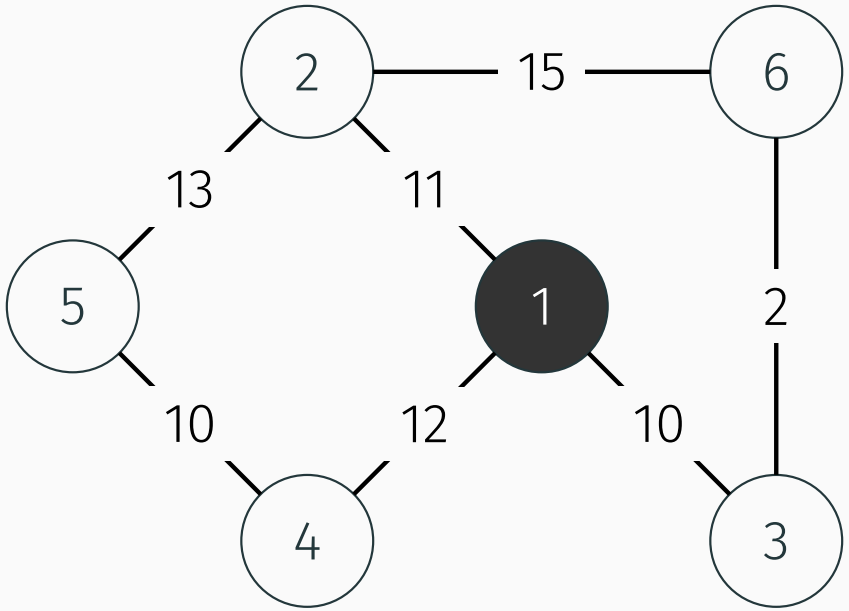
Algoritmo di Dijkstra



“Relax” degli adiacenti al nodo 1:

`dist`	0	11	10	12	∞	∞

Algoritmo di Dijkstra

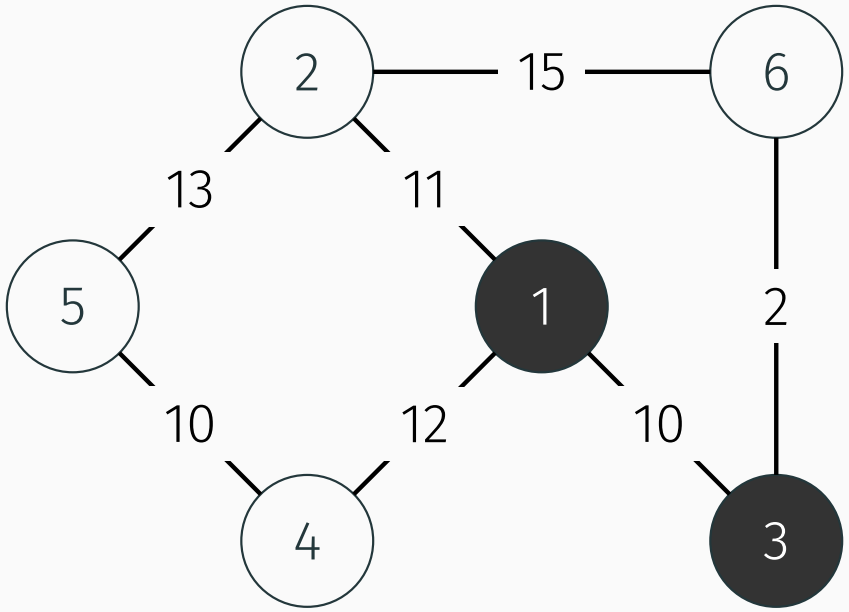


“Relax” degli adiacenti al nodo 1:

`dist`	0	11	10	12	∞	∞

Considero il nodo a distanza minore non ancora completato.

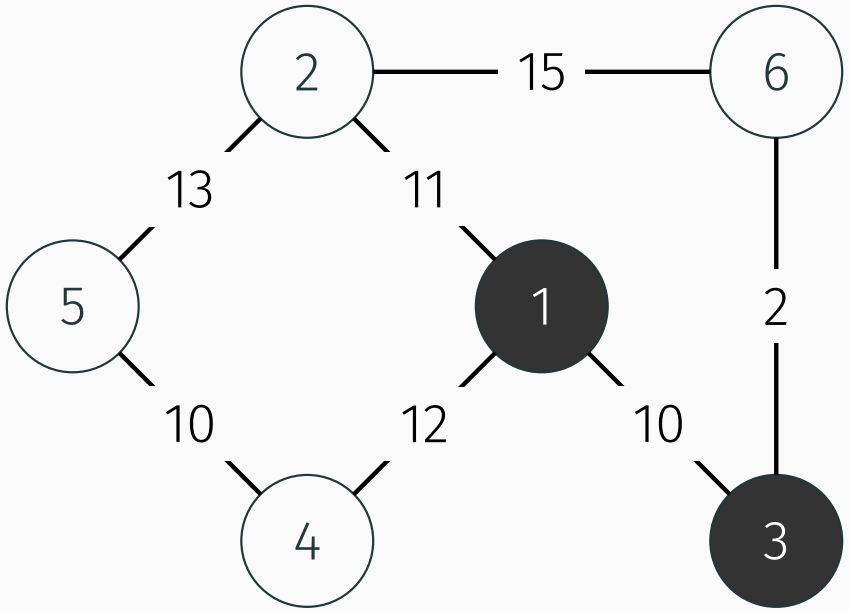
Algoritmo di Dijkstra



“Relax” degli adiacenti al nodo 3:

`dist`	0	11	10	12	∞	12

Algoritmo di Dijkstra

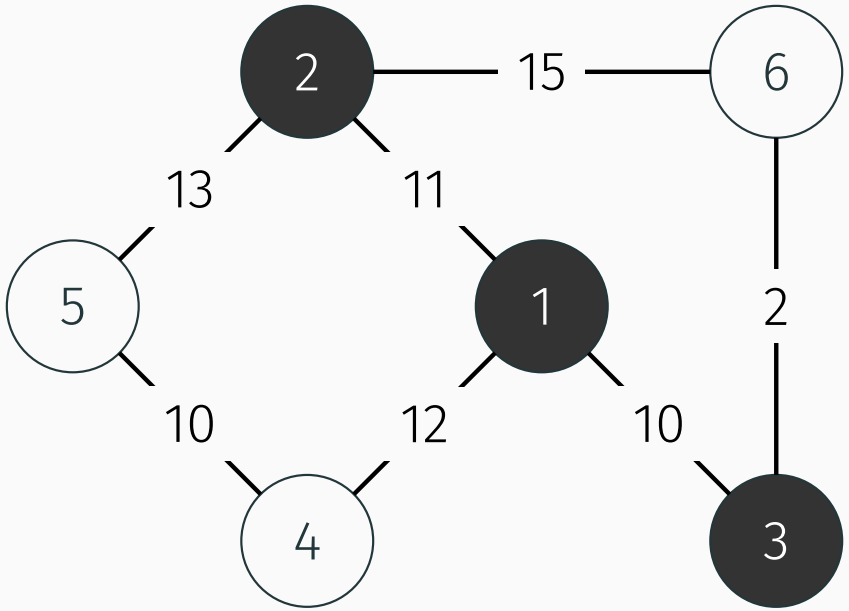


“Relax” degli adiacenti al nodo 3:

`dist`	0	11	10	12	∞	12

Considero il nodo a distanza minore non ancora completato.

Algoritmo di Dijkstra

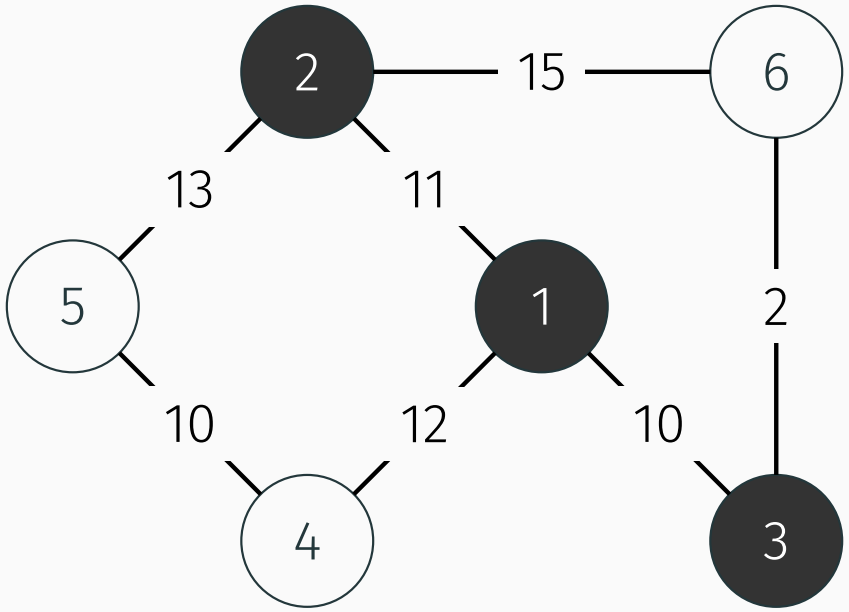


“Relax” degli adiacenti al nodo 2:

`dist`

0	11	10	12	24	12

Algoritmo di Dijkstra

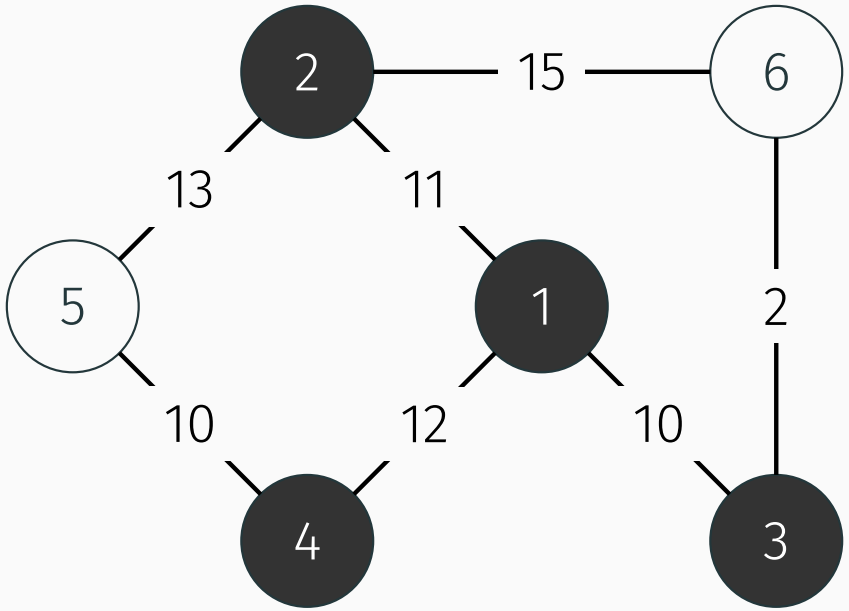


“Relax” degli adiacenti al nodo 2:

`dist`	0	11	10	12	24	12

Considero il nodo a distanza minore non ancora completato.

Algoritmo di Dijkstra

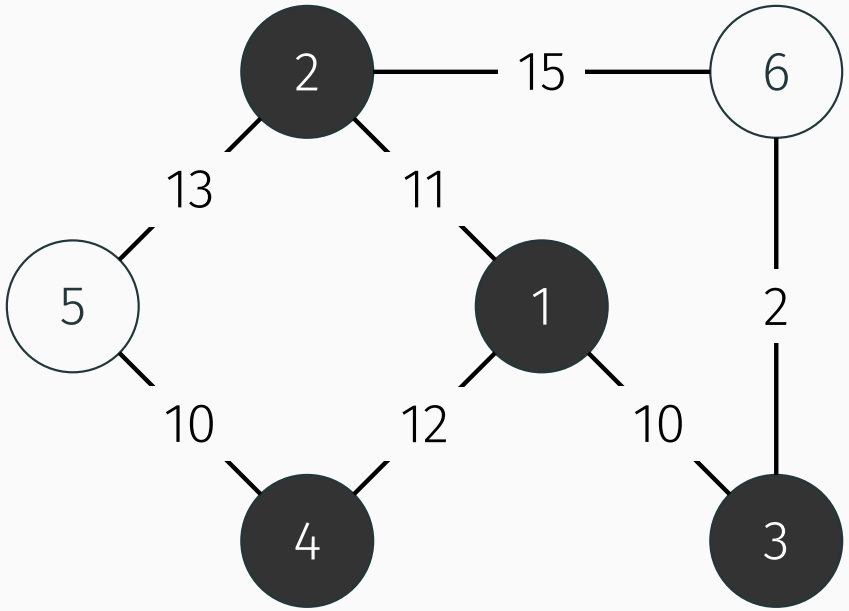


“Relax” degli adiacenti al nodo 4:

`dist`

0	11	10	12	22	12

Algoritmo di Dijkstra

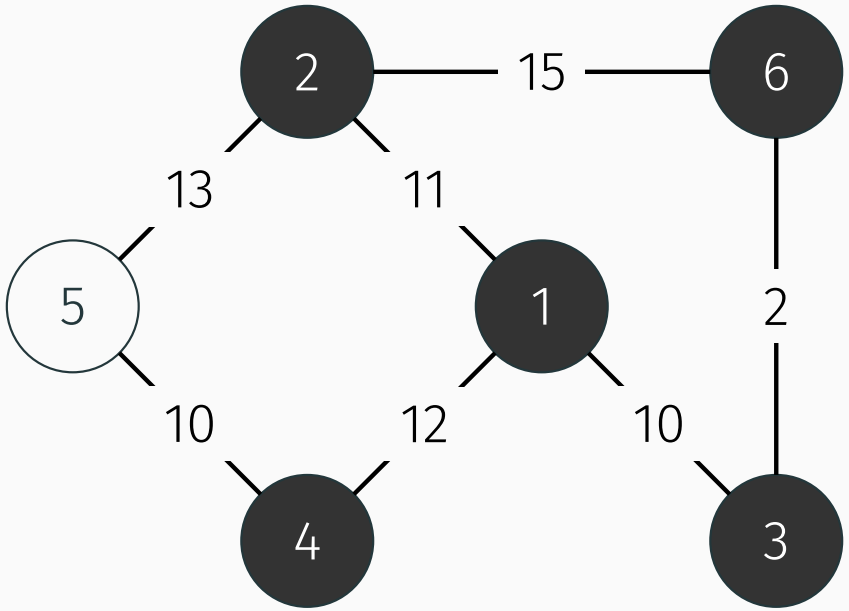


“Relax” degli adiacenti al nodo 4:

`dist`	0	11	10	12	22	12

Considero il nodo a distanza minore non ancora completato.

Algoritmo di Dijkstra

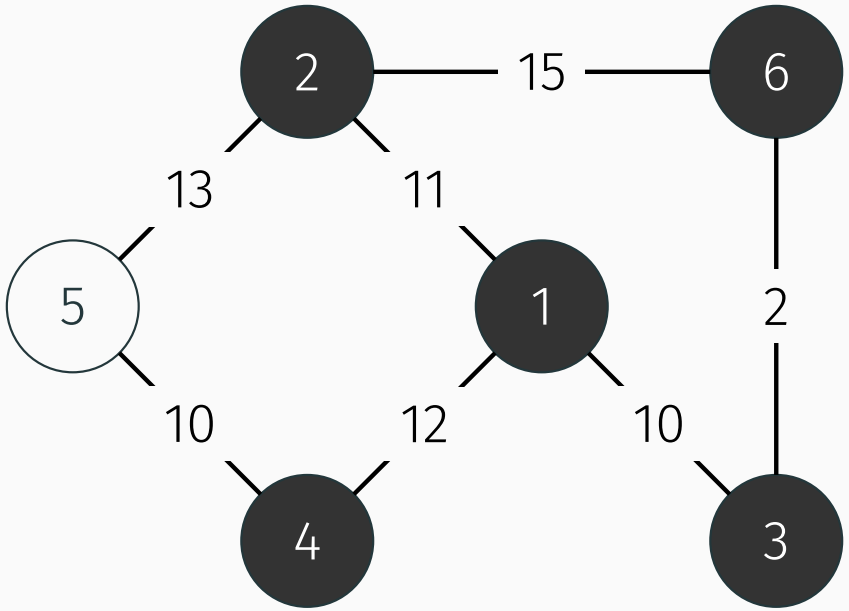


“Relax” degli adiacenti al nodo 6:

`dist`

0	11	10	12	22	12

Algoritmo di Dijkstra

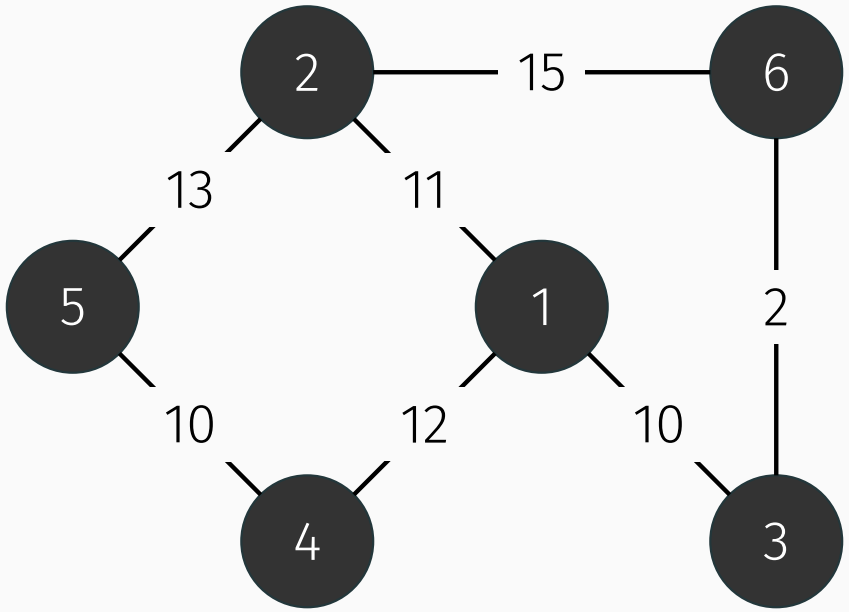


“Relax” degli adiacenti al nodo 6:

`dist`	0	11	10	12	22	12

Considero il nodo a distanza minore non ancora completato.

Algoritmo di Dijkstra



“Relax” degli adiacenti al nodo 5:

`dist`	0	11	10	12	22	12

Tutti i nodi sono stati completati e quindi le distanze sono tutte definitive.

Algoritmo di Dijkstra: complessità computazionale

Qual è la complessità dell'algoritmo di Dijkstra?

Ricordiamo che un grafo è espresso formalmente come $G = (V, E)$.

Algoritmo di Dijkstra: complessità computazionale

Qual è la complessità dell'algoritmo di Dijkstra?

Ricordiamo che un grafo è espresso formalmente come $G = (V, E)$.

Complessità computazionale dell'algoritmo di Dijkstra

Stiamo sicuramente esplorando tutti gli $|E|$ archi. Inoltre, per ciascuno dei $|V|$ vertici, dobbiamo effettuare una ricerca del minimo.

- Se tale ricerca è implementata in modo naïve, richiede tempo lineare: $\mathcal{O}(|E| + |V| \cdot |V|) = \mathcal{O}(|E| + |V|^2)$.

Algoritmo di Dijkstra: complessità computazionale

Qual è la complessità dell'algoritmo di Dijkstra?

Ricordiamo che un grafo è espresso formalmente come $G = (V, E)$.

Complessità computazionale dell'algoritmo di Dijkstra

Stiamo sicuramente esplorando tutti gli $|E|$ archi. Inoltre, per ciascuno dei $|V|$ vertici, dobbiamo effettuare una ricerca del minimo.

- Se tale ricerca è implementata in modo naïve, richiede tempo lineare: $\mathcal{O}(|E| + |V| \cdot |V|) = \mathcal{O}(|E| + |V|^2)$.
- Se tale ricerca è implementata in modo efficiente (ad esempio tramite heap), richiede tempo logaritmico: $\mathcal{O}(|E| + |V| \cdot \log |V|)$.

Tocca a voi!

Esercizio

Implementare una soluzione per il problema “ponti”.

Riferimenti

Questa presentazione e soluzione in C++ dell’esercizio:

`https://files.chiodini.org/OII_Territoriali_2019/`

- Piattaforma di allenamento con correttore e forum:
  `https://training.olinfo.it`
- Guida alle selezioni territoriali del prof. Bugatti:
  `http://www.imparando.net/sito/olimpiadi_di_informatica.htm`