Teoria dei grafi

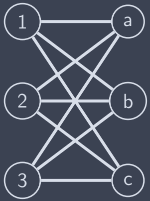
Introduzione e primi algoritmi

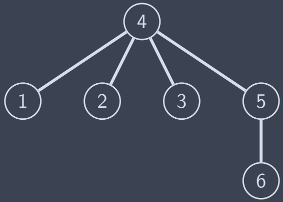
Lorenzo Ferrari, Davide Bartoli

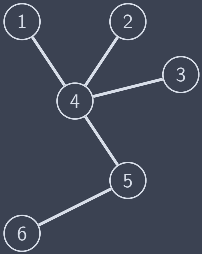
April 28, 2023

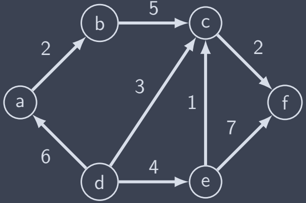
Table of contents

Esempi









Perchè studiamo i grafi?

- Tantissimi problemi possono essere ridotti a grafi
- I grafi sono bellissimi \(^-^)/

Perchè studiamo i grafi?

- Tantissimi problemi possono essere ridotti a grafi
- I grafi sono bellissimi `\(^-^)/`

Tutto è grafo. I grafi trovano applicazione nelle aree più disparate.

Esempi di problemi

- Data la descrizione di una città, trova il cammino più breve da A a B (o determina che è impossibile raggiungere B da A)
- Ci sono N città e conosci la distanza tra ogni coppia di città. Per costruire una strada che collega due città il costo è direttamente proporzionale alla distanza. Trova il costo minimo per connettere tutte e N le città.
- Date relazioni di dipendenza del tipo "il pacchetto x_i va installato prima del pacchetto y_i", trova un ordine per installare tutti i pacchetti o determina che è impossibile.

Grafo $G = (V, E)$

Un grafo è definito da due insiemi:

- V è l'insieme dei **nodi**/**vertici**
- E è l'insieme degli **archi**

Definizioni

Nodi e Archi

Nodo

- Talvolta chiamati anche *vertici*
- ogni nodo ha una label (un nome univoco)

Definizioni

Nodi e Archi

Nodo

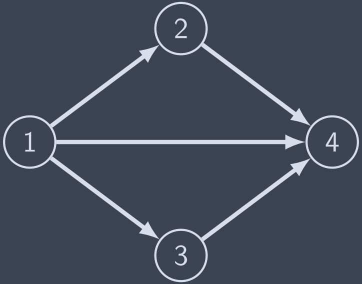
- Talvolta chiamati anche *vertici*
- ogni nodo ha una label (un nome univoco)

Arco

- Ogni arco è definito da una coppia di nodi
- Un arco connette i nodi che lo definiscono
- In alcuni casi, un arco puè connettere un nodo a se stesso
- In alcuni casi, un arco può avere un peso

Esempio

- $G = (V, E)$
- $V = \{1, 2, 3, 4\}$
- $E = \{(1, 2), (1, 3), (1, 4), (2, 4), (3, 4)\}$



Cammino

Un cammino di lunghezza n in un grafo $G = (V, E)$ è una sequenza di nodi $v_0, \ldots, v_n \in V$ tale che

$$(v_{i-1}, v_i) \in E \ \forall\ 1 \leq i \leq n.$$

Un cammino si dice *semplice* se tutti i v_i sono diversi uno dall'altro.

Definizioni

Cammini e Cicli

Cammino

Un cammino di lunghezza n in un grafo $G = (V, E)$ è una sequenza di nodi $v_0, \ldots, v_n \in V$ tale che

$$(v_{i-1}, v_i) \in E \;\forall\; 1 \leq i \leq n.$$

Un cammino si dice *semplice* se tutti i v_i sono diversi uno dall'altro.

Ciclo

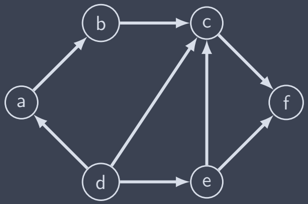
Un ciclo è un cammino in cui il primo e l'ultimo nodo coincidono.

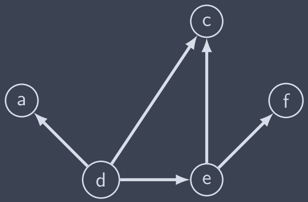
Definizioni

Sottografo

Sottografo

Un grafo $G' = (V', E')$ è sottografo di $G = (V, E)$ se e soltanto se $V' \subseteq V$ e $E' \subseteq E$.





Dimensioni di un grafo

Definizioni

- $n = |V|$: numero di nodi
- $m = |E|$: numero di archi

Dimensioni di un grafo

Definizioni

- $n = |V|$: numero di nodi
- $m = |E|$: numero di archi

Relaioni tra n e m

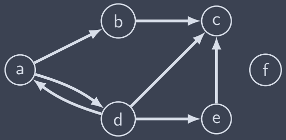
- In un grafo non diretto, $m \leq \frac{n(n-1)}{2} = O(n^2)$
- In un grafo diretto, $m \leq n^2 - n = O(n^2)$

Grafi diretti e non diretti

Grafi diretti $G = (V, E)$

- E è un insieme di coppie *ordinate* di nodi (u, v)

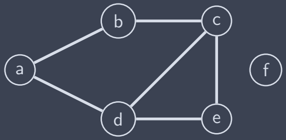
```
V = { a,b,c,d,e,f }
E = { (a,b),(a,d),(b,c),
      (d,a),(d,c),(d,e),
      (e,c) }
```



Grafi indiretti $G = (V, E)$

- E è un insieme di coppie *non ordinate* di nodi $[u, v]$

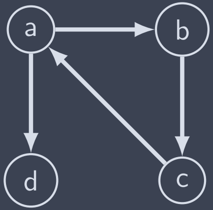
```
V = { a,b,c,d,e,f }
E = { [a,b],[a,d],[b,c],
      [c,d],[c,e],[e,d] }
```



Grafi ciclici e aciclici

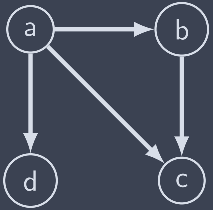
Grafo ciclico

- Contiene dei cicli



Grafo aciclico

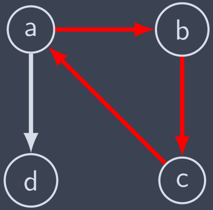
- Non contiene cicli



Grafi ciclici e aciclici

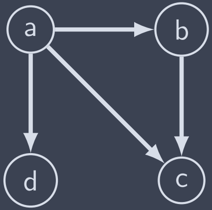
Grafo ciclico

- Contiene dei cicli



Grafo aciclico

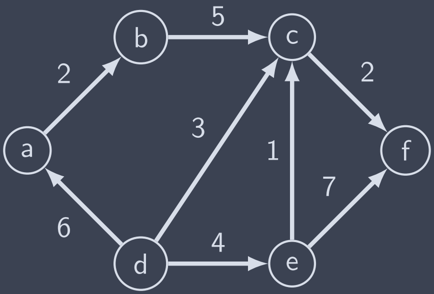
- Non contiene cicli



Grafi pesati

Grafo pesato

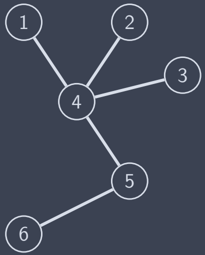
- a ogni arco è associato un *peso*
- generalmente il peso indica il costo di percorrere quell'arco



Alberi

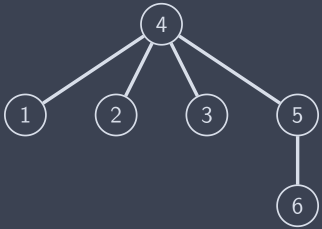
Albero

- Grafo *connesso* con $m = n - 1$



Albero radicato

- Grafo *connesso* con $m = n - 1$ e in cui un nodo è la **radice**



Caratteristica: un albero non presenta cicli.

Rappresentazioni

Ci sono due implementazioni “classiche”.

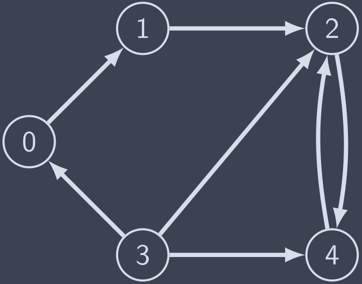
Rappresentazioni

Ci sono due implementazioni "classiche".

- Matrici di adiacenza
- Liste di adiacenza

Matrice di adiacenza

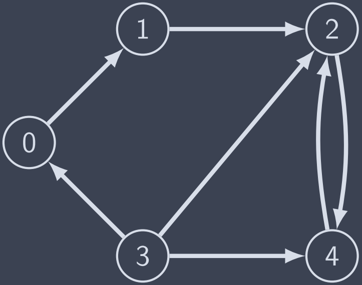
$$m_{uv} = \begin{cases} 1, & \text{se } (u,v) \in E \\ 0, & \text{se } (u,v) \notin E \end{cases}$$



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

Lista di adiacenza

$$G.adj(u) = \{v \ : \ (u, v) \in E\}$$



$0 \rightarrow 1$

$1 \rightarrow 2$

$2 \rightarrow 4$

$3 \rightarrow 0 \;|\; 2 \;|\; 4$

$4 \rightarrow 2$

Breadth-first search

Problema

Dato un grafo $G = (V, E)$ e un vertice $r \in V$ (radice), visitare esattamente una volta tutti i vertici del grafo che si possono raggiungere da r.

Breadth-first search (BFS)

Attraversa il grafo visitando i nodi per livelli: prima i nodi a distanza 1 dalla radice, poi a distanza 2, etc.
Alcune applicazioni:

- percorso minimo tra due nodi
- componenti connesse

Breadth-first search

```
vector<int> adj[100000];
vector<bool> vis(100000);

void bfs(int pos){
  // salvo i nodi in una coda
  queue<int> q;
  q.push(pos);
  while(!q.empty()){
    int cur=q.front();
    q.pop();
    // se ho già visitato il nodo mi fermo
    if(vis[cur])continue;
    vis[cur]=1;
    for(int x:adj[pos]){
      cur.push(x);
    }
  }
}
```

Depth-first search

Problema

Dato un grafo $G = (V, E)$ e un vertice $r \in V$ (radice), visitare esattamente una volta tutti i vertici del grafo che si possono raggiungere da r.

Depth-first search (DFS)

Attraversa il grafo visitando tutti i nodi raggiungibili da un nodo, poi tutti i noti raggiungibili da quei nodi... Applicazioni:

- topological sort
- trovare cicli
- componenti connesse
- componenti fortemente connesse

Depth-first search

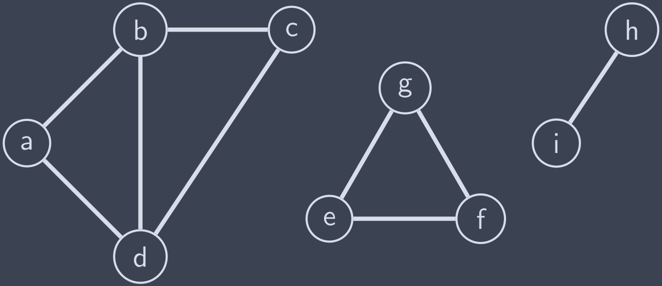
```
vector<int> adj[100000];
vector<bool> vis(100000);

void dfs(int pos){
  vis[pos]=1;
  // scorro tutti i vicini
  for(int x:adj[pos]){
    if(!vis[x]){
      // se non li ho ancora visitati
      // li visito
      dfs(x);
    }
  }
}
```

Componenti connesse

Problema

Dato un grafo $G = (V, E)$ non diretto, trovare il numero di componenti connesse.



Componenti connesse

Come possiamo contare il numero di componenti connesse?

Componenti connesse

Come possiamo contare il numero di componenti connesse?
Possiamo utilizzare una delle 2 visite viste prima (BFS e DFS).

Componenti connesse

Come possiamo contare il numero di componenti connesse?
Possiamo utilizzare una delle 2 visite viste prima (BFS e DFS).

- iteriamo su tutti i nodi, se non sono stati visitati, eseguiamo una visita

Componenti connesse

Come possiamo contare il numero di componenti connesse?
Possiamo utilizzare una delle 2 visite viste prima (BFS e DFS).

- iteriamo su tutti i nodi, se non sono stati visitati, eseguiamo una visita
- in questo modo visitiamo tutti i nodi della stessa componente connessa

Componenti connesse

Come possiamo contare il numero di componenti connesse?
Possiamo utilizzare una delle 2 visite viste prima (BFS e DFS).

- iteriamo su tutti i nodi, se non sono stati visitati, eseguiamo una visita
- in questo modo visitiamo tutti i nodi della stessa componente connessa
- il numero di visite effettuate è il numero di componenti connesse

Ponti

Dato un grafo di $N \leq 10\,000$ e $M \leq 20\,000$ archi, trova il minimo numero di archi da costruire per rendere il grafo connesso.

https://training.olinfo.it/#/task/ponti/statement

Problemi

Ponti

Ponti

Dato un grafo di $N \leq 10\,000$ e $M \leq 20\,000$ archi, trova il minimo numero di archi da costruire per rendere il grafo connesso.

https://training.olinfo.it/#/task/ponti/statement

- se $M = 0$ la risposta è $N - 1$

Ponti

Dato un grafo di $N \leq 10\,000$ e $M \leq 20\,000$ archi, trova il minimo numero di archi da costruire per rendere il grafo connesso.

https://training.olinfo.it/#/task/ponti/statement

- se $M = 0$ la risposta è $N - 1$
- possiamo considerare ogni componente connessa come un nodo

Problemi

Ponti

Ponti

Dato un grafo di $N \leq 10\,000$ e $M \leq 20\,000$ archi, trova il minimo numero di archi da costruire per rendere il grafo connesso.

https://training.olinfo.it/#/task/ponti/statement

- se $M = 0$ la risposta è $N - 1$
- possiamo considerare ogni componente connessa come un nodo
- se ci sono K componenti connesse, la risposta è $K - 1$

Problemi

Ponti

```cpp
vector<bool> vis;
vector<vector<int>> adj;

void dfs(int v) {
  vis[v] = true;
  for (int u : adj[v]) {
    if (!vis[u]) {
      dfs(u);
    }
  }
}
```

```cpp
int n; cin >> n;
int m; cin >> m;
adj.resize(n);
vis.assign(n, false);
for (int i = 0, a, b; i < m; ++i) {
    cin >> a >> b;
    adj[a].push_back(b);
    adj[b].push_back(a);
}

int ans = 0;
for (int i = 0; i < n; ++i) {
    if (!vis[i]) {
        ++ans;
        dfs(i);
    }
}

cout << ans << "\n";
```

Distanza minima

Problema

Dato un grafo $G = (V, E)$ non diretto, trovare la minima distanza tra due nodi a e b, ovvero il minimo numero di archi da attraversare per raggiungere b partendo da a.

Distanza minima

Problema

Dato un grafo $G = (V, E)$ non diretto, trovare la minima distanza tra due nodi a e b, ovvero il minimo numero di archi da attraversare per raggiungere b partendo da a.

- Possiamo utilizzare una BFS, dato che visita i nodi ordinati per distanza dal nodo di partenza.

Distanza minima

Problema

Dato un grafo $G = (V, E)$ non diretto, trovare la minima distanza tra due nodi a e b, ovvero il minimo numero di archi da attraversare per raggiungere b partendo da a.

- Possiamo utilizzare una BFS, dato che visita i nodi ordinati per distanza dal nodo di partenza.
- Nella coda della BFS teniamo il nodo attuale e la sua distanza da a.

Distanza minima

Problema

Dato un grafo $G = (V, E)$ non diretto, trovare la minima distanza tra due nodi a e b, ovvero il minimo numero di archi da attraversare per raggiungere b partendo da a.

- Possiamo utilizzare una BFS, dato che visita i nodi ordinati per distanza dal nodo di partenza.
- Nella coda della BFS teniamo il nodo attuale e la sua distanza da a.
- Quando troviamo b, abbiamo trovato la distanza minima.

Domande?

Problemi

https://cses.fi/problemset/
https://training.olinfo.it/#/task/ponti/statement
https://training.olinfo.it/#/task/tecla/statement
https://training.olinfo.it/#/task/sentieri/statement