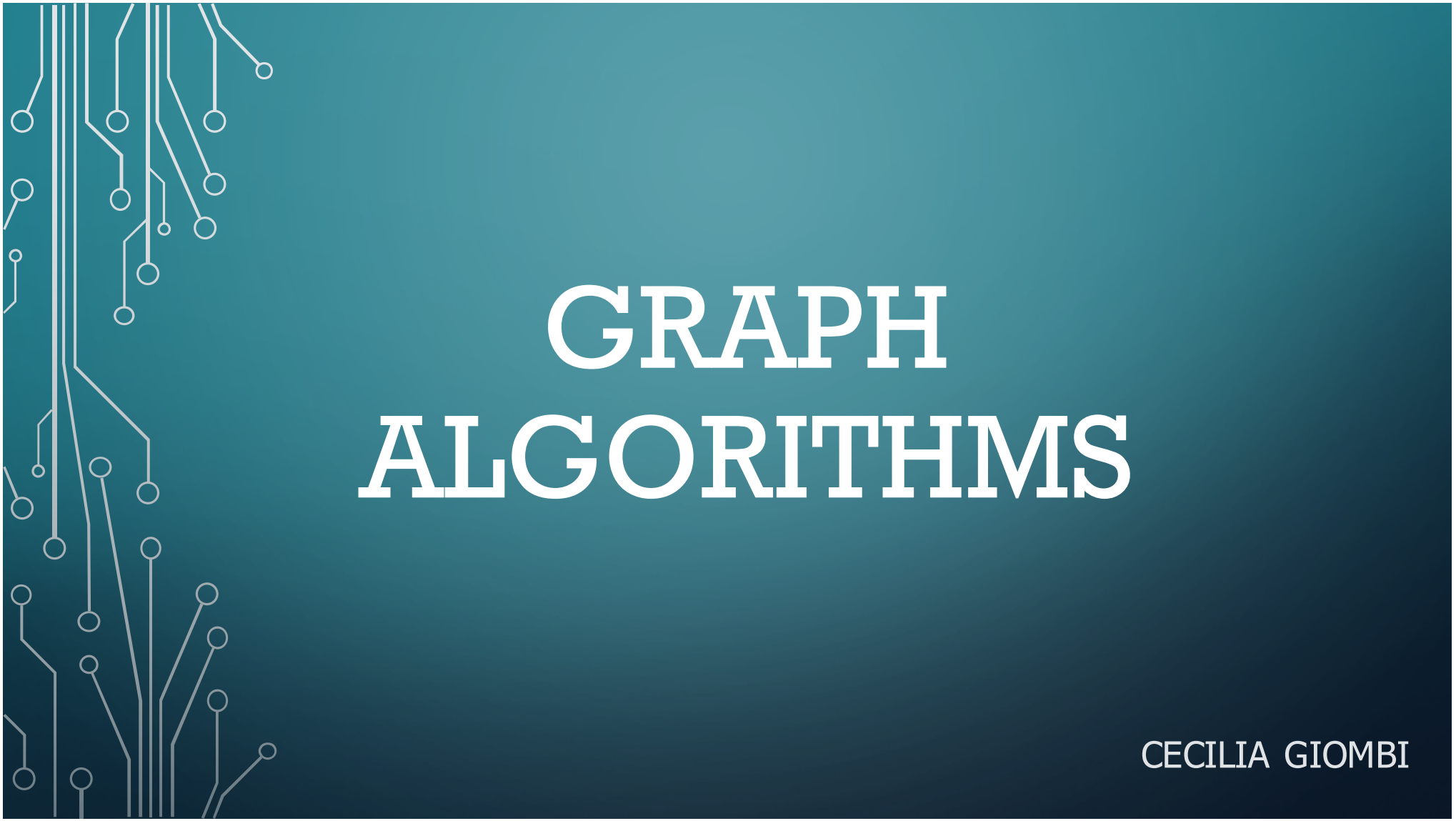

GRAPH
ALGORITHMS
CECILIA GIOMBI

IL PROBLEMA

1. Visita di un grafo
2. Ricerca del cammino di costo minimo
3. Equidistribuzione delle risorse scarse

LA SOLUZIONE è la progettazione di ALGORITMI

procedura costituita da una sequenza finita di operazioni elementari che, iterate una o più volte, consentono di produrre la soluzione ad un determinato problema.

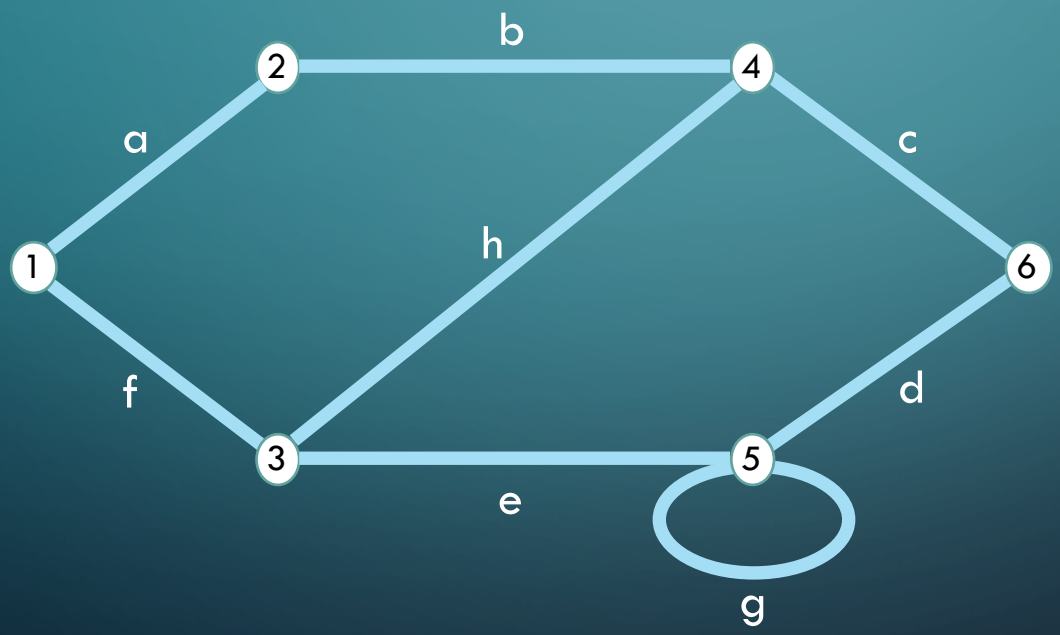
DEFINIZIONI

- Un GRAFO G è una coppia di insiemi di vertici e lati ovvero $G = (V, E)$ dove V è l'insieme dei vertici ed E è l'insieme dei lati.
- Un GRAFO ORIENTATO o DIGRAFO D è una terna $D = (V(D), E(D), \varphi_D)$ dove V è l'insieme dei vertici, E è l'insieme degli archi e φ_D è una funzione di incidenza che associa ad ogni arco e una coppia ordinata di vertici ovvero $\varphi_D(e) = (u, v)\ \forall e \in E(D)$.
- Nei grafi orientati ha senso definire il vertice chiamato SORGENTE ovvero quel vertice privo di archi entranti e il POZZO cioè il vertice privo di archi uscenti.

Un grafo, orientato o non, può essere rappresentato utilizzando una matrice o una lista di adiacenza.

Esempio.

Consideriamo il grafo G in figura.



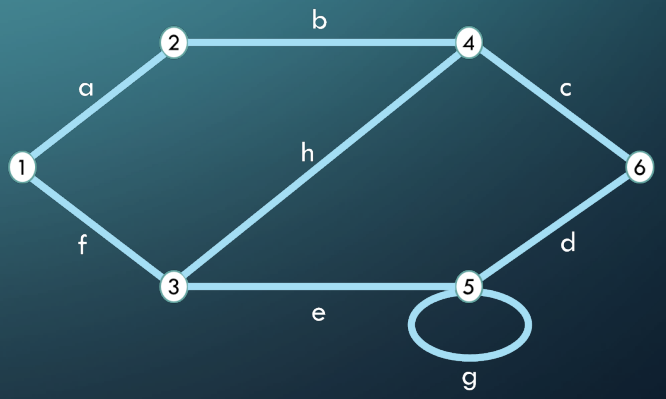
$$V(G) = \{1,2,3,4,5,6\}$$
$$E(G) = \{a, b, c, d, e, f, g, h\}$$

Rappresentiamo il grafo dell'esempio utilizzando la matrice di adiacenza M di ordine $n = |V|$ definita nel seguente modo:

$$M_{i,j} = \begin{cases} 1 & se\ (v_i, v_j) \in E(G) \\ 0 & altrimenti \end{cases}$$

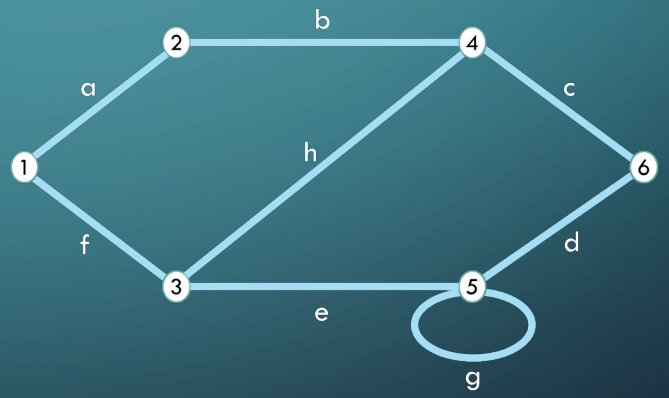
Se il grafo è non orientato per definizione risulta $(v_i, v_j) \in E(G) \iff (v_j, v_i) \in E(G)$ pertanto la matrice di adiacenza sarà simmetrica rispetto alla diagonale principale (come in questo esempio).

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$



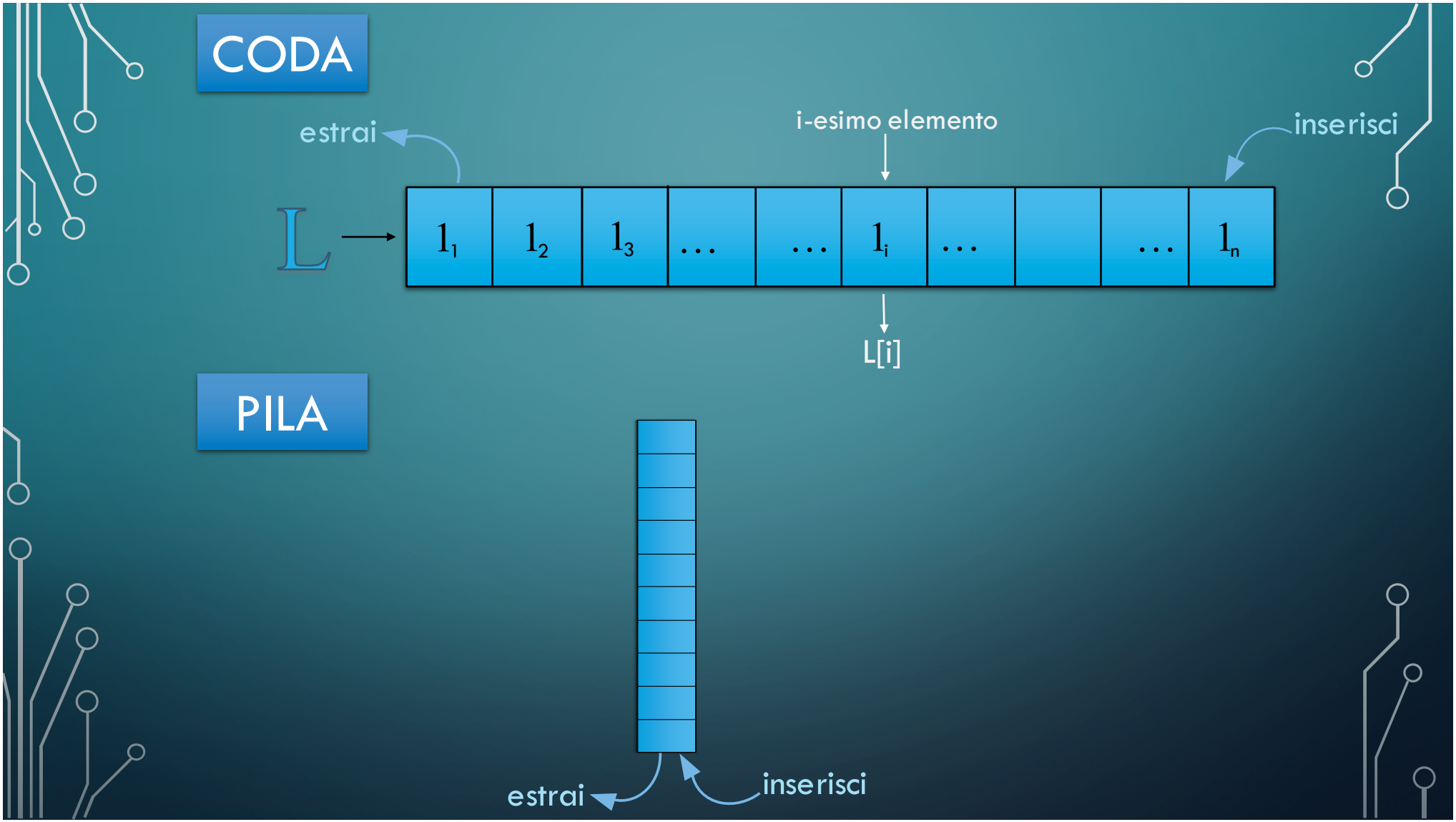
Rappresentiamo invece ora il grafo dell'esempio utilizzando una famiglia di n liste $L_1, L_2, \dots, L_n$ di vertici, denominate liste di adiacenza del grafo G.
Per ogni vertice $v_i \in V(G)$, vengono memorizzati i suoi vertici adiacenti nella lista L_i, in altre parole $L_i = N(v_i)$ dove con $N(v_i)$ indico i vertici collegati a v_i.

$L_1 = [2,3]$
$L_2 = [1,4]$
$L_3 = [1,4,5]$
$L_4 = [2,3,6]$
$L_5 = [3,5,6]$
$L_6 = [4,5]$



	Matrice di adiacenza	Lista di adiacenza
Grafo orientato	$\|V\|^2$	$\|E\|$
Grafo non orientato	$\|V\|$	$2\|E\|$

Osserviamo inoltre che le liste di adiacenze hanno sempre una complessità $\theta(V + E)$ sia per i grafi orientati che per i grafi non orientati.


CODA
estrai
i-esimo elemento
inserisci
L
l_1
l_2
l_3
...
...
l_i
...
...
l_n
L[i]
PILA
estrai
inserisci

ALGORITMI

1. Breadth-first-search (BFS)
2. Depth-first-search (DFS)
3. Dijkstra
4. Bellamn-Ford
5. Floyd-Roy-Warshall
6. accenno Johnson

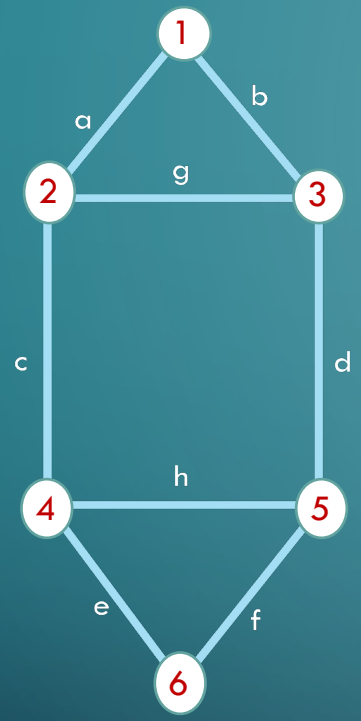
1) BREADTH FIRST SEARCH (BFS)

Input: Il grafo G, orientato o non, ed un vertice sorgente $s \in V(G)$
Output: Un albero T con radice in s e una lista D con le distanza di tutti i vertici da s

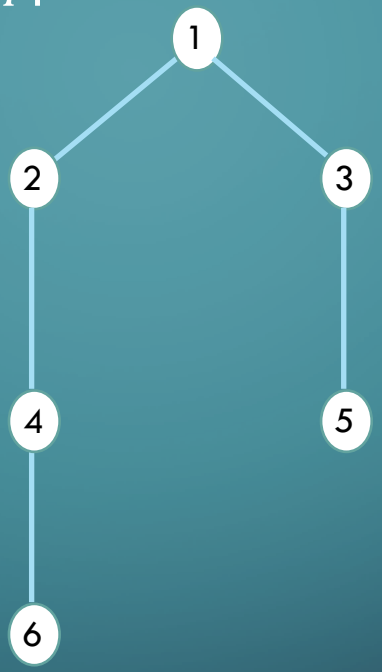
1: sia Q la coda $Q = \{s\}, T = (\{s\}, \emptyset)$
2: sia $D = [\infty, \infty, \dots, \infty]$
3: $D[s] = 0$
4: fintanto che $Q \neq \emptyset$ ripeti:
5: estrai v da Q e marcalo
6: $\forall u \epsilon N(v)$ ripeti:
7: se u non è marcato e $D[u] = \infty$
8: allora aggiungi u a Q e marcalo, $D[u] = D[v] + 1$,
$V(T) = V(T) \cup \{u\},\ E(T) = E(T) \cup \{(vu)\}$
9: fine ciclo
10: fine condizione
11: restituisci D e T

$O(n + m)$

Esempio.



T:



1: sia Q coda $Q = \{s\}, T = (\{s\}, \emptyset)$
2: sia $D = [\infty, \infty, \dots, \infty]$
3: $D[s] = 0$
4: fintanto che $Q \neq \emptyset$ ripeti:
5: estrai v da Q e marcalo
6: $\forall u \epsilon N(v)$ ripeti:
7: se u non è marcato e $D[u] = \infty$
8: allora aggiungi u a Q e marcalo,
$D[u] = D[v] + 1$,
$V(T) = V(T) \cup \{u\}$,
$E(T) = E(T) \cup \{(vu)\}$
9: fine ciclo
10: fine condizione
11: restituisci D e T

$Q = \{\cancel{1},\cancel{2},\cancel{3},\cancel{4},\cancel{5},\cancel{6}\}$

$D = [0, \underset{1}{\cancel{\infty}}, \underset{1}{\cancel{\infty}}, \underset{2}{\cancel{\infty}}, \underset{2}{\cancel{\infty}}, \underset{3}{\cancel{\infty}}]$

❖ ***Teorema.***

Nel caso peggiore la complessità dell'algoritmo è $O(|V| + |E|)$.

Dim. Assumiamo che il grafo $G = (V, E)$ sia connesso.

La fase di inizializzazione (1-3), essendo ripetuta per tutti i vertici del grafo, ha una complessità di $O(|V|)$. Poiché ogni vertice entra nella coda Q una sola volta, il ciclo esterno (4-10) esegue al massimo n iterazioni, dove $n = |V|$.

Per ogni iterazione del ciclo esterno si eseguono tante iterazioni del ciclo più interno (6-9) quanti sono gli spigoli incidenti nel vertice che si sta esaminando, che saranno al massino m.

Andando a sommare le lunghezze di tutte le liste di adiacenza, si ottiene una complessità temporale di $O(|E|)$.

Quindi sommando le operazioni del ciclo esterno e di quello interno, si ottiene che la complessità dell'algoritmo è $O(|V| + |E|)$.

1: sia Q coda $Q = \{s\}, T = (\{s\}, \emptyset)$
2: sia $D = [\infty, \infty, \dots, \infty]$
3: $D[s] = 0$
4: fintanto che $Q \neq \emptyset$ ripeti:
5: estrai v da Q e marcalo
6: $\forall u \epsilon N(v)$ ripeti:
7: se u non è marcato e $D[u] = \infty$
8: allora aggiungi u a Q e marcalo, $D[u] = D[v] + 1$,
$V(T) = V(T) \cup \{u\}, \; E(T) = E(T) \cup \{(vu)\}$
9: fine ciclo
10: fine condizione
11: restituisci D e T

Nella dimostrazione seguente useremo $d(v,u)$ per indicare la lunghezza del cammino più corto da v ad u che è anche conosciuta come distanza da v ad u.

❖ ***Teorema***.

Sia D la lista delle distanze prodotte dall'algoritmo, sia s la sorgente e v un vertice tale che $D[v] \neq \infty$. Allora $D[v]$ è la lunghezza del cammino più corto da s a v.

Dim. Si ha che $D[v] = \infty$ se e solo se non ci sono cammini che collegano s a v.

Perciò se $D[v] \neq \infty$, allora v è raggiungibile da s da un cammino di lunghezza $D[v]$, la lunghezza $d(s,v)$ di ogni cammino più corto $s - v$ soddisfa che $d(s,v) \leq D(v)$.

Usando l'induzione su $d(s,v)$ si mostra che l'uguaglianza è verificata.

Per il caso base $s = v$, si ha che $d(s,v) = D(v) = 0$ poiché il cammino banale ha lunghezza 0. Assumiamo per induzione che se $d(s,v) = k$, allora $d(s,v) = D(v)$.

Sia $d(s,u) = k + 1$, essendo il corrispondente cammino $s - u$ più corto $(s, v_1, v_2, \dots, v_k, u)$. Dall'ipotesi di induzione, $(s, v_1, v_2, \dots, v_k)$ è il cammino più corto da s a v_k di lunghezza $d(s, v_k) = D[v_k] = k$. Ovvero $D[v_k] < D[u]$.

Quando si processa la lista di adiacenza di v_k, l'algoritmo raggiunge u e quindi $D[u] \leq k + 1$ e da sopra $D[u] > D[v_k] = k$, quindi $D[u] = k + 1$ e quindi $d(s,u) = D(u) = k + 1$.

2) DEPTH FIRST SEARCH (DFS)

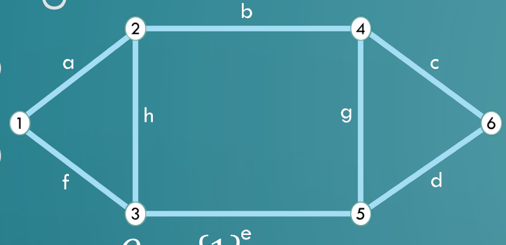
Input: Il grafo G, orientato o non, ed un vertice sorgente $s \in V(G)$
Output: Un albero T con radice in s e una lista D con le distanza di tutti i vertici da s

1: sia Q la pila $Q = \{s\}, T = (\{s\}, \emptyset)$
2: sia $D = [\infty, \infty, \dots, \infty]$
3: $D[s] = 0$
4: fintanto che $Q \neq \emptyset$ ripeti:
5: estrai v da Q e marcalo
6: $\forall u \epsilon N(v)$ ripeti:
7: se u non è marcato e $D[u] = \infty$
8: allora aggiungi u a Q, $D[u] = D[v] + 1$,
$V(T) = V(T) \cup \{u\},\ E(T) = E(T) \cup \{(vu)\}$
9: fine ciclo
10: fine condizione
11: restituisci D e T

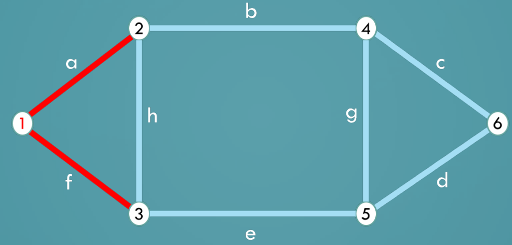
$O(n + m)$

Esempio.

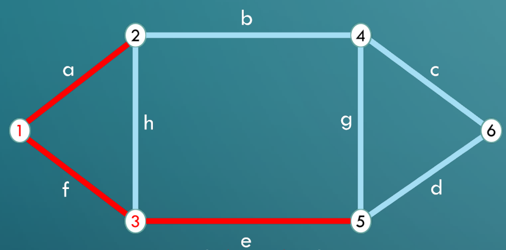
sia Q la pila $Q = \{s\}, T = (\{s\}, \emptyset)$
sia $D = [\infty, \infty, \dots, \infty]$
$D[s] = 0$
fintanto che $Q \neq \emptyset$ ripeti:
    estrai v da Q e marcalo
    $\forall u \epsilon N(v)$ ripeti:
        se u non è marcato e $D[u] = \infty$
        allora aggiungi u a Q, $D[u] = D[v] + 1$,
        $V(T) = V(T) \cup \{u\},\ E(T) = E(T) \cup \{(vu)\}$
    fine ciclo
fine condizione
restituisci D e T



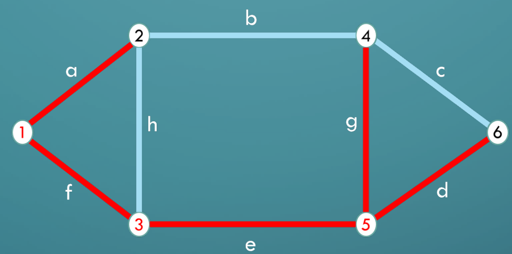
$Q = \{1\}$
$D = [0, \infty, \infty, \infty, \infty, \infty]$



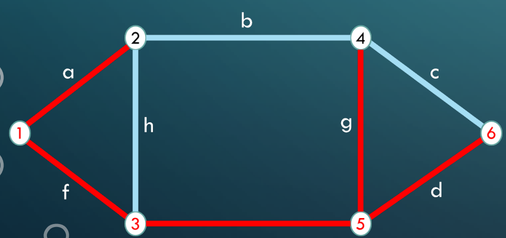
$Q = \{\not{1},2,3\}$
$D = [0,1,1, \infty, \infty, \infty]$



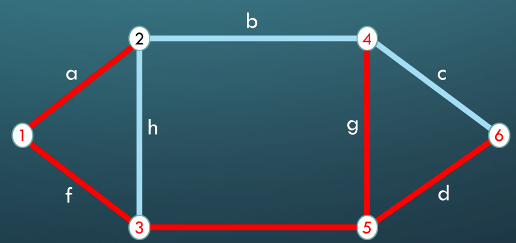
$Q = \{\not{1},2,\not{3},5\}$
$D = [0,1,1, \infty, 2, \infty]$



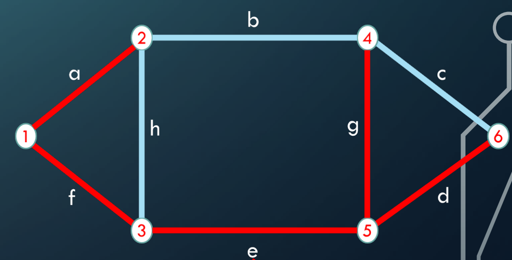
$Q = \{\not{1},2,\not{3},\not{5},4,6\}$
$D = [0,1,1,2,3,3]$



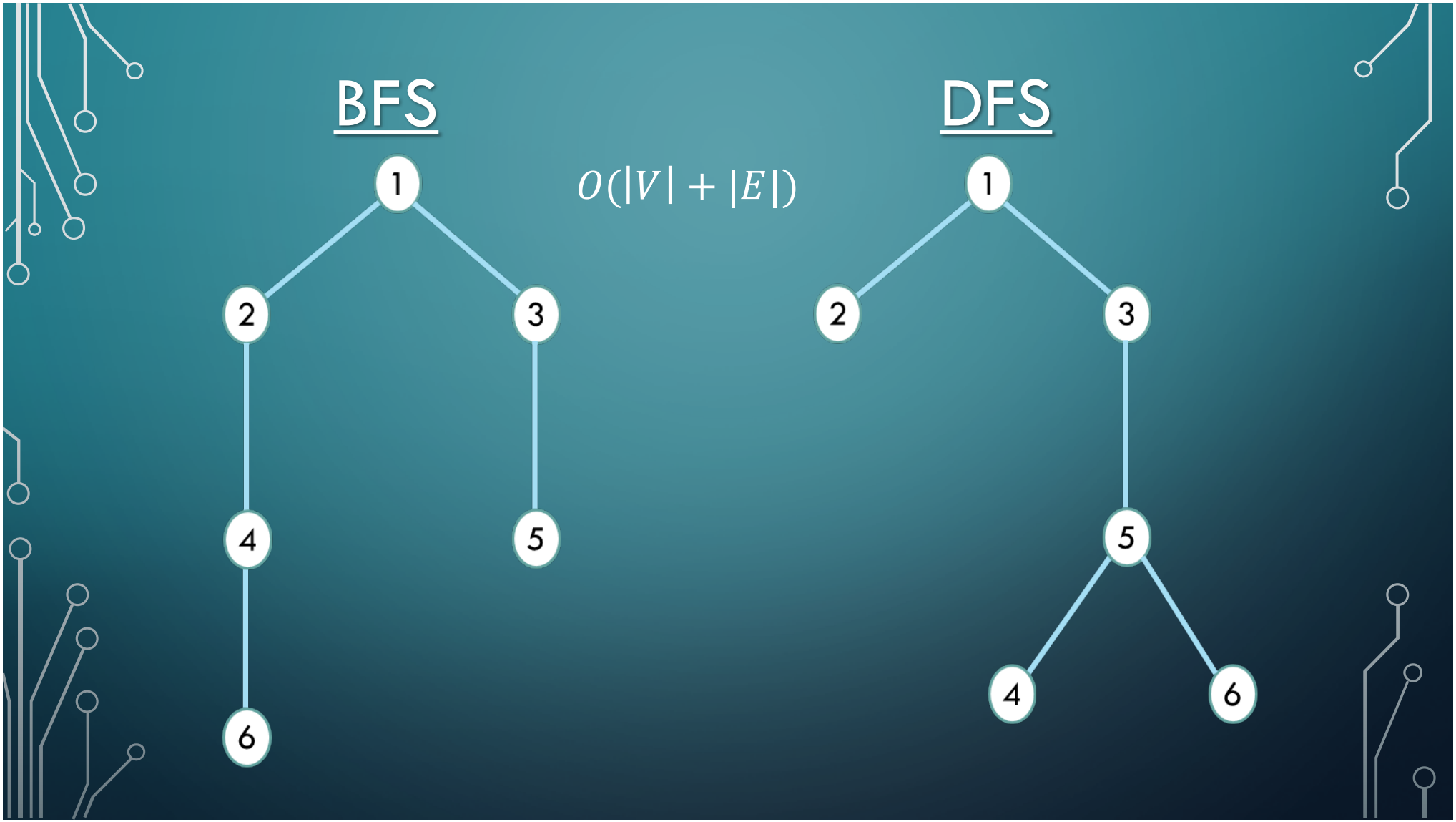
$Q = \{\not{1},2,\not{3},\not{5},4,\not{6}\}$
$D = [0,1,1,2,3,3]$



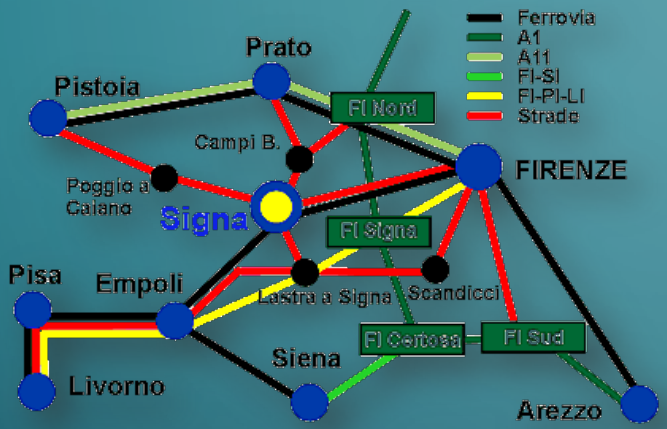
$Q = \{\not{1},2,\not{3},\not{5},\not{4},\not{6}\}$
$D = [0,1,1,2,3,3]$

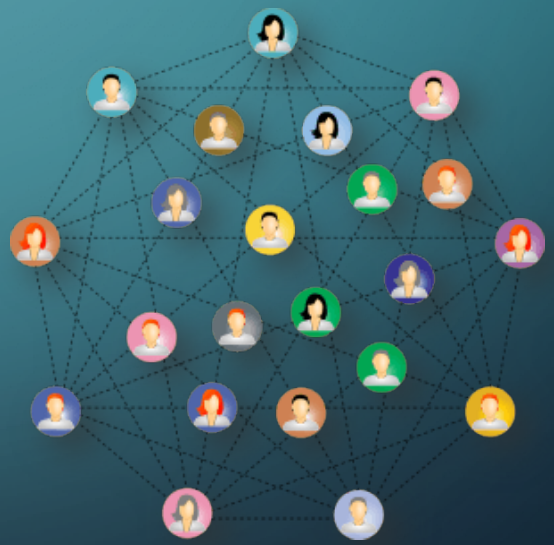


$Q = \{\not{1},\not{2},\not{3},\not{5},\not{4},\not{6}\}$
$D = [0,1,1,2,3,3]$


BFS
1
2
3
4
5
6
$O(|V| + |E|)$
DFS
1
2
3
5
4
6

CAMMINI DI COSTO MINIMO





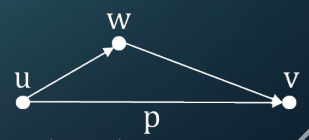
DEFINIZIONI

- Un GRAFO PESATO è un grafo $G = (V, E)$ dove si assegna un peso w a ciascun lato di G.
- Il COSTO DEL CAMMINO p: u $\rightsquigarrow v$ è la somma dei costi degli spigoli che lo compongono:

$$W(p) = \sum_{(v_i, v_{i+1}) \in p} w(v_i, v_{i+1})$$

- Si indichi con $d(u, v)$ il costo del cammino di costo minimo da u a v, che non è necessariamente il cammino più breve, cioè $d(u, v) = minW(p)$.
  La funzione distanza d su G è definita nel seguente modo:

$$d(u, v) = \begin{cases} 0 & \text{se u = v} \\ \infty & \text{se non ci sono cammini da u a v} \\ \min\{w(p)\} & \text{altrimenti} \end{cases}$$



e soddisfa le seguenti proprietà: 1. simmetria $d(u, v) = d(v, u)$
2. disuguaglianza triangolare $d(u, w) + d(w, v) \geq d(u, v)$

- La matrice delle distanze D è definita come $D = [d(v_i, v_j)]$.

Lemma.

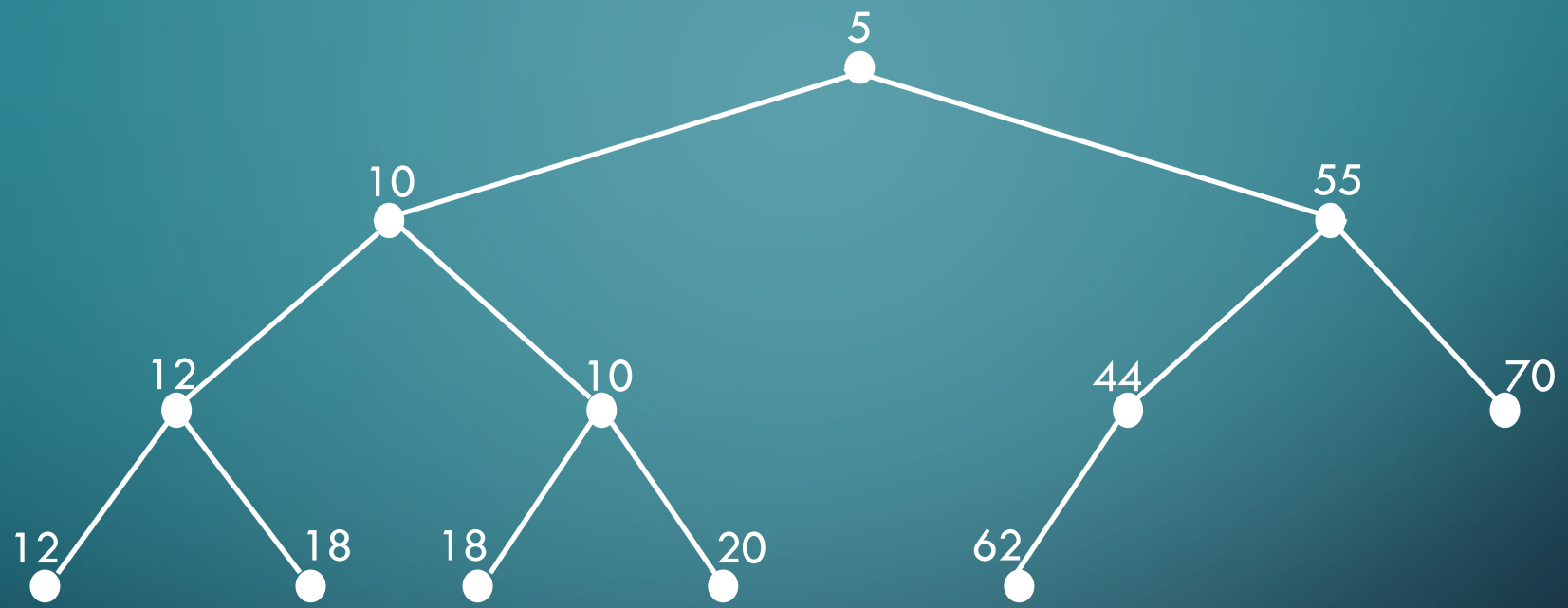
Dato il grafo connesso $G = (V, E)$ di ordine $n = |V|$ e fissato un vertice v, se non ci sono cicli di costo negativo in G, allora esiste un cammino minimo da v ad un qualunque altro vertice $u \in V$ costituito al massimo da $n - 1$ lati.

Dim. Sia P il cammino costituito al massimo da n-1lati da v ad un qualunque altro vertice u.

$$P: v_0 = v, v_1, \dots, v_k = u$$

Sia P' lo stesso cammino di P a cui sono stati tolti tutti i cicli presenti in P ma, poiché P non ha cicli di costo negativo per ipotesi, il costo di P è non minore del costo di P'. Quindi è possibile rimuovere tutti i cicli da P e ottenere un cammino P' da v a u di peso minore. Poiché il cammino finale è aciclico, esso deve avere non più di n-1 lati.

Esempio. È rappresentato in figura un heap e l'array corrispondente.



5	10	55	12	10	44	70	12	18	18	20	62	

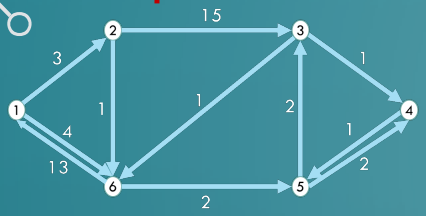
3) DIJKSTRA

Input: Il grafo G, orientato o non, ed un vertice sorgente $s \in V(G)$
Output: La lista D con la stima dei costi dei cammini e la lista P dei predecessori

1: $\forall v \epsilon V(G)$ ripeti:
2: $\quad D[v] = \infty, \pi(v) = \text{null}$
3: fine ciclo
4: $D[s] = 0$
5: sia Q la coda di priorità contenente tutti i vertici v ciascuno con priorità $D[v]$
6: fintanto che $Q \neq \emptyset$ ripeti:
7: $\quad$ sia u il vertice estratto da Q con priorità $D[u]$ minima
8: $\quad \forall v \epsilon N(u)$ ripeti:
9: $\quad\quad$ se $D[v] > D[u] + w(u, v)$
10: $\quad\quad$ allora $D[v] = D[u] + w(u, v), \pi(v) = u$
11 $\quad$ fine ciclo
12: fine condizione
13: restituisci D e P

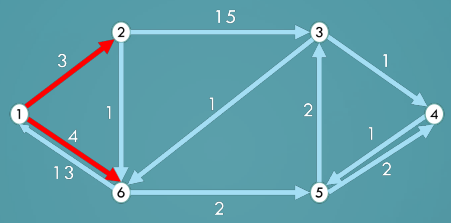
$O(mlog(n))$

Esempio.



$Q = \{1,2,3,4,5,6\}$

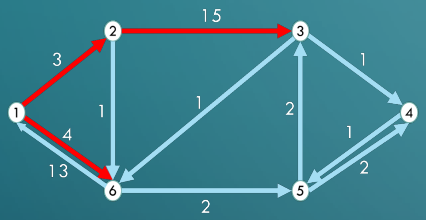
$D = [0, \infty, \infty, \infty, \infty, \infty]$



$Q = \{\cancel{1},2,3,4,5,6\}$

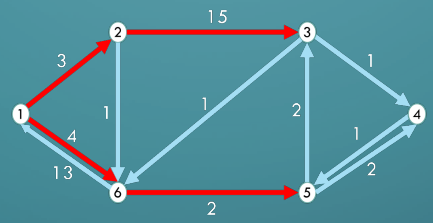
$D = [0, \cancel{\infty}, \infty, \infty, \infty, \cancel{\infty}]$

3 4



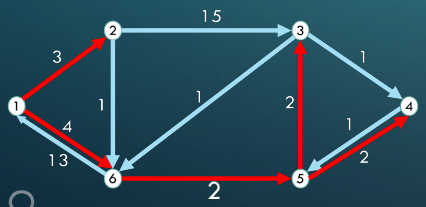
$Q = \{\cancel{2},6,3,4,5\}$

$D = [3,4,18, \infty, \infty]$



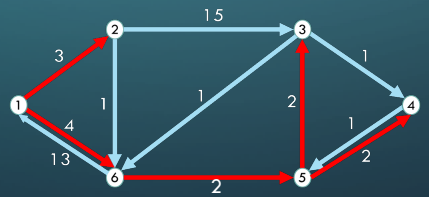
$Q = \{\cancel{6},3,4,5\}$

$D = [4,18, \infty, 6]$



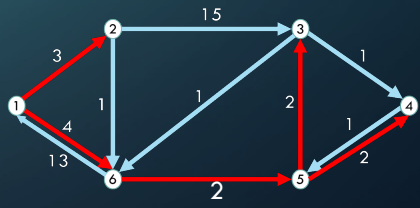
$Q = \{\cancel{5},3,4\}$

$D = [6,18, \infty]$



$Q = \{\cancel{3},4\}$

$D = [8,8]$



$Q = \{\cancel{4}\}$

$D = [8]$

1: $\forall v \epsilon V(G)$ ripeti:
2: $\mathrm{D}[\mathrm{v}] = \infty, \pi(\mathrm{v}) = \mathrm{null}$
3: fine ciclo
4: $D[s] = 0$
5: sia Q la coda di priorità contenente tutti i vertici v ciascuno con priorità $\mathrm{D}[\mathrm{v}]$
6: fintanto che $Q \neq \emptyset$ ripeti:
7: sia u il vertice estratto da Q con priorità $\mathrm{D}[\mathrm{u}]$ minima
8: $\forall v \epsilon N(u)$ ripeti:
9: se $\mathrm{D}[\mathrm{v}] > \mathrm{D}[\mathrm{u}] + w(u,v)$
10: allora $\mathrm{D}[\mathrm{v}] = \mathrm{D}[\mathrm{u}] + w(u,v), \pi(\mathrm{v}) = \mathrm{u}$

11 fine ciclo
12: fine condizione
13: restituisci D e T

4) BELLMAN-FORD

Input: Il grafo G, orientato o non, ed un vertice sorgente $s \in V(G)$

Output: La lista D con la stima dei costi dei cammini e la lista P dei predecessori

1: $\forall v \epsilon V(G)$ ripeti:

2: $\quad \mathrm{D[v]} = \infty, \pi(\mathrm{v}) = \mathrm{null}$

3: fine ciclo

4: $D[s] = 0$

5: per $i = 1, \dots, n-1$ ripeti:

6: $\quad \forall (u, v) \epsilon E$ ripeti:

7: $\quad\quad$ se $\mathrm{D[v]} > \mathrm{D[u]} + w(u, v)$

8: $\quad\quad$ allora $\mathrm{D[v]} = \mathrm{D[u]} + w(u, v), \pi(\mathrm{v}) = \mathrm{u}$

9: $\quad$ fine ciclo

10: fine ciclo

11 $\forall (u, v) \epsilon E$ ripeti:

12: $\quad$ se $\mathrm{D[v]} > \mathrm{D[u]} + w(u, v)$

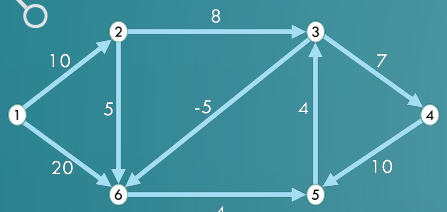
13: $\quad$ allora c'è un ciclo di costo negativo!

14: fine ciclo

15:restituisci D e P

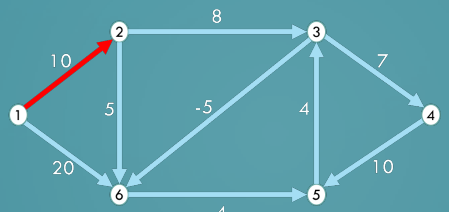
$O(nm)$

Esempio.



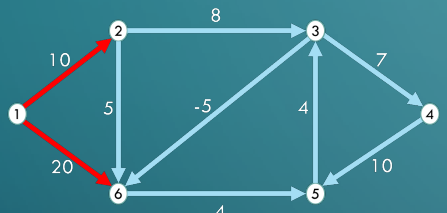
$D = [0, \infty, \infty, \infty, \infty, \infty]$

$i = 1 \quad | \quad (1-2)$



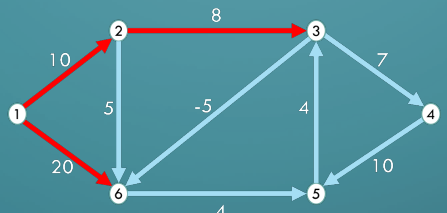
$D = [0,10, \infty, \infty, \infty, \infty]$

$i = 1 \quad | \quad (1-6)$



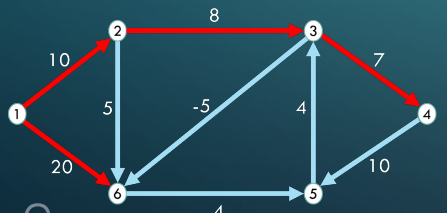
$D = [0,10, \infty, \infty, \infty, 20]$

$i = 1 \quad | \quad (2-3)$



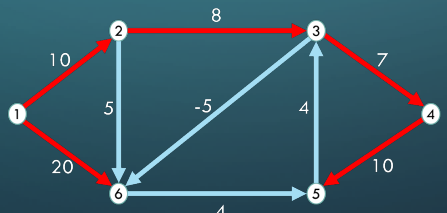
$D = [0,10,18, \infty, \infty, 20]$

$i = 1 \quad | \quad (3-4)$



$D = [0,10,18,25, \infty, 20]$

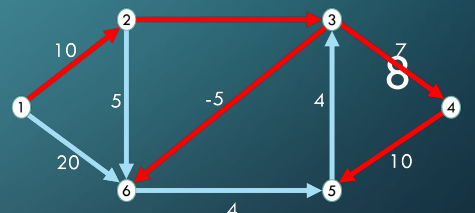
$i = 1 \quad | \quad (4-5)$



$D = [0,10,18,25,35,20]$

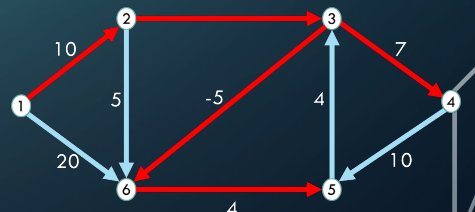
$i = 1 \quad | \quad (5-3)$

1: $\forall v \epsilon V(G)$ ripeti:
2: $\mathrm{D[v]} = \infty, \pi(\mathrm{v}) = \mathrm{null}$
3: fine ciclo
4: $D[s] = 0$
5: per $i = 1, \dots, n-1$ ripeti:
6: $\forall (u,v) \epsilon E$ ripeti:
7: se $\mathrm{D[v]} > \mathrm{D[u]} + w(u,v)$
8: allora $\mathrm{D[v]} = \mathrm{D[u]} + w(u,v), \pi(v) = \mathrm{u}$
9: fine ciclo
10: fine ciclo
11 $\forall (u,v) \epsilon E$ ripeti:
12: se $\mathrm{D[v]} > \mathrm{D[u]} + w(u,v)$
13: allora c'è un ciclo di costo negativo!
14: fine ciclo
15: restituisci D e P



$D = [0,10,18,25,35,13]$

$i = 1 \quad | \quad (3-6)$



$D = [0,10,18,25,17,13]$

$i = 1 \quad | \quad (6-5)$

PROBLEMA: Cammini di costo minimo da sorgente singola

DIJKSTRA	BELLMAN-FORD
• Si può applicare solamente a (di)grafi con pesi positivi	• Si può applicare a (di)grafi con pesi sia positivi sia negativi
• Usa coda di priorità Q	• È più semplice da codificare perché non usa strutture dati di difficile implementazione
• Complessità minore: $O(mlog(n))$	• Complessità maggiore: $O(nm)$

PROGRAMMAZIONE DINAMICA

- La matrice dei pesi W è definita nel seguente modo:

$$W_{(i.j)} = \begin{cases} 0 & \text{se i} = \text{j} \\ w(i,j) & \text{se } (i,j) \in \text{E(G)} \\ \infty & \text{altrimenti} \end{cases}$$

- Il costo del cammino p dal vertice i a j con vertici intermedi in $\{1,2,\dots,k\}$ è definita nel seguente modo:

$$D_k(i,j) = \begin{cases} w(i,j) & \text{se } k = 0 \\ \min\{D_{k-1}(i,j), D_{k-1}(i,k) + D_{k-1}(k,j)\} & \text{se } k > 0 \end{cases}$$

- La matrice dei predecessori Π è inizialmente definita nel modo seguente:

$$\pi^{(0)}(i,j) = \begin{cases} i & \text{se } (i,j) \in \text{E(G)} \\ \text{null} & \text{altrimenti} \end{cases}$$

5) Floyd-Roy-Warshall

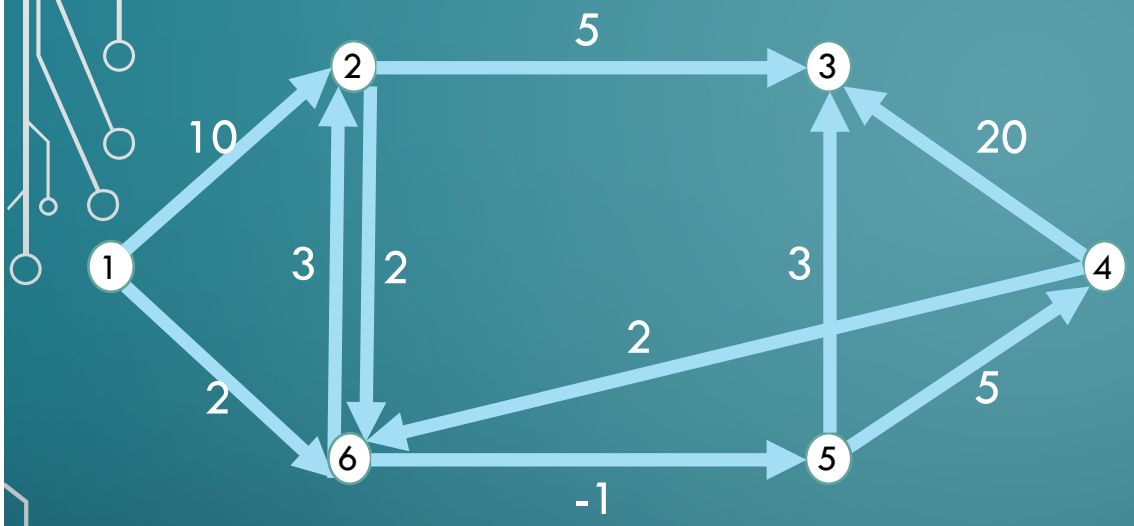
Input: Il grafo orientato e pesato G

Output: La matrice Π dei predecessori e la D dei costi dei cammini

1: $D^{(0)} = W,\ \Pi^{(0)} = 0$
2: per $k = 1, \dots, n$ ripeti:
3: per $i = 1, \dots, n$ ripeti:
4: per $j = 1, \dots, n$ ripeti:
5: se $D[i,j] > D[i,k] + D[k,j]$
6: allora $D[i,j] = D[i,k] + D[k,j],\ \ \pi(i,j) = k$
7: fine ciclo
8: fine ciclo
9: fine ciclo
10: restituisci D e Π

$O(n^3)$

Esempio.



1: $D^{(0)} = W,\ P^{(0)} = 0$
2: per k $= 1, \ldots, n$ ripeti:
3: per i $= 1, \ldots, n$ ripeti:
4: per j $= 1, \ldots, n$ ripeti:
5: se $D[i,j] > D[i,k] + D[k,j]$
6: allora $D[i,j] = D[i,k] + D[k,j]$, $\pi(i,j) = k$
7: fine ciclo
8: fine ciclo
9: fine ciclo
10: restituisci D e Π

$$D^{(0)} = W = \begin{bmatrix} 0 & 10 & 0 & 0 & 0 & 2 \\ 0 & 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 2 \\ 0 & 0 & 3 & 5 & 0 & 0 \\ 0 & 3 & 0 & 0 & -1 & 0 \end{bmatrix}$$

1-5-3
1-6-5-3
1-6-5-3

$$\Pi^{(6)} = \begin{bmatrix} 1 & 6 & 5 & 5 & 6 & 1 \\ / & 2 & 5 & 5 & 6 & 2 \\ / & / & 3 & / & / & / \\ / & 6 & 5 & 4 & 6 & 4 \\ / & 6 & 5 & 5 & 5 & 4 \\ / & 6 & 5 & 5 & 6 & 6 \end{bmatrix}$$

BIBLIOGRAFIA

- *Algorithmic Graph Theory and Sage* di D. Joyner, M. V. Nguyen, D. Phillips, i.e. capitolo 2

SITOGRAFIA

- *Cammini di costo minimo: problemi e algoritmi* di M. Liverani
- *Cammini minimi: algoritmi di Bellman-Ford e di Dijkstra* di L. De Giovanni
- *Algoritmo di Johnson per grafi sparsi* di D. Cantone

FINE

Grazie dell'attenzione