

Grafi, visite e cammini minimi

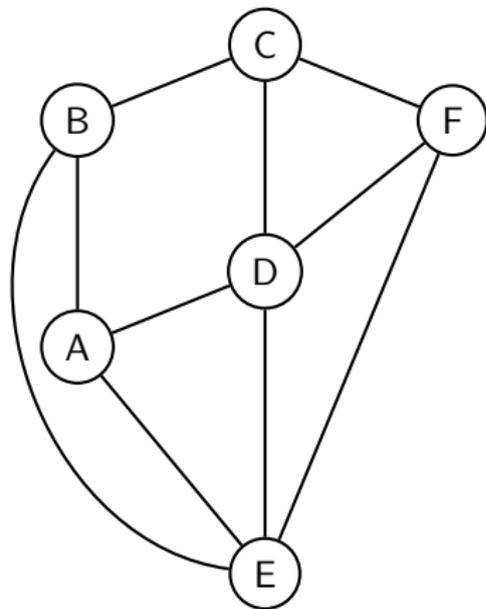
Giada Franz e Edoardo Morassutto

Online, 10 aprile 2021



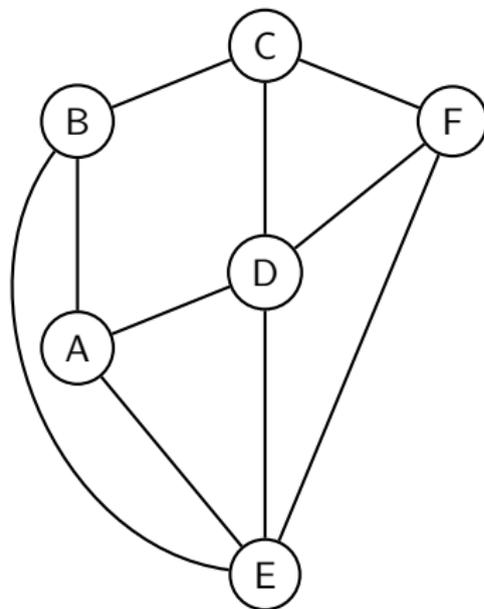
Cos'è un grafo?

Un insieme di N vertici



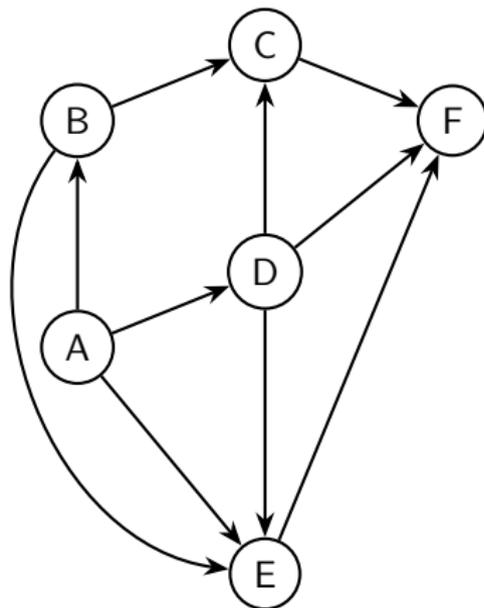
Cos'è un grafo?

Un insieme di N vertici e un insieme di M archi (coppie di nodi)



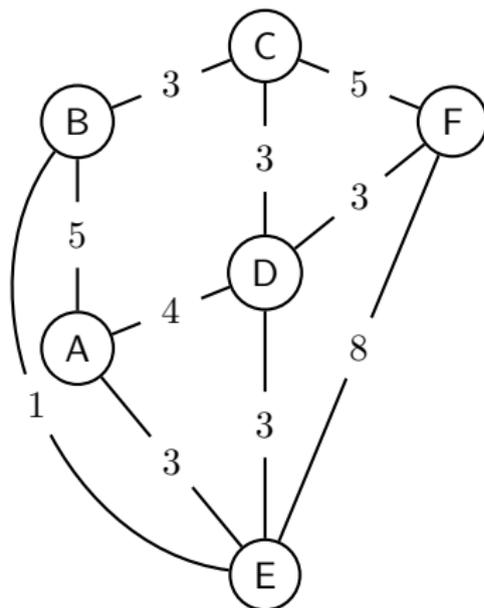
Cos'è un grafo?

Un insieme di N vertici e un insieme di M archi (coppie di nodi), eventualmente orientati

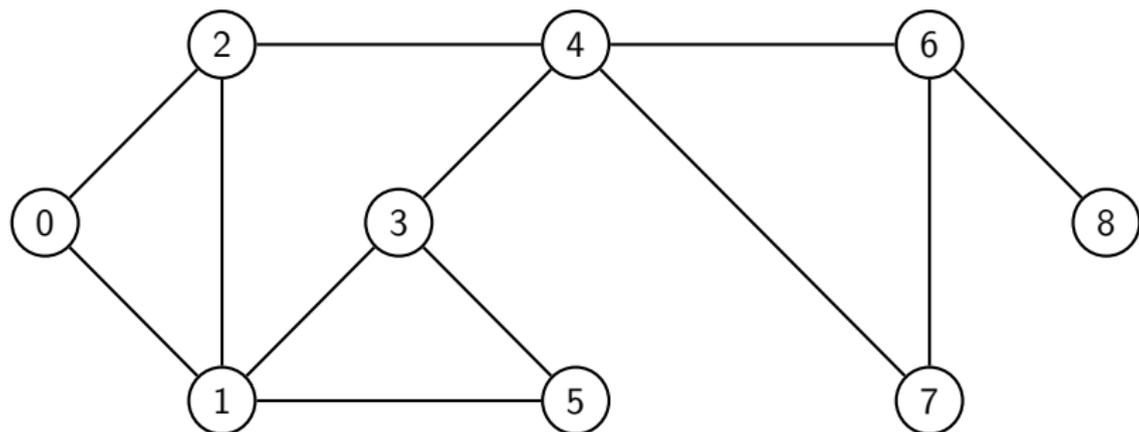


Cos'è un grafo?

Un insieme di N vertici e un insieme di M archi (coppie di nodi), eventualmente orientati e/o pesati.

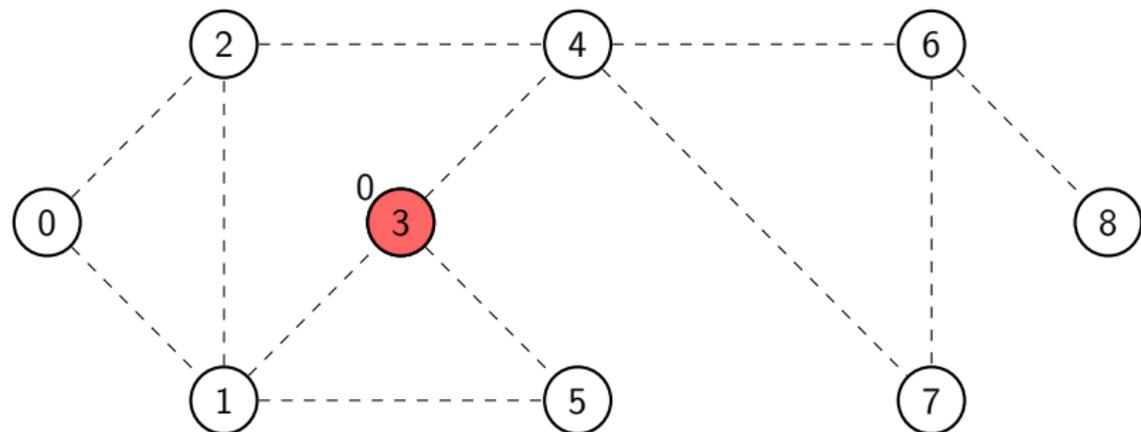


BFS o visita in ampiezza



BFS o visita in ampiezza

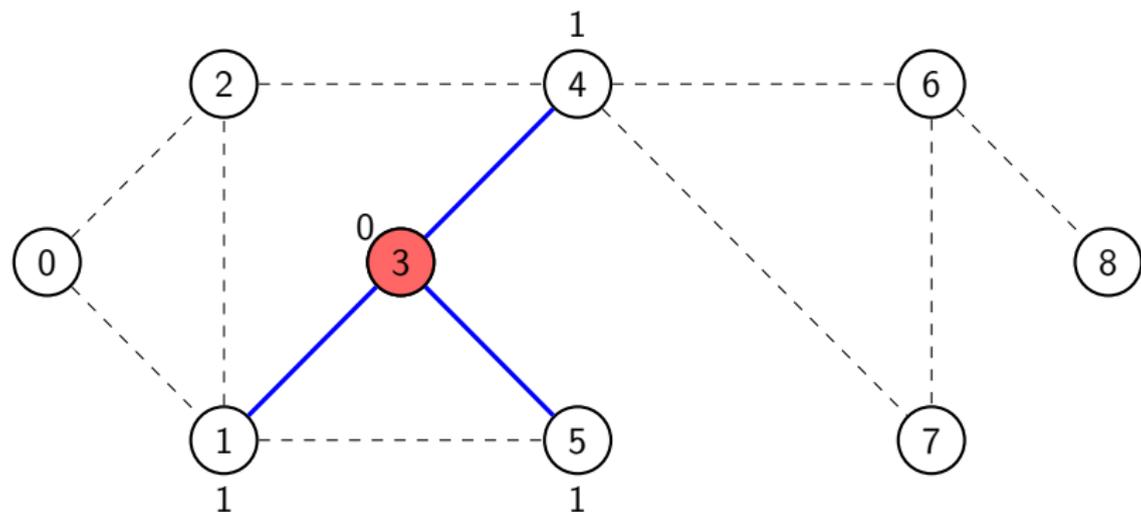
Inizia da un nodo ed imposta la sua distanza a 0.



$$Q = \{3\}$$

BFS o visita in ampiezza

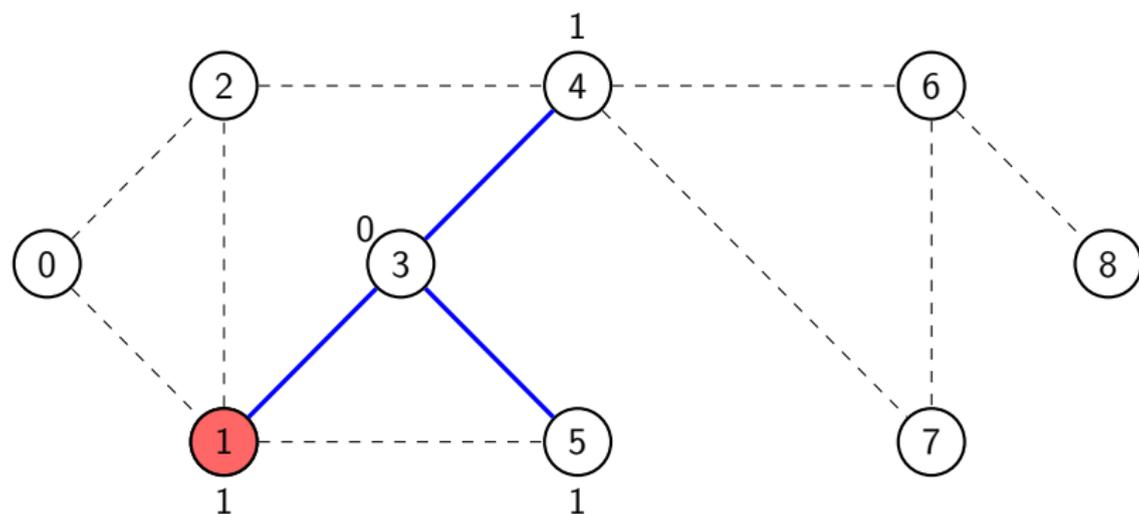
Espandi il nodo visitando tutti i suoi vicini, aggiornando la loro distanza e accodandoli alla coda.



$$Q = \{1, 4, 5\}$$

BFS o visita in ampiezza

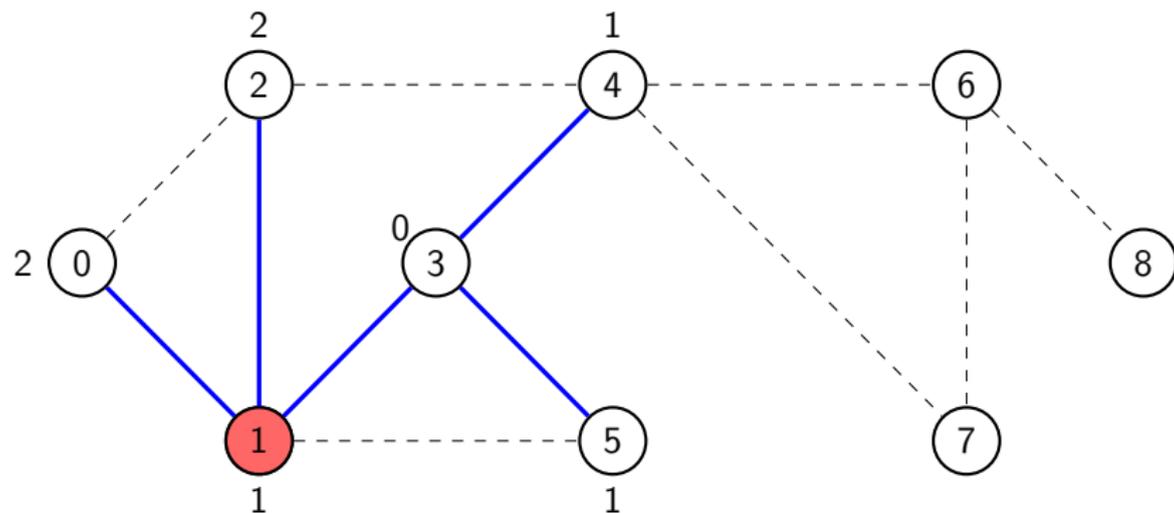
Seleziona il nodo successivo usando la coda.



$$Q = \{1, 4, 5\}$$

BFS o visita in ampiezza

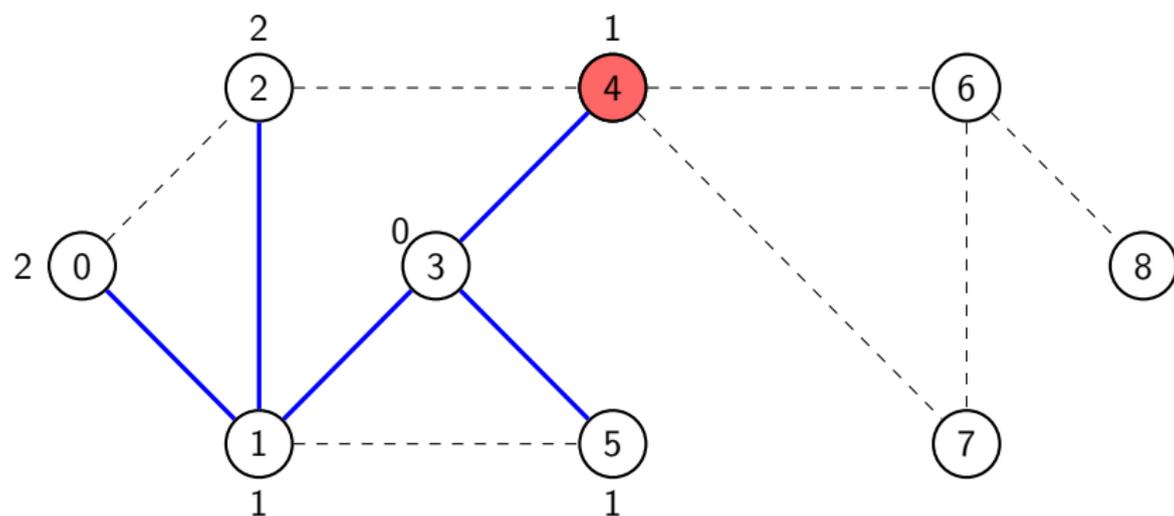
Espandi il nodo, visita i suoi vicini, aggiornando la loro distanza (se necessario!) e aggiungili alla coda (se hai aggiornato la distanza!)



$$Q = \{4, 5, 0, 2\}$$

BFS o visita in ampiezza

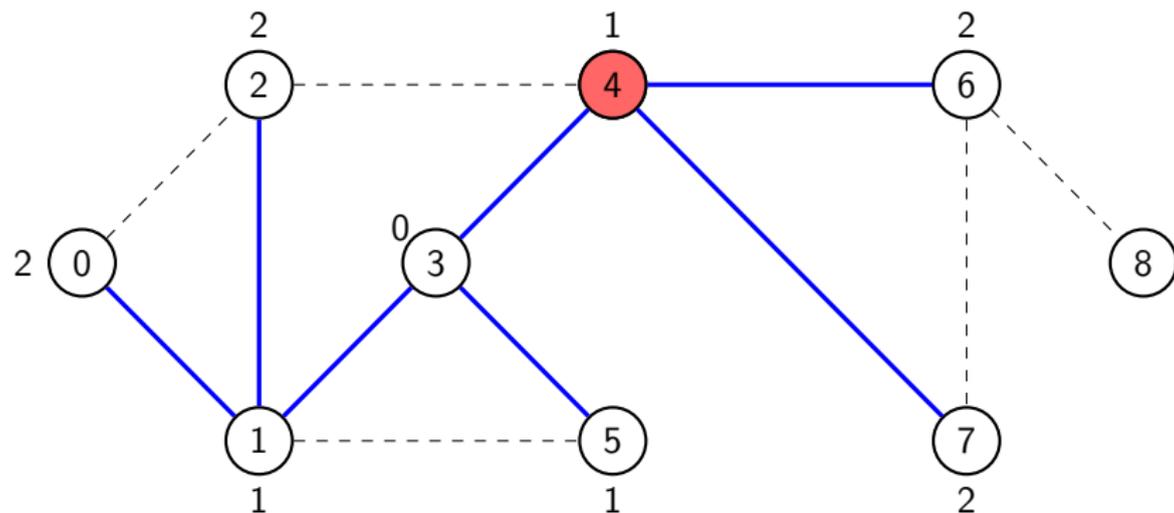
Seleziona il nodo successivo usando la coda.



$$Q = \{4, 5, 0, 2\}$$

BFS o visita in ampiezza

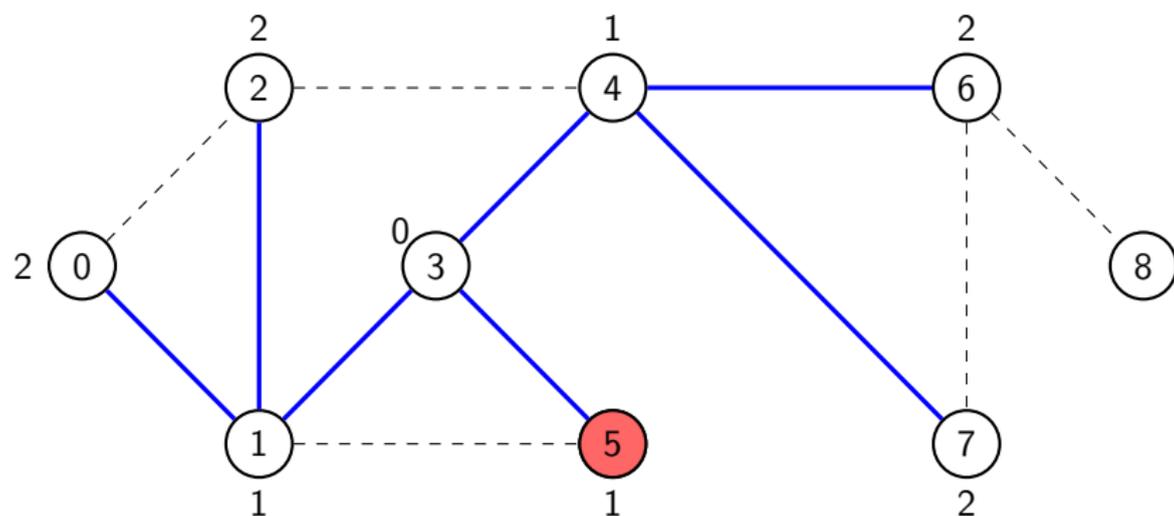
Espandi il nodo, visita i suoi vicini, aggiornando la loro distanza (se necessario!) e aggiungili alla coda (se hai aggiornato la distanza!)



$$Q = \{5, 0, 2, 6, 7\}$$

BFS o visita in ampiezza

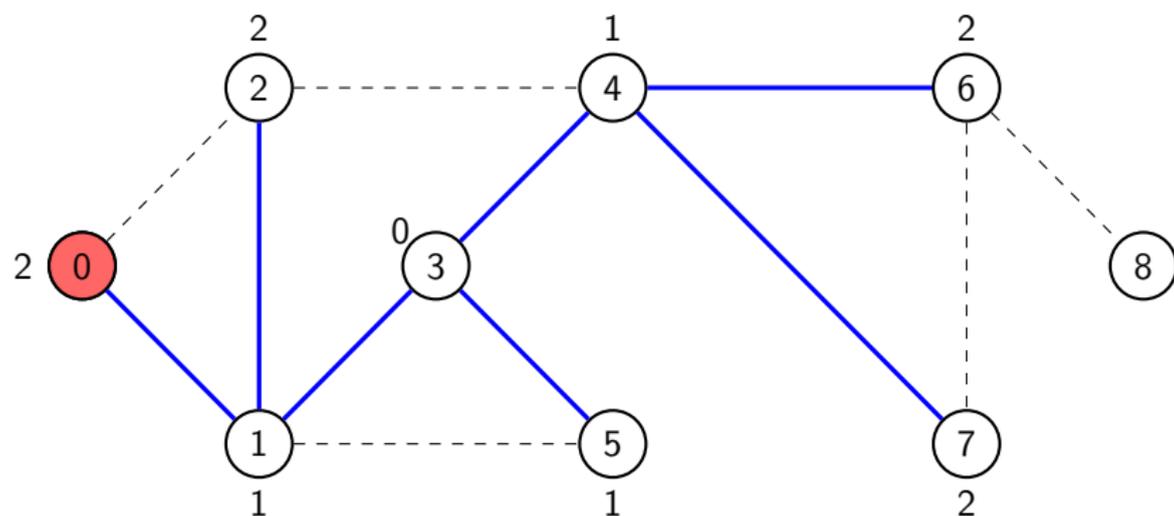
Seleziona il nodo successivo usando la coda.



$$Q = \{5, 0, 2, 6, 7\}$$

BFS o visita in ampiezza

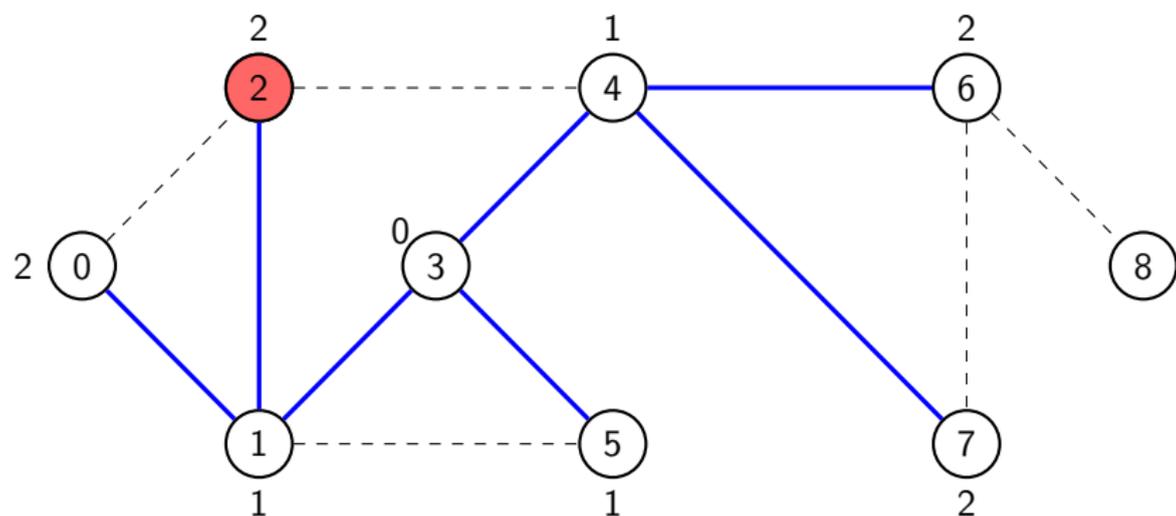
Seleziona il nodo successivo usando la coda.



$$Q = \{0, 2, 6, 7\}$$

BFS o visita in ampiezza

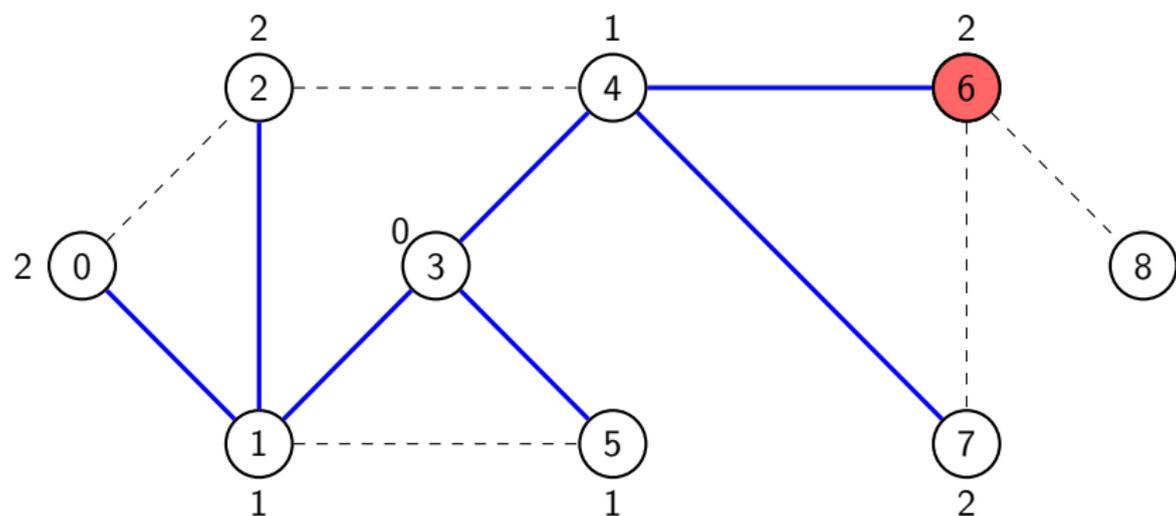
Seleziona il nodo successivo usando la coda.



$$Q = \{2, 6, 7\}$$

BFS o visita in ampiezza

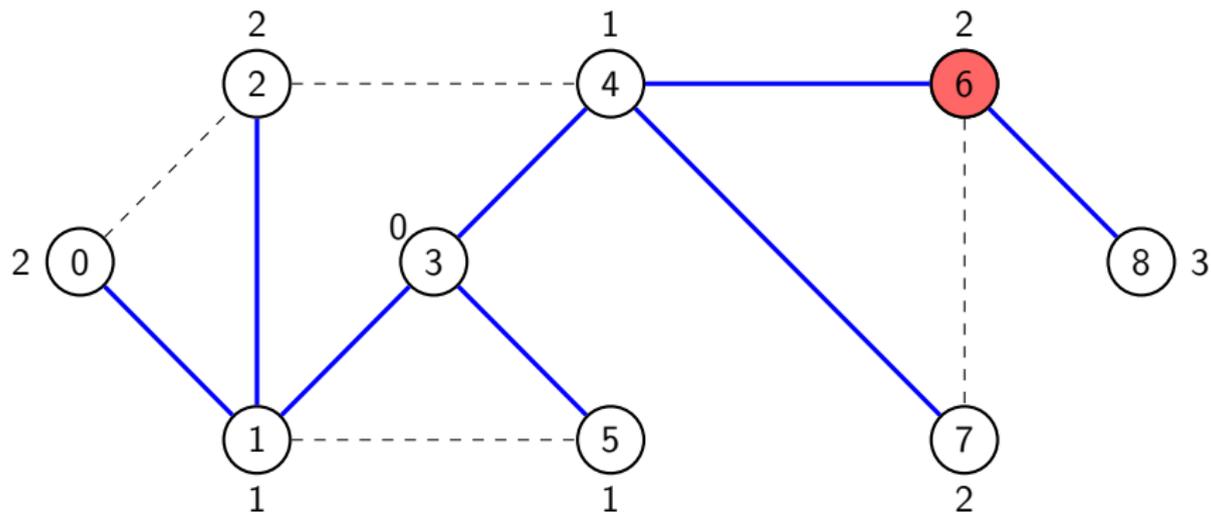
Seleziona il nodo successivo usando la coda.



$$Q = \{6, 7\}$$

BFS o visita in ampiezza

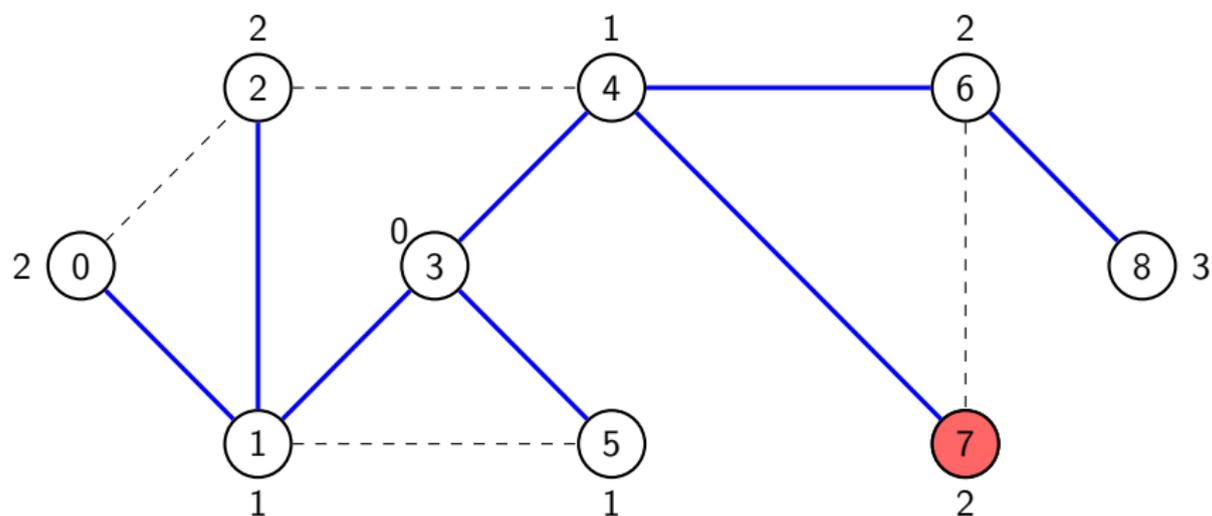
Espandi il nodo, visita i suoi vicini, aggiornando la loro distanza (se necessario!) e aggiungili alla coda (se hai aggiornato la distanza!)



$$Q = \{7, 8\}$$

BFS o visita in ampiezza

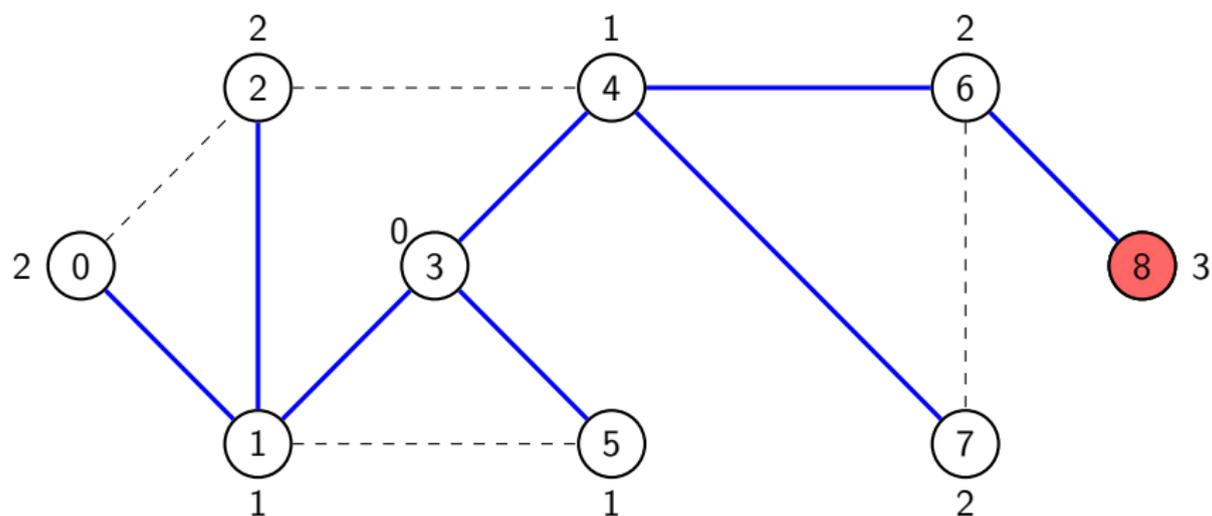
Seleziona il nodo successivo usando la coda.



$$Q = \{7, 8\}$$

BFS o visita in ampiezza

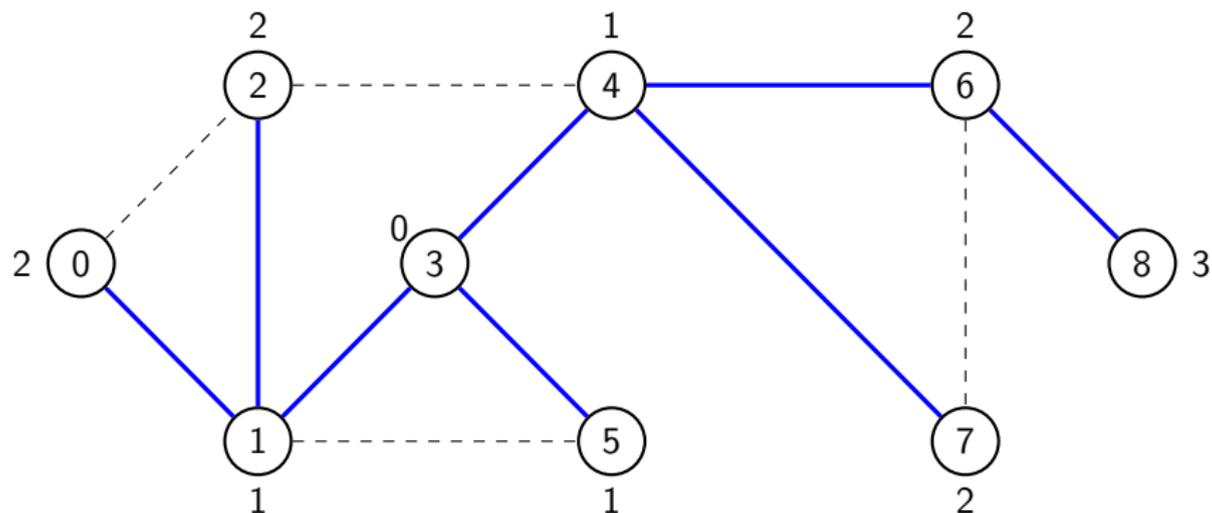
Seleziona il nodo successivo usando la coda.



$$Q = \{8\}$$

BFS o visita in ampiezza

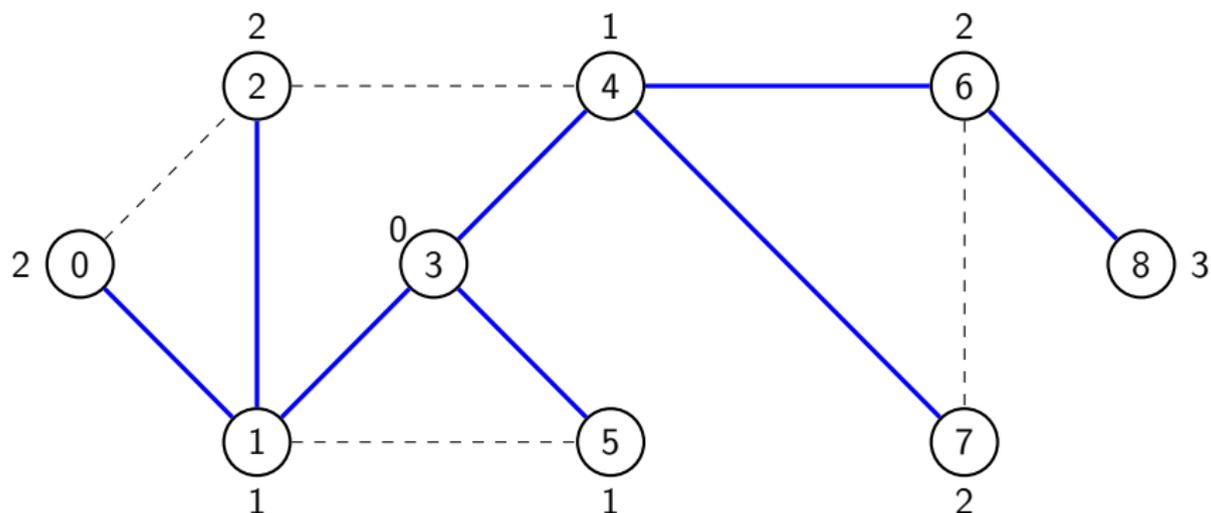
La coda è vuota, l'algorithmo termina.



$$Q = \{\}$$

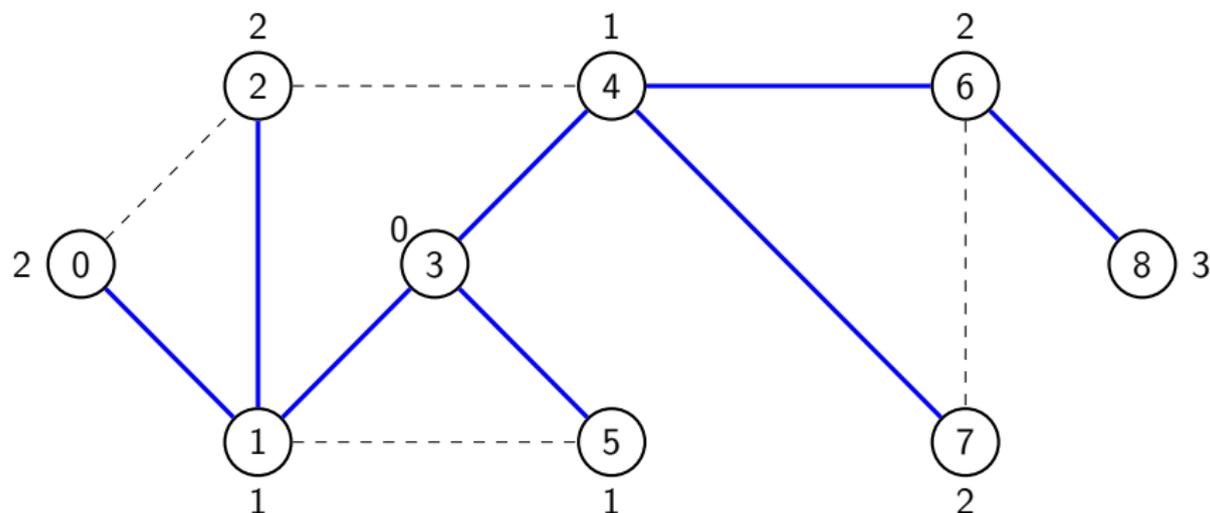
BFS o visita in ampiezza

- In quali condizioni i valori di distanza dei nodi sono *corretti*?

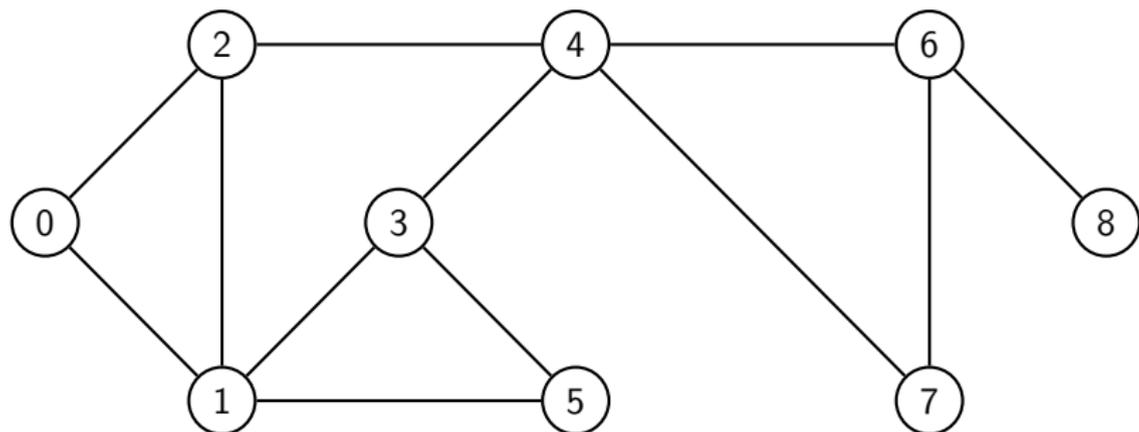


BFS o visita in ampiezza

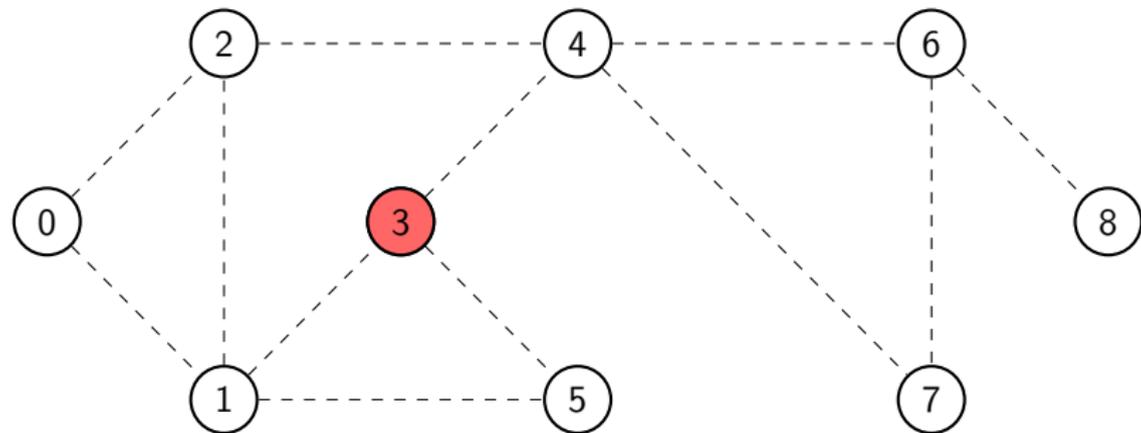
- In quali condizioni i valori di distanza dei nodi sono *corretti*?
- Qual è la complessità computazionale di questo algoritmo?



DFS o visita in profondità

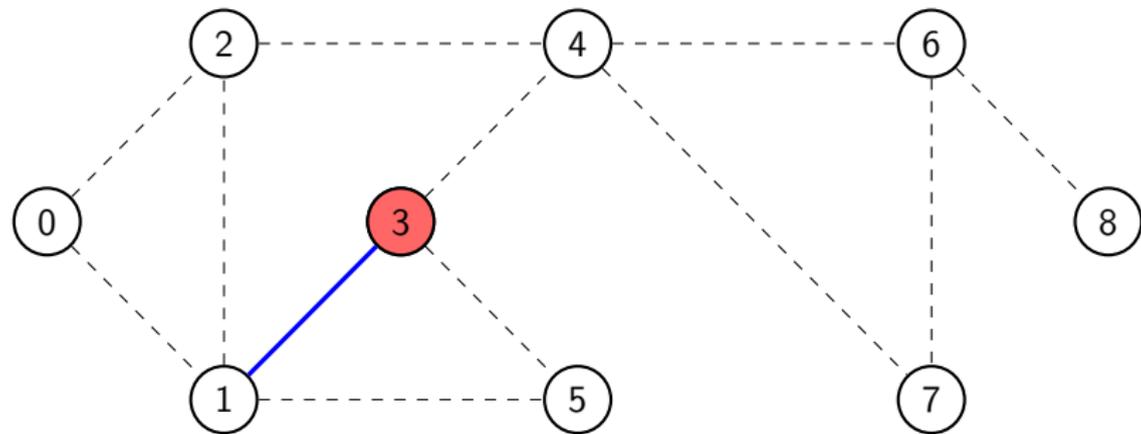


DFS o visita in profondità



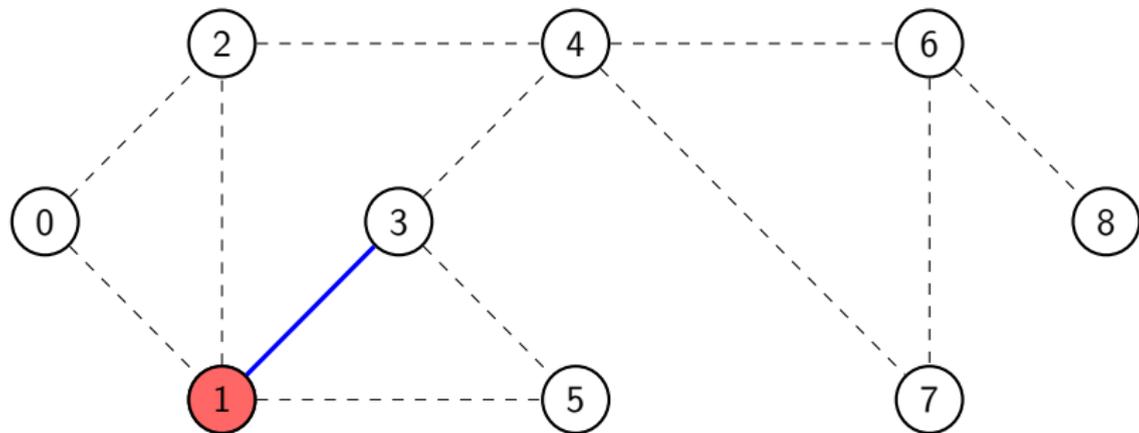
$$S = \{3\}$$

DFS o visita in profondità



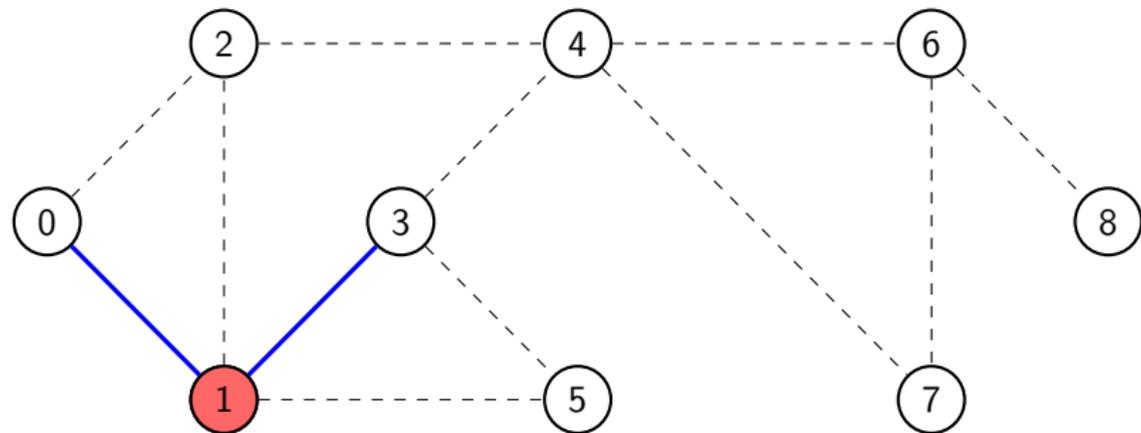
$$S = \{3\}$$

DFS o visita in profondità



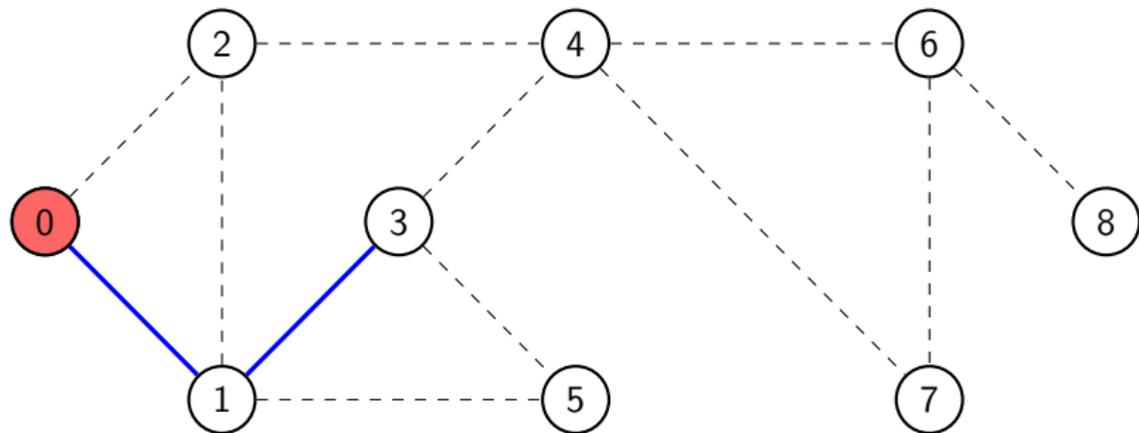
$$S = \{3, 1\}$$

DFS o visita in profondità



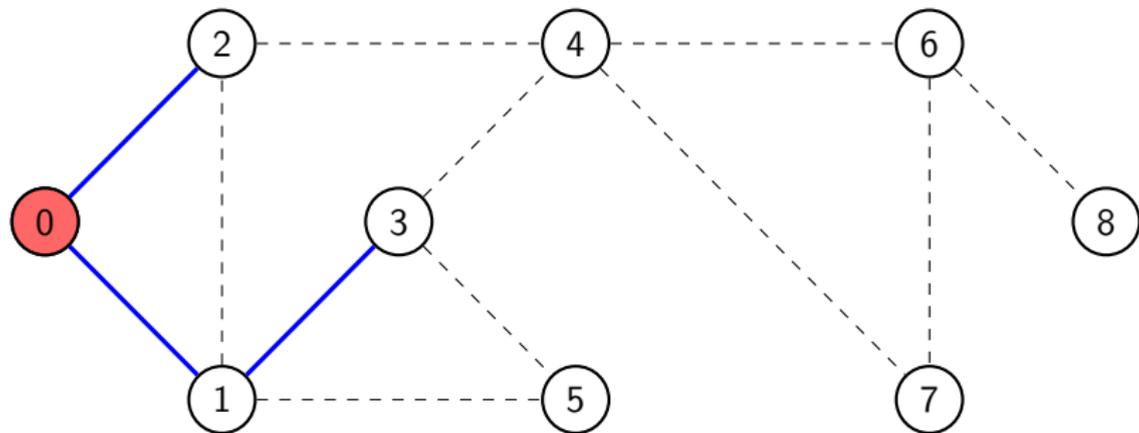
$$S = \{3, \underline{1}\}$$

DFS o visita in profondità



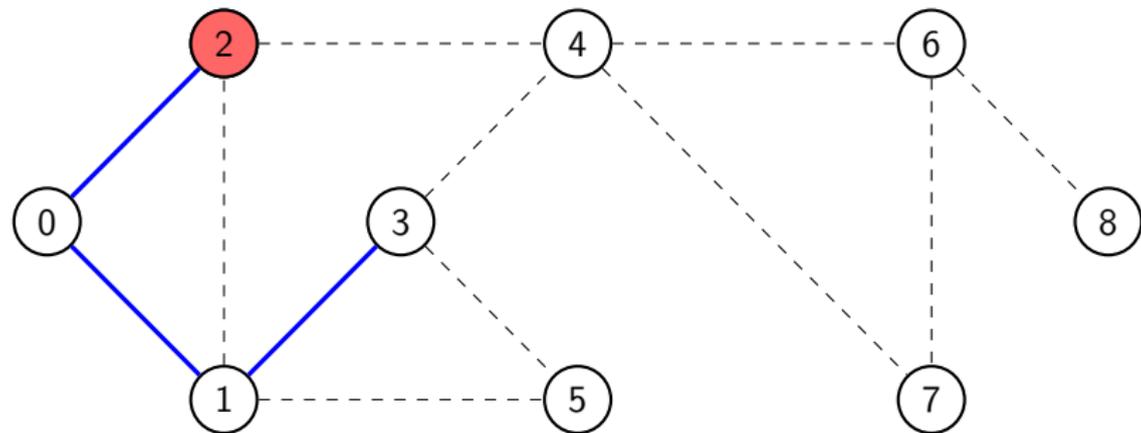
$$S = \{3, 1, 0\}$$

DFS o visita in profondità



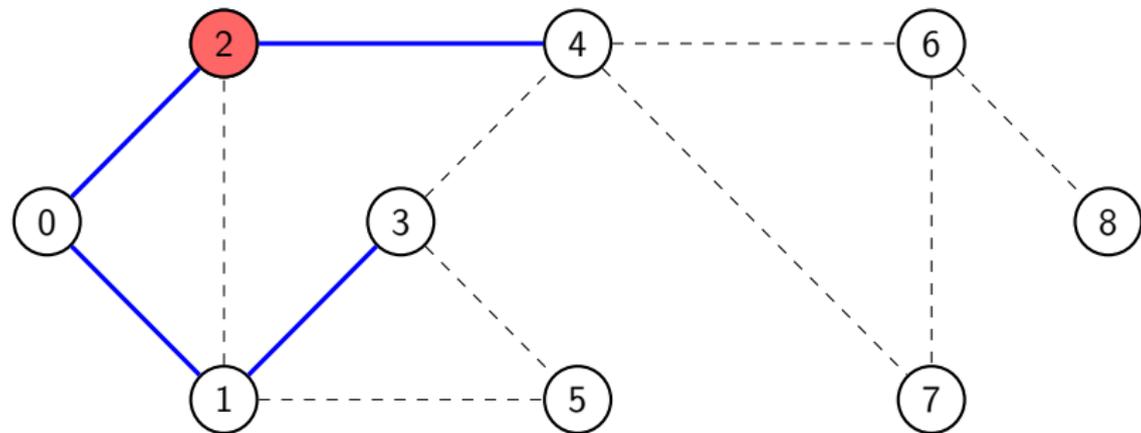
$$S = \{3, 1, \underline{0}\}$$

DFS o visita in profondità



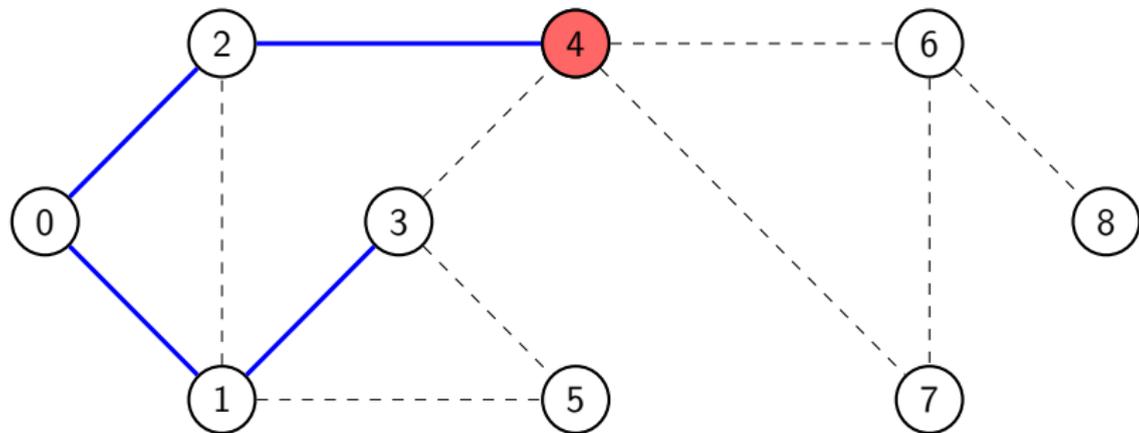
$$S = \{3, 1, 0, 2\}$$

DFS o visita in profondità



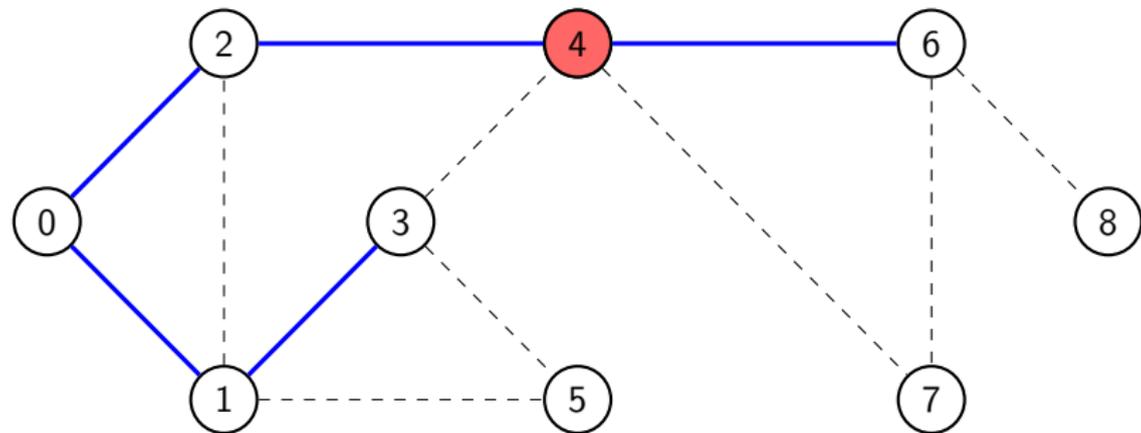
$$S = \{3, 1, 0, \underline{2}\}$$

DFS o visita in profondità



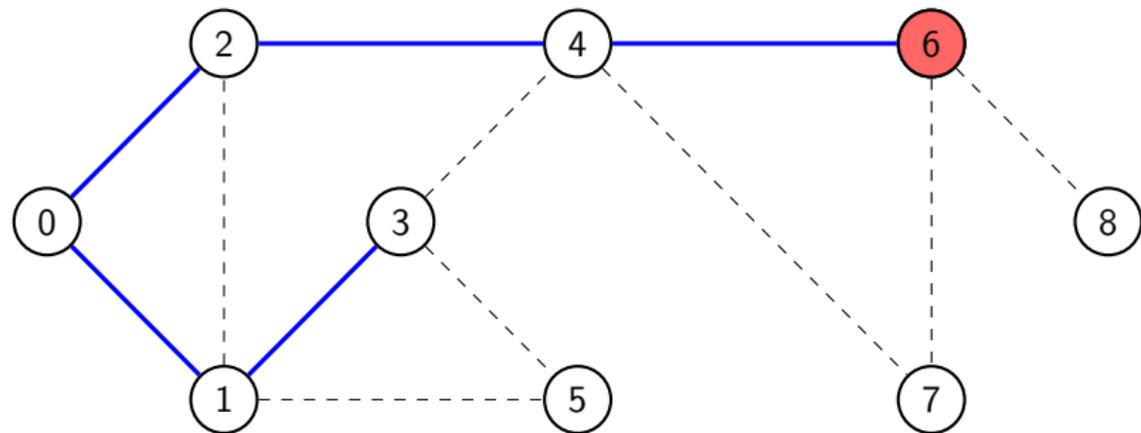
$$S = \{3, 1, 0, 2, 4\}$$

DFS o visita in profondità



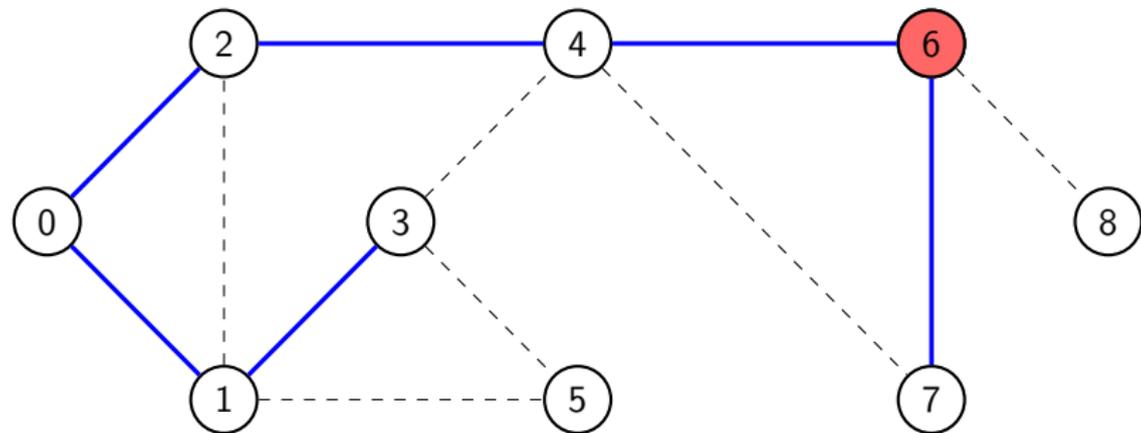
$$S = \{3, 1, 0, 2, \underline{4}\}$$

DFS o visita in profondità



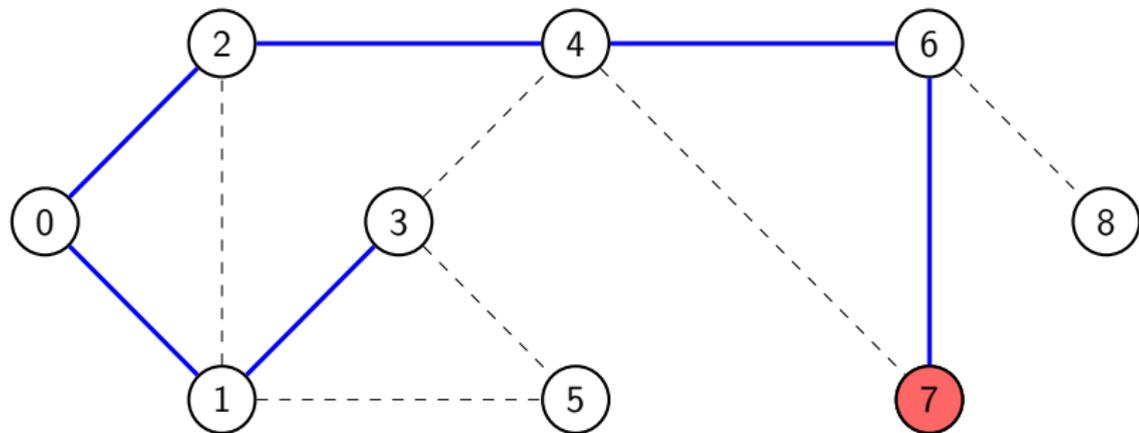
$$S = \{3, 1, 0, 2, 4, 6\}$$

DFS o visita in profondità



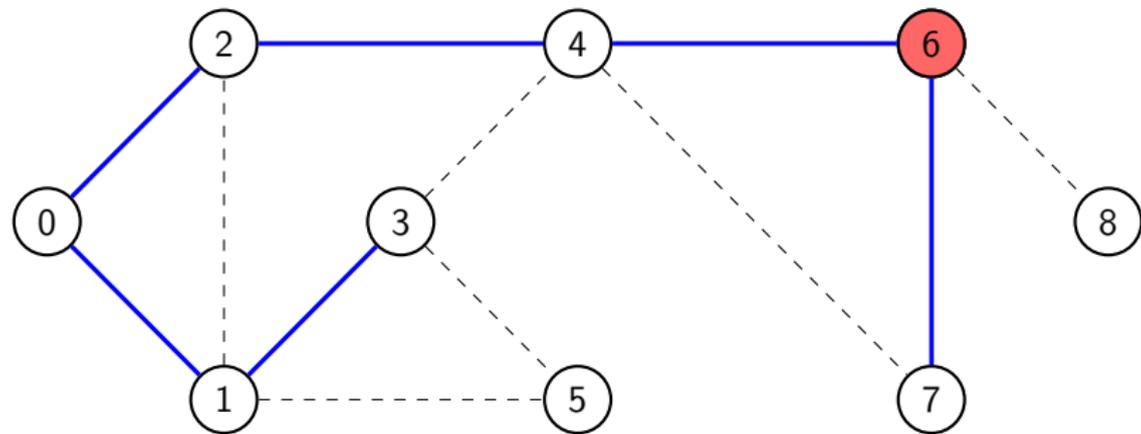
$$S = \{3, 1, 0, 2, 4, \underline{6}\}$$

DFS o visita in profondità



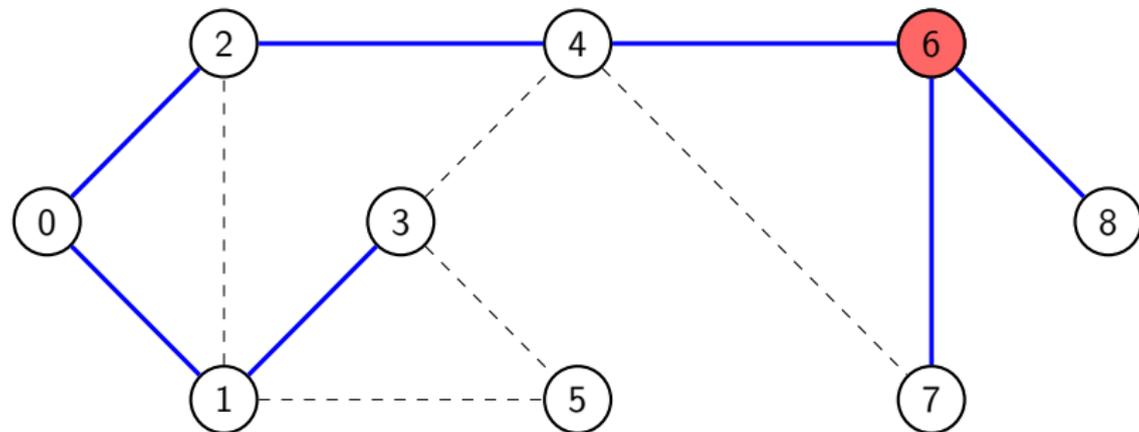
$$S = \{3, 1, 0, 2, 4, 6, 7\}$$

DFS o visita in profondità



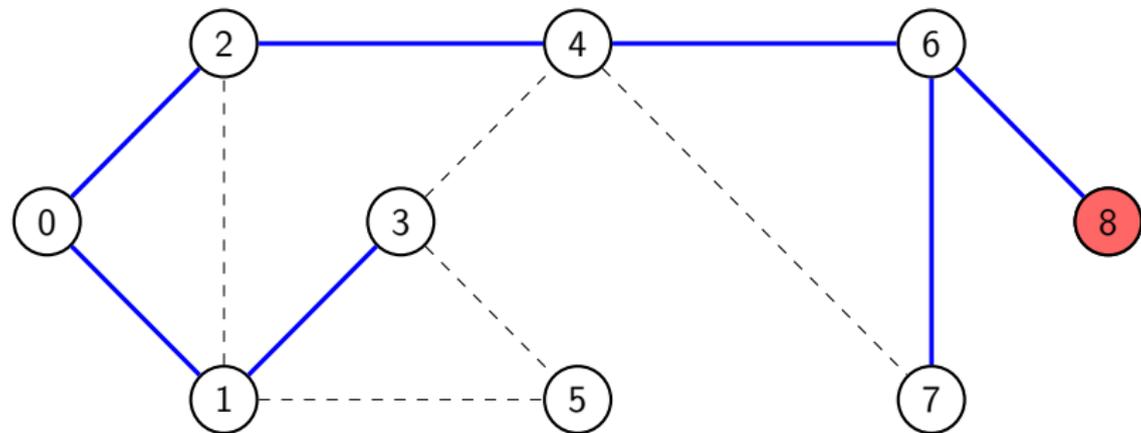
$$S = \{3, 1, 0, 2, 4, 6\}$$

DFS o visita in profondità



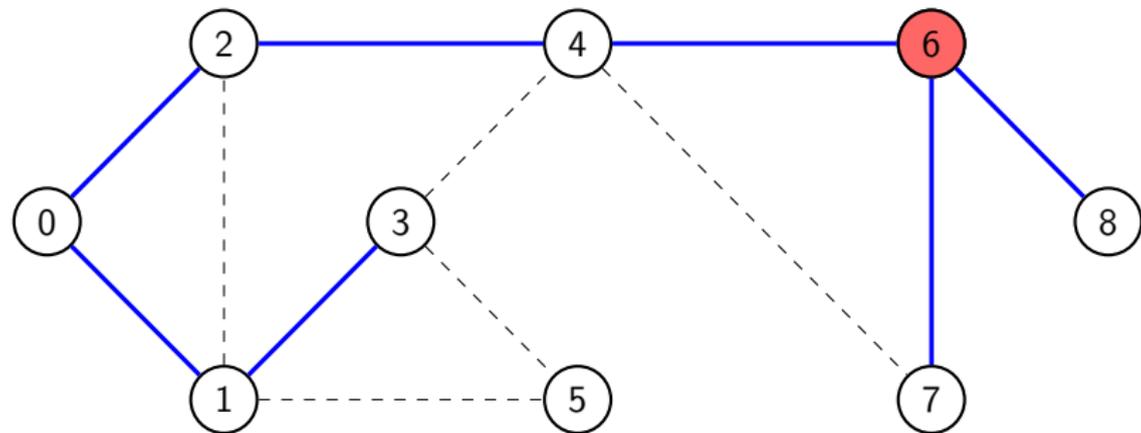
$$S = \{3, 1, 0, 2, 4, \underline{6}\}$$

DFS o visita in profondità



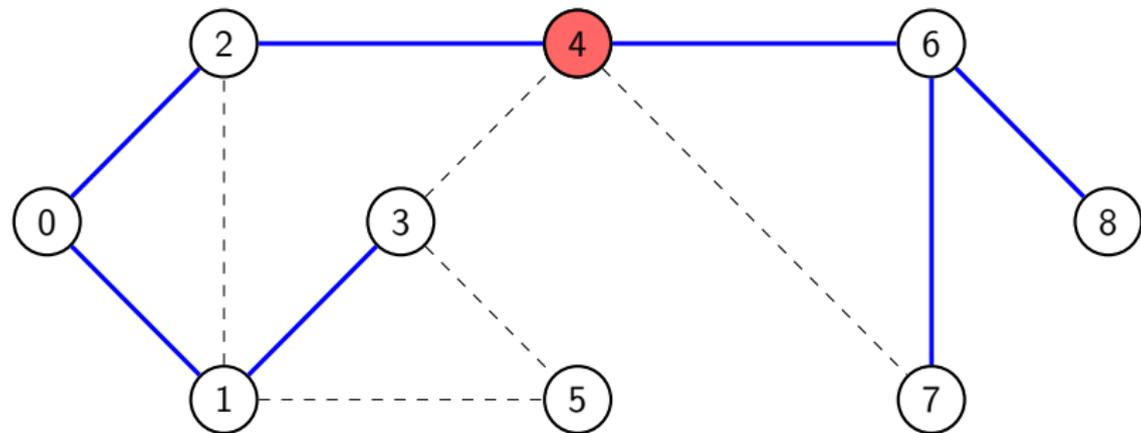
$$S = \{3, 1, 0, 2, 4, 6, 8\}$$

DFS o visita in profondità



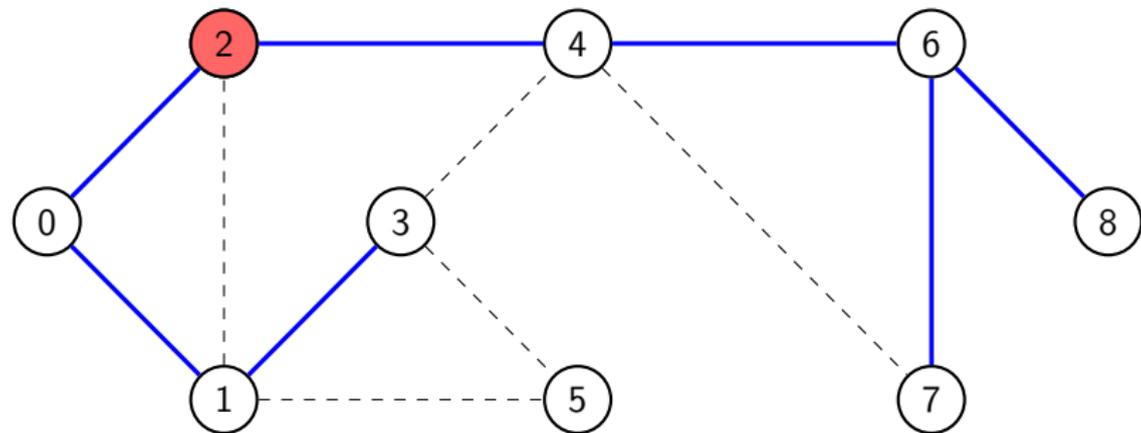
$$S = \{3, 1, 0, 2, 4, 6\}$$

DFS o visita in profondità



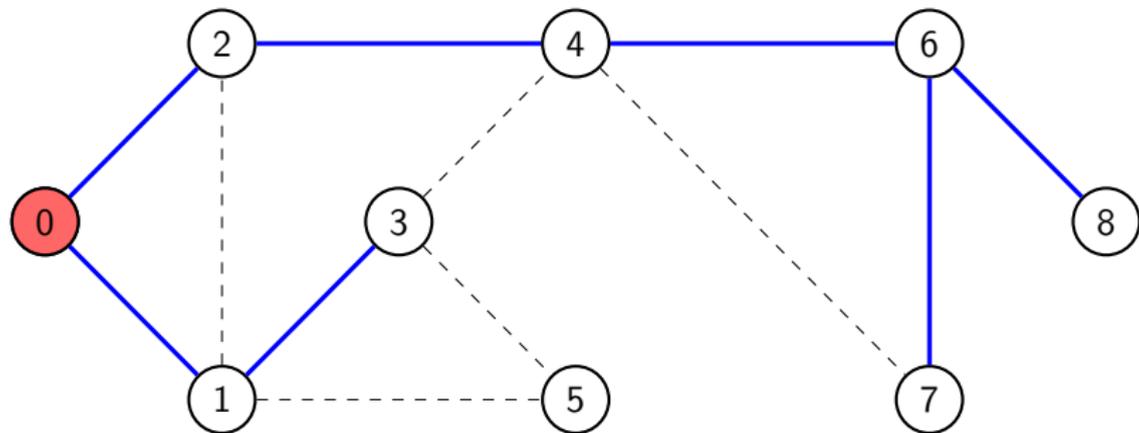
$$S = \{3, 1, 0, 2, 4\}$$

DFS o visita in profondità



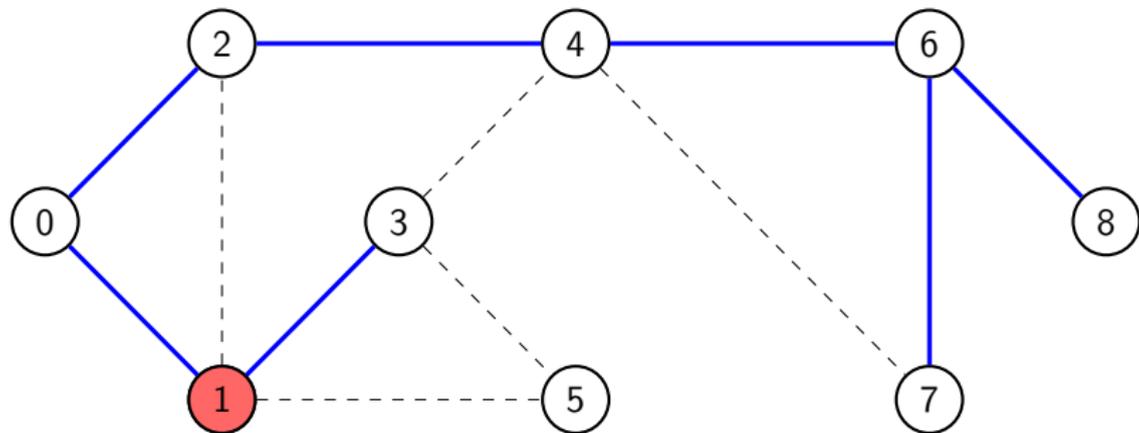
$$S = \{3, 1, 0, 2\}$$

DFS o visita in profondità



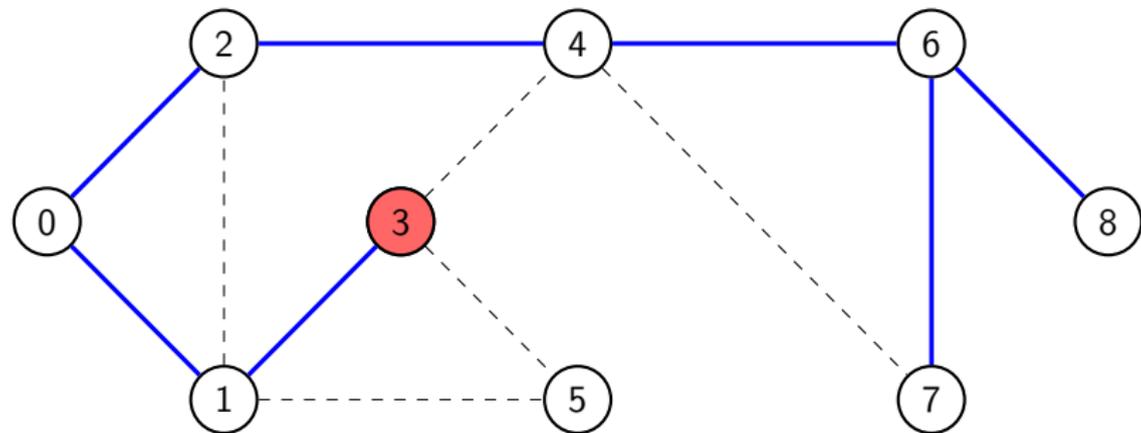
$$S = \{3, 1, 0\}$$

DFS o visita in profondità



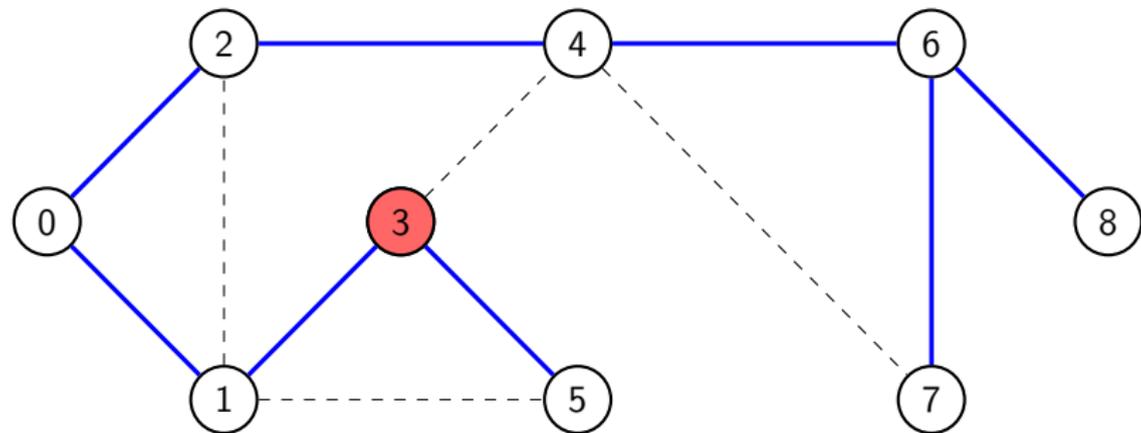
$$S = \{3, 1\}$$

DFS o visita in profondità



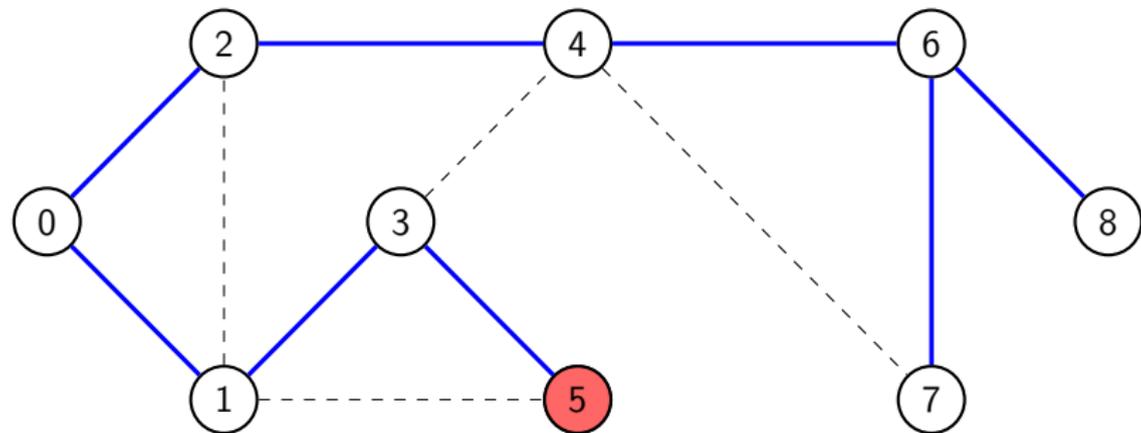
$$S = \{3\}$$

DFS o visita in profondità



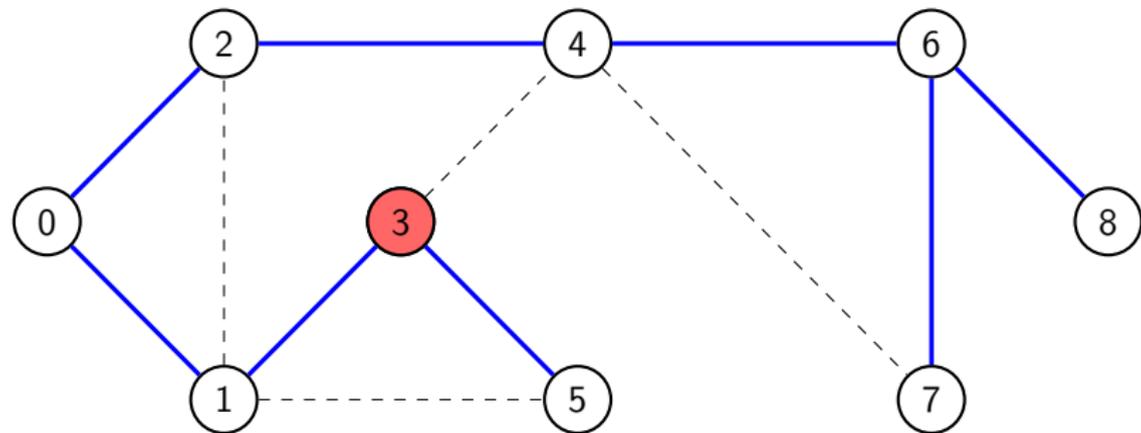
$$S = \{3\}$$

DFS o visita in profondità



$$S = \{3, 5\}$$

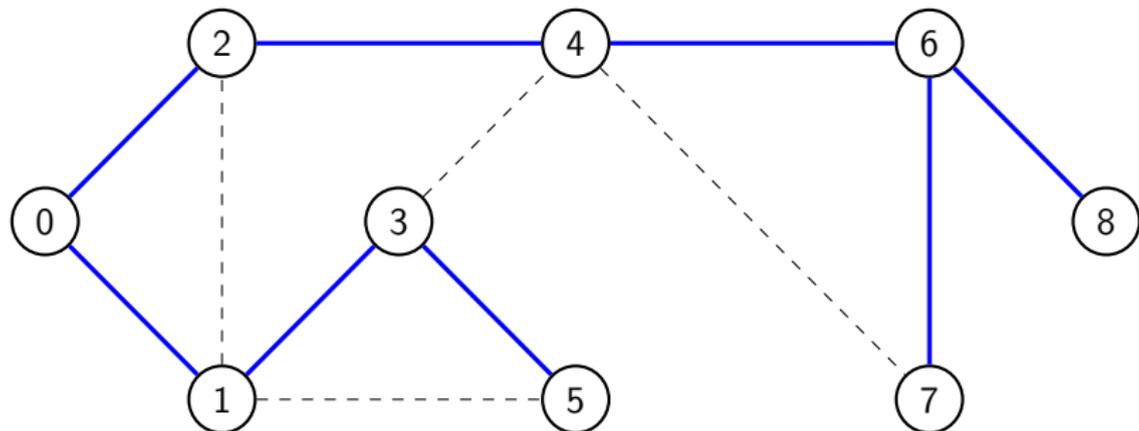
DFS o visita in profondità



$$S = \{3\}$$

DFS o visita in profondità

La pila è vuota, l'algorithm termina.



$$S = \{\}$$

Esercizio

Trovare le componenti connesse di un grafo.

Due nodi sono nella stessa componente connessa se e solo se esiste un cammino che li collega.

Esercizio

Dire se un grafo dato è bipartito ed eventualmente mostrarne la bipartizione.

Un grafo è bipartito se è possibile dividere i nodi in due gruppi e tutti gli archi collegano nodi di un gruppo con nodi dell'altro.

Problema

Siamo interessati a trovare il **percorso più breve** fra due vertici in un grafo con archi pesati.

Problema

Siamo interessati a trovare il **percorso più breve** fra due vertici in un grafo con archi pesati.

In realtà tutti gli algoritmi che studieremo ci daranno un risultato leggermente più generale, cioè troveremo i percorsi più brevi da un dato vertice a *tutti* gli altri vertici del grafo.

Problema

Siamo interessati a trovare il **percorso più breve** fra due vertici in un grafo con archi pesati.

In realtà tutti gli algoritmi che studieremo ci daranno un risultato leggermente più generale, cioè troveremo i percorsi più brevi da un dato vertice a *tutti* gli altri vertici del grafo.

Osservazione

Possiamo assumere che il grafo sia connesso: dal nodo in questione è possibile raggiungere ogni altro nodo.

Struttura dei cammini minimi

Fissiamo un vertice v e per ogni altro vertice w segniamo sul grafo un cammino minimo che collega v a w .

Allora si possono scegliere tali cammini minimi in modo che il sottografo risultante sia un albero, detto **albero dei cammini minimi**.

Algoritmo di Dijkstra

L'algoritmo di Dijkstra risolve il problema dei cammini minimi da un vertice nel caso in cui il grafo abbia **pesi** degli archi **non negativi**.

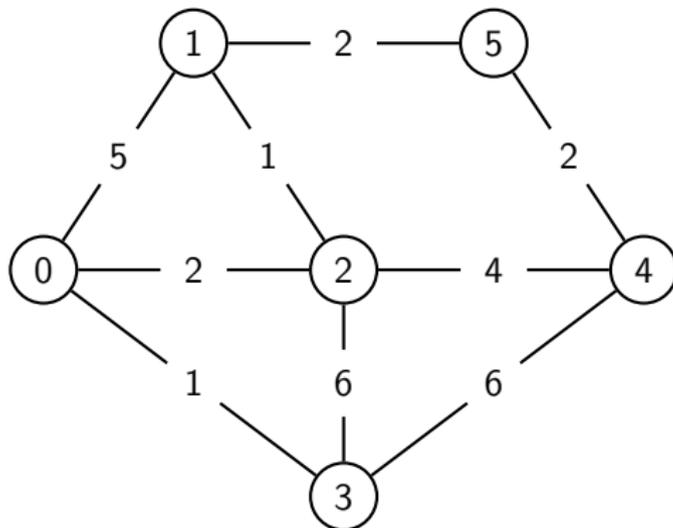
Osservazione

Se il grafo ha archi con pesi negativi il problema potrebbe diventare non ben definito: possono esistere cicli di lunghezza negativa!

Algoritmo di Dijkstra

Algoritmo

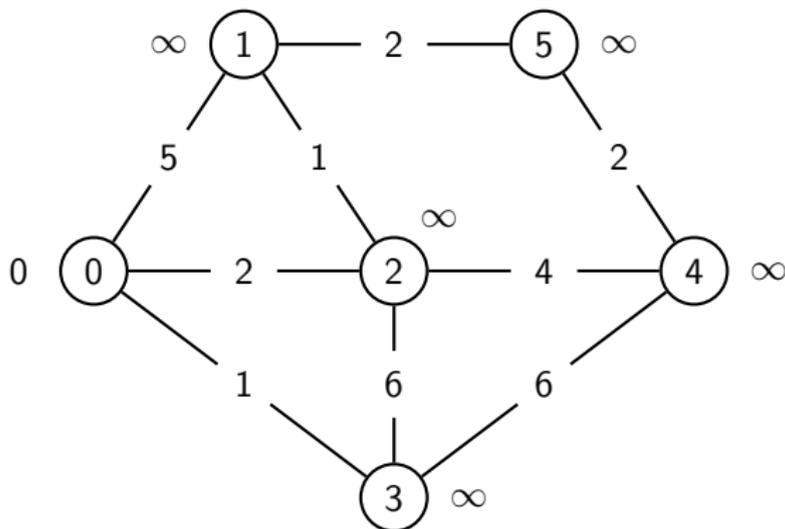
Vogliamo calcolare la distanza minima dal nodo 0 verso tutti gli altri nodi.



Algoritmo di Dijkstra

Algoritmo

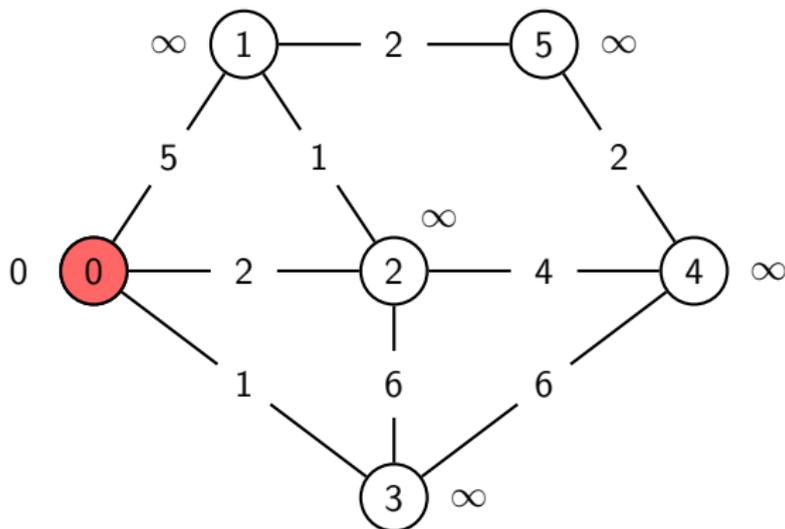
Inizializzazione: la distanza di tutti i nodi è ∞ , **tranne per la sorgente**.



Algoritmo di Dijkstra

Algoritmo

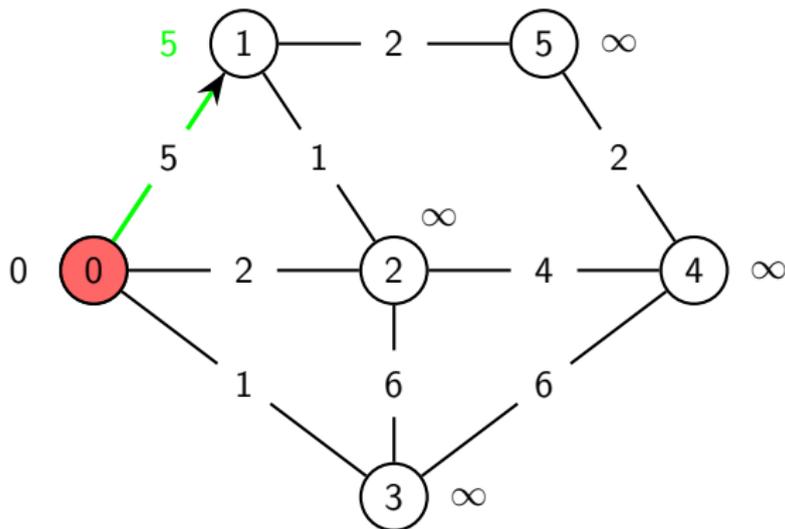
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

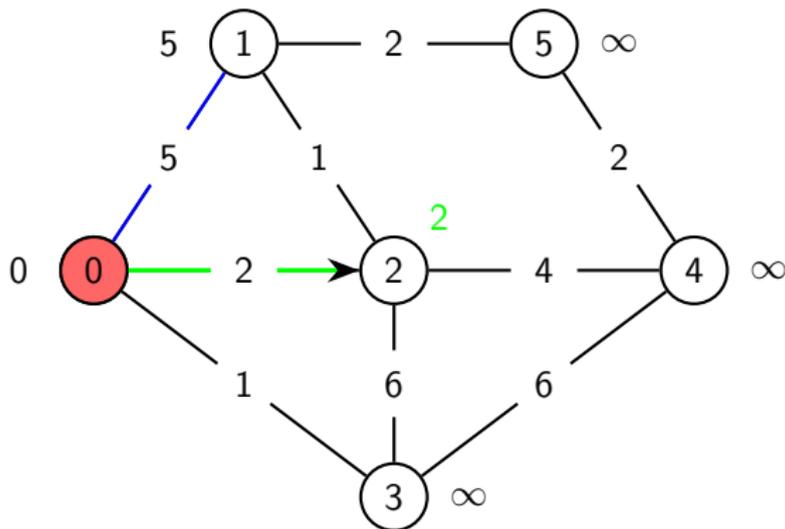


$$0 + 5 < \infty$$

Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

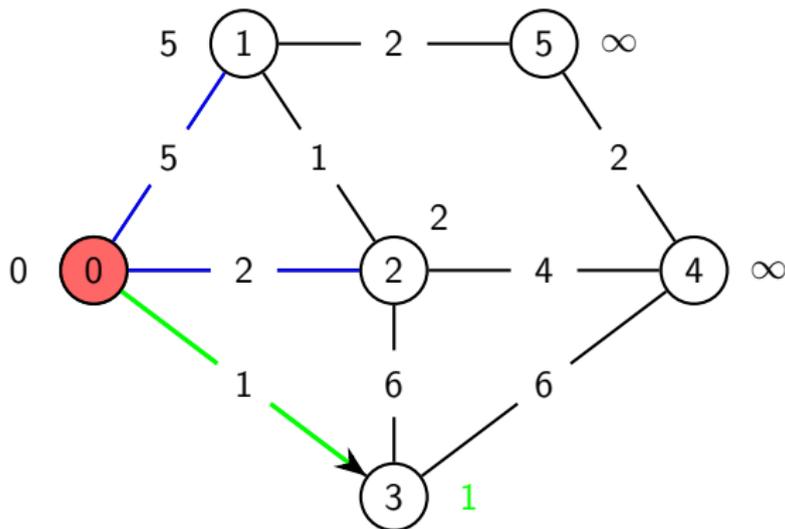


$$0 + 2 < \infty$$

Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

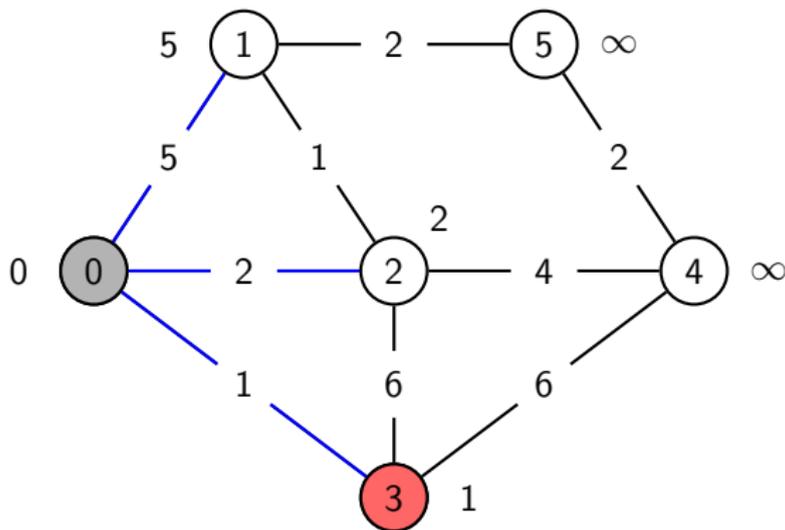


$$0 + 1 < \infty$$

Algoritmo di Dijkstra

Algoritmo

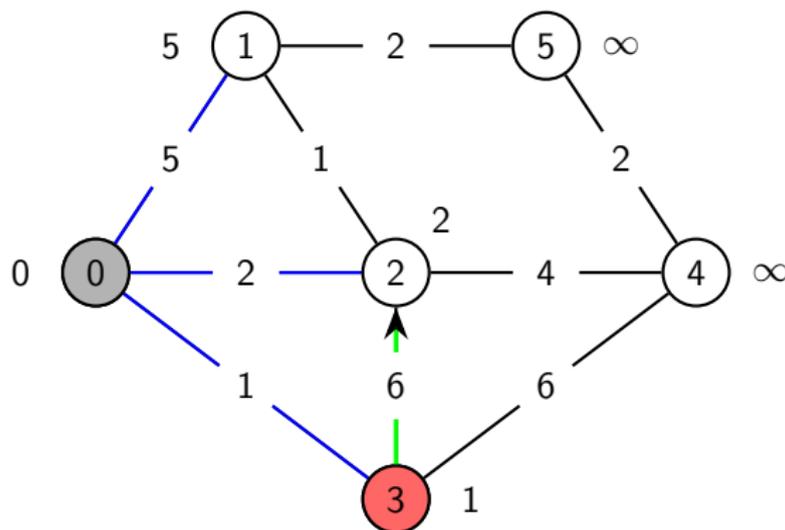
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

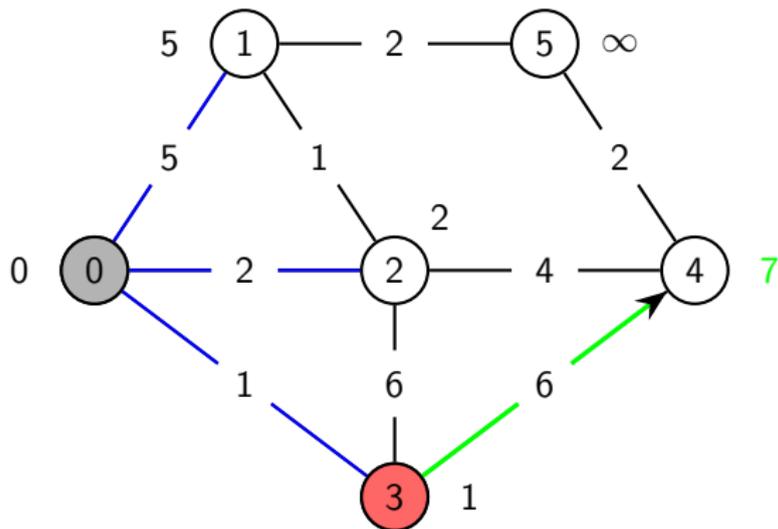


$$1 + 6 \not\leq 2$$

Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

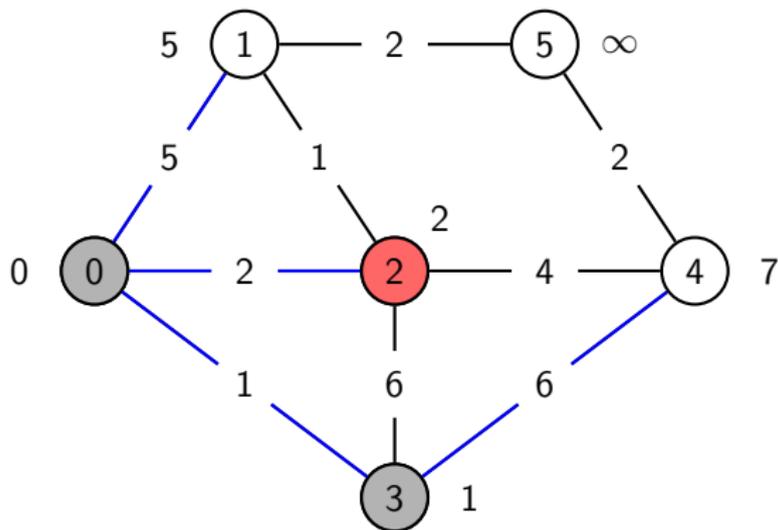


$$1 + 6 < \infty$$

Algoritmo di Dijkstra

Algoritmo

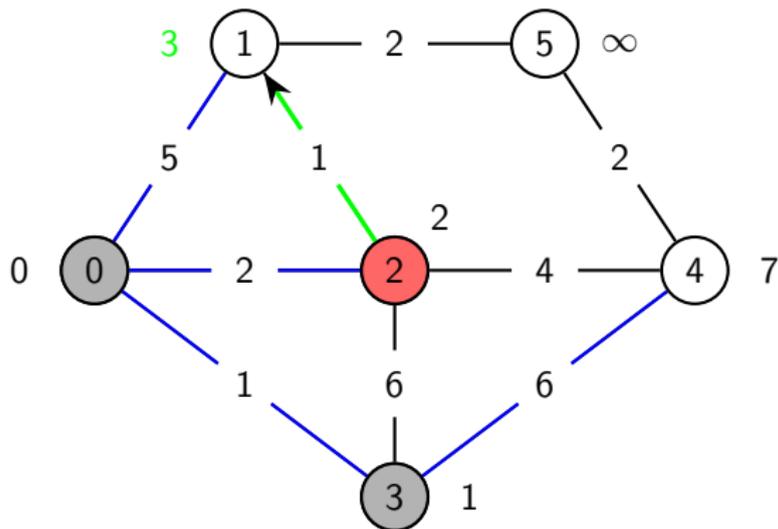
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

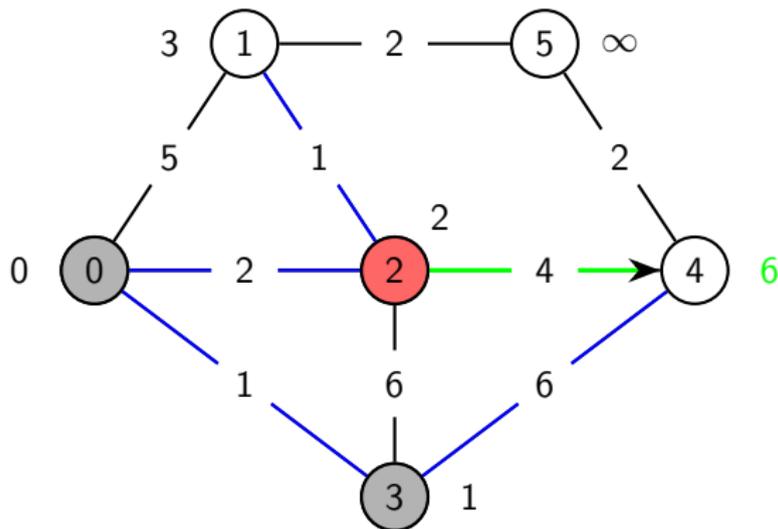


$$2 + 1 < 5$$

Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

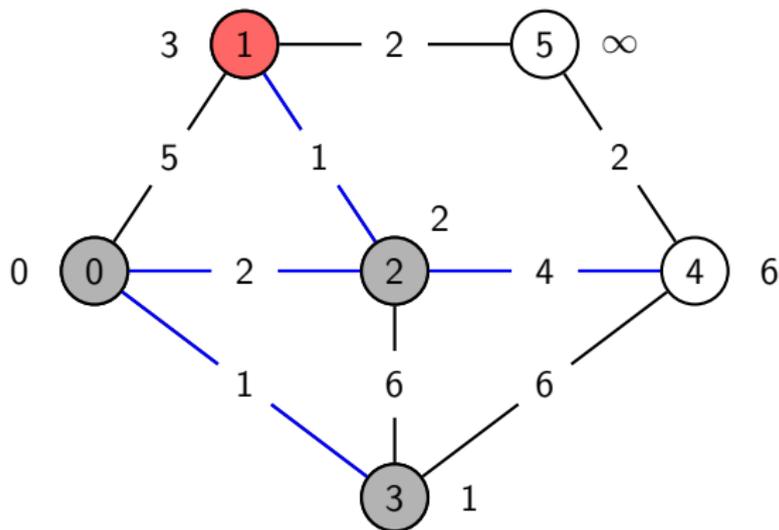


$$2 + 4 < 6$$

Algoritmo di Dijkstra

Algoritmo

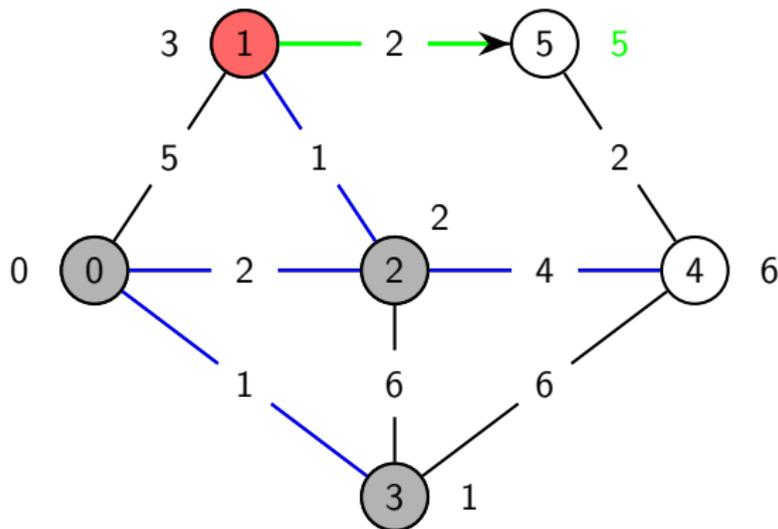
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

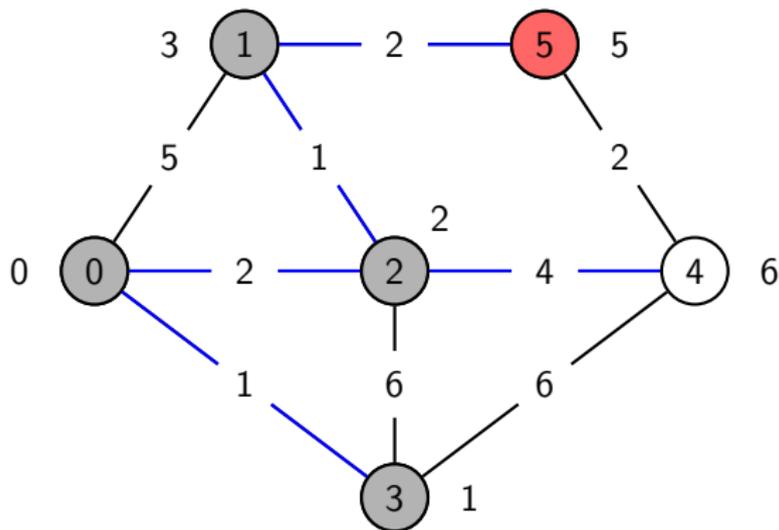


$$3 + 2 < \infty$$

Algoritmo di Dijkstra

Algoritmo

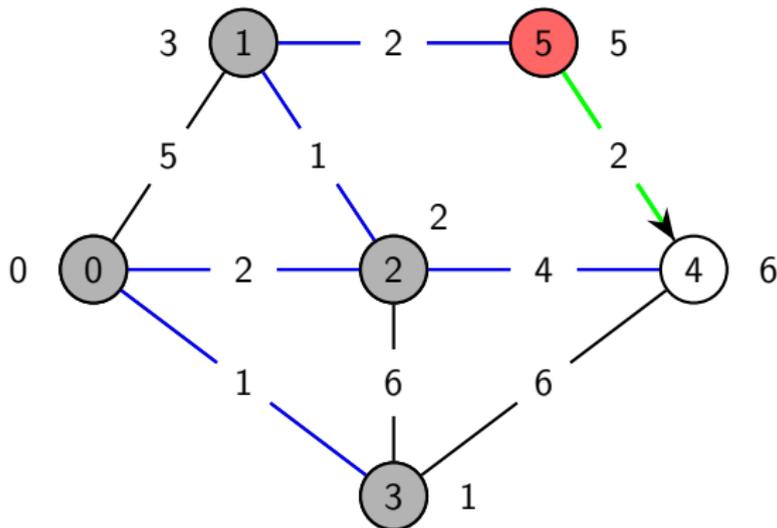
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Espandiamo il nodo visitando i suoi vicini e aggiornando (se necessario!) la loro distanza.

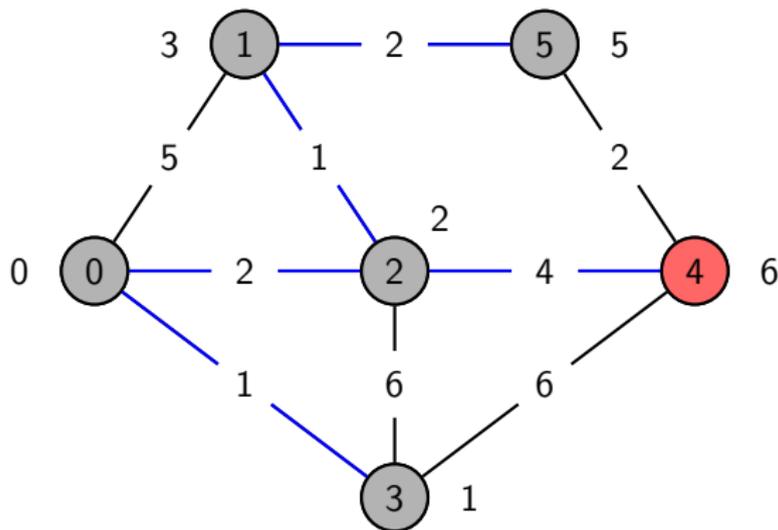


$$5 + 2 \not\leq 6$$

Algoritmo di Dijkstra

Algoritmo

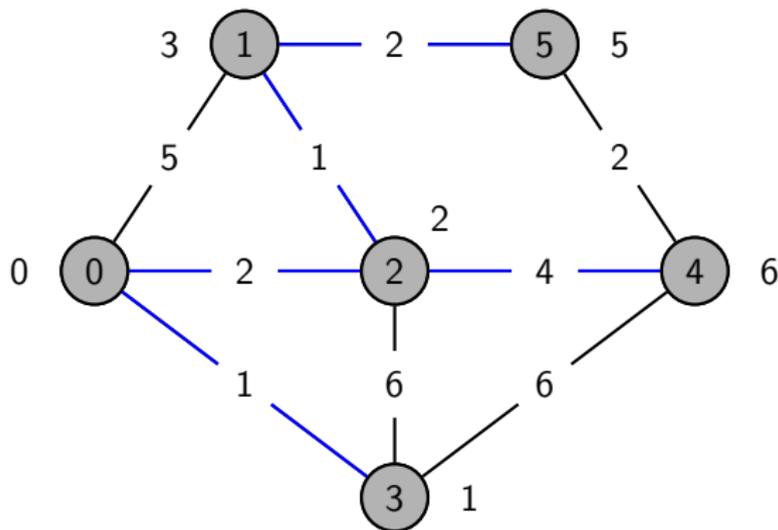
Selezioniamo il nodo non espanso con distanza minima.



Algoritmo di Dijkstra

Algoritmo

Tutti i nodi sono stati espansi, l'algoritmo termina.



Algoritmo di Dijkstra

Complessità

L'algoritmo effettua N step: ogni nodo viene espanso esattamente una volta.
All'interno di uno step si deve:

- Selezionare il nodo non visitato con distanza minima
- Visitare tutti i suoi vicini

Algoritmo di Dijkstra

Complessità

L'algoritmo effettua N step: ogni nodo viene espanso esattamente una volta. All'interno di uno step si deve:

- Selezionare il nodo non visitato con distanza minima
- Visitare tutti i suoi vicini

Ogni arco viene processato esattamente una volta, ogni nodo viene processato esattamente una volta.

La complessità dell'algoritmo dipende da come viene trovato il vertice non ancora processato con distanza minore. Abbiamo quindi due casi:

Algoritmo di Dijkstra

Complessità

L'algoritmo effettua N step: ogni nodo viene espanso esattamente una volta. All'interno di uno step si deve:

- Selezionare il nodo non visitato con distanza minima
- Visitare tutti i suoi vicini

Ogni arco viene processato esattamente una volta, ogni nodo viene processato esattamente una volta.

La complessità dell'algoritmo dipende da come viene trovato il vertice non ancora processato con distanza minore. Abbiamo quindi due casi:

- se si scorrono tutti i vertici, la complessità è $\mathcal{O}(N^2 + M) = \mathcal{O}(N^2)$;

Algoritmo di Dijkstra

Complessità

L'algoritmo effettua N step: ogni nodo viene espanso esattamente una volta. All'interno di uno step si deve:

- Selezionare il nodo non visitato con distanza minima
- Visitare tutti i suoi vicini

Ogni arco viene processato esattamente una volta, ogni nodo viene processato esattamente una volta.

La complessità dell'algoritmo dipende da come viene trovato il vertice non ancora processato con distanza minore. Abbiamo quindi due casi:

- se si scorrono tutti i vertici, la complessità è $\mathcal{O}(N^2 + M) = \mathcal{O}(N^2)$;
- se si utilizza una priority queue, la complessità è invece $\mathcal{O}(M \log M)$.

Algoritmo di Bellman-Ford

L'algoritmo di Bellman-Ford ha un costo computazionalmente maggiore di quello di Dijkstra, ma risolve il problema anche nel caso in cui ci sono dei **pesi** degli archi **negativi** e può essere sfruttato per trovare cicli di peso negativo.

Algoritmo di Bellman-Ford

L'algoritmo di Bellman-Ford ha un costo computazionalmente maggiore di quello di Dijkstra, ma risolve il problema anche nel caso in cui ci sono dei **pesi** degli archi **negativi** e può essere sfruttato per trovare cicli di peso negativo.

Osservazione

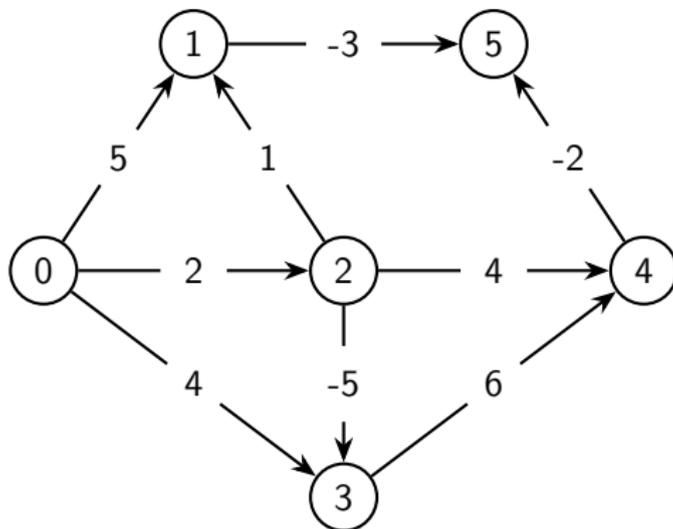
Se un grafo non orientato ha pesi negativi in alcuni archi, allora avrà sempre "cicli" di peso negativo.

Quindi ci interessiamo del caso orientato.

Algoritmo di Bellman-Ford

Algoritmo

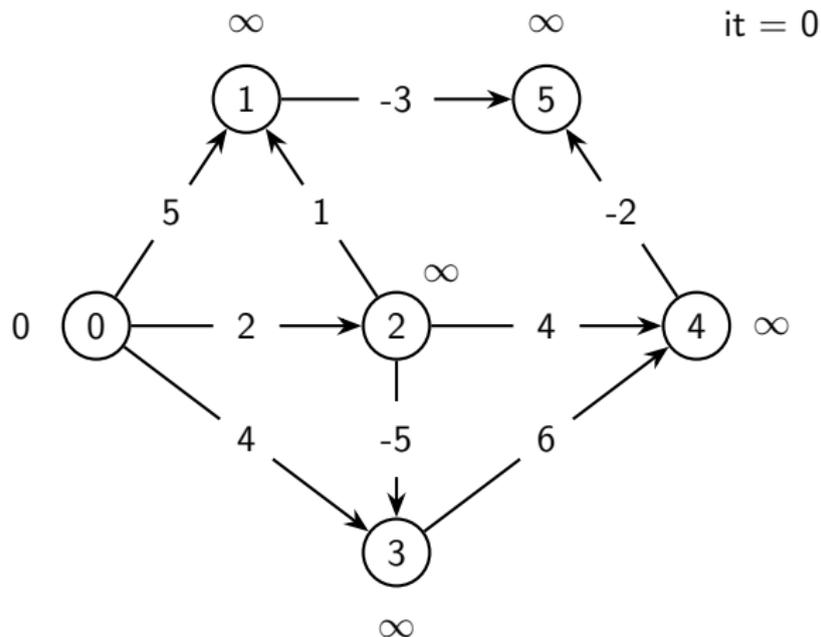
Vogliamo calcolare la distanza minima dal nodo 0 verso tutti gli altri nodi.



Algoritmo di Bellman-Ford

Algoritmo

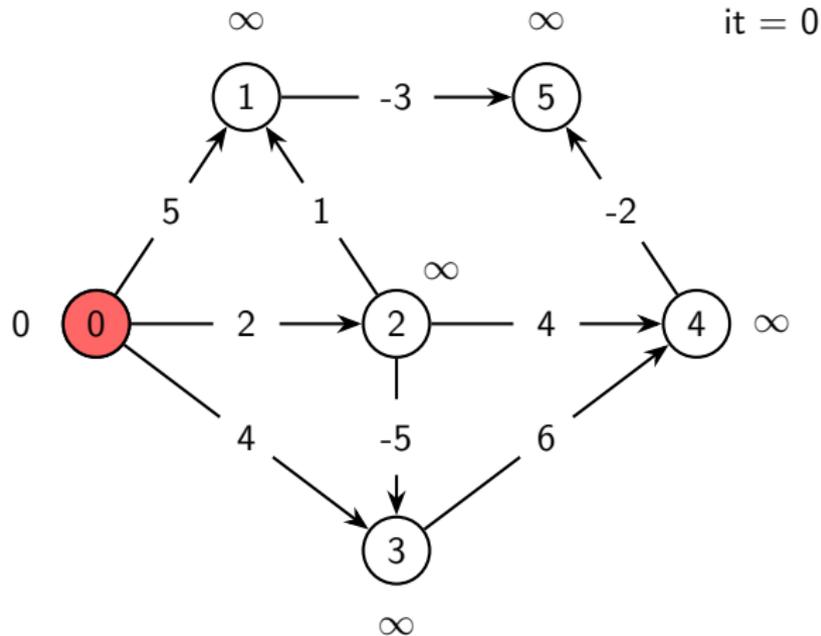
Inizializzazione: la distanza di tutti i nodi è ∞ , **tranne per la sorgente**.



Algoritmo di Bellman-Ford

Algoritmo

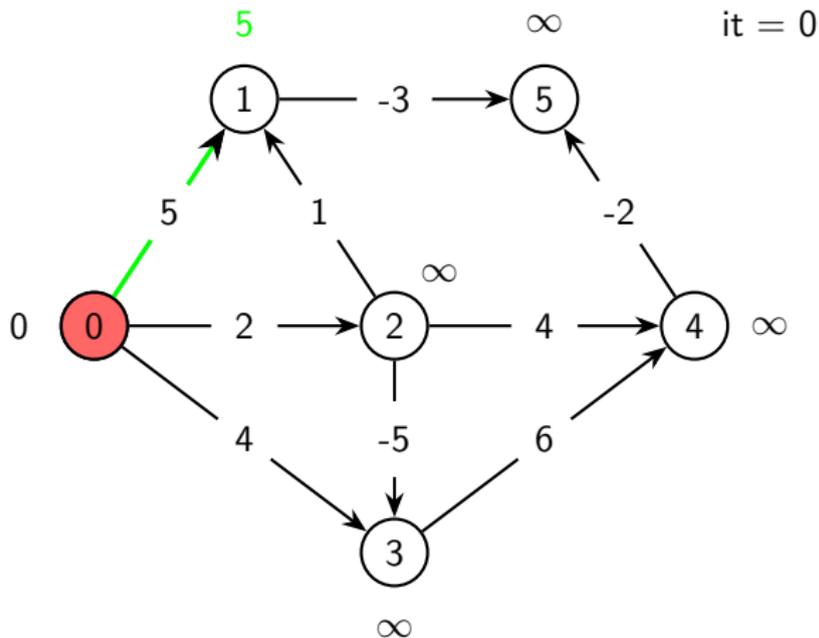
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

Algoritmo

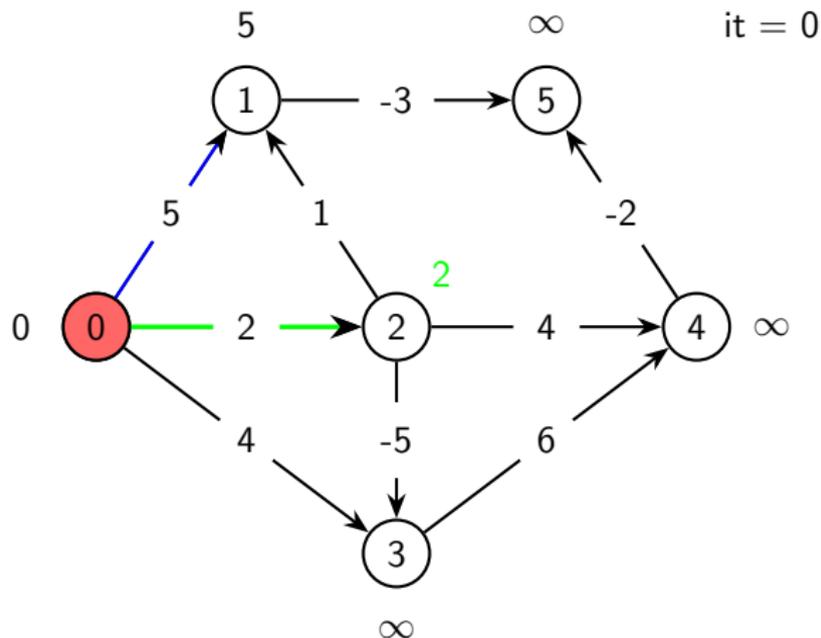
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

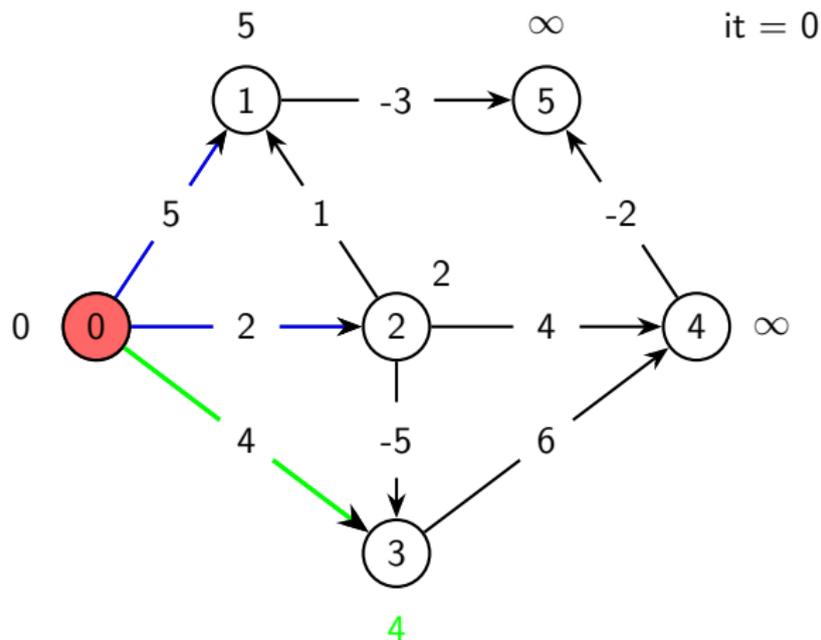
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

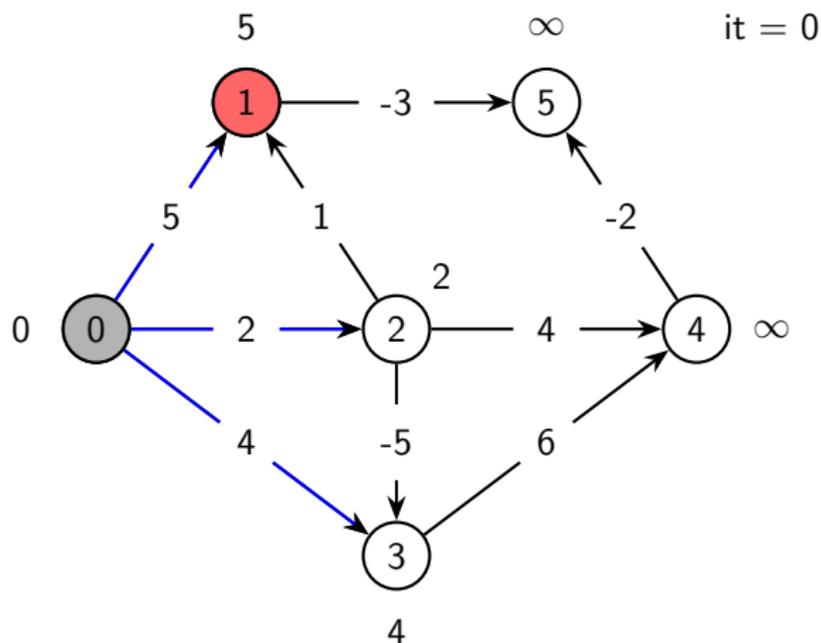
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

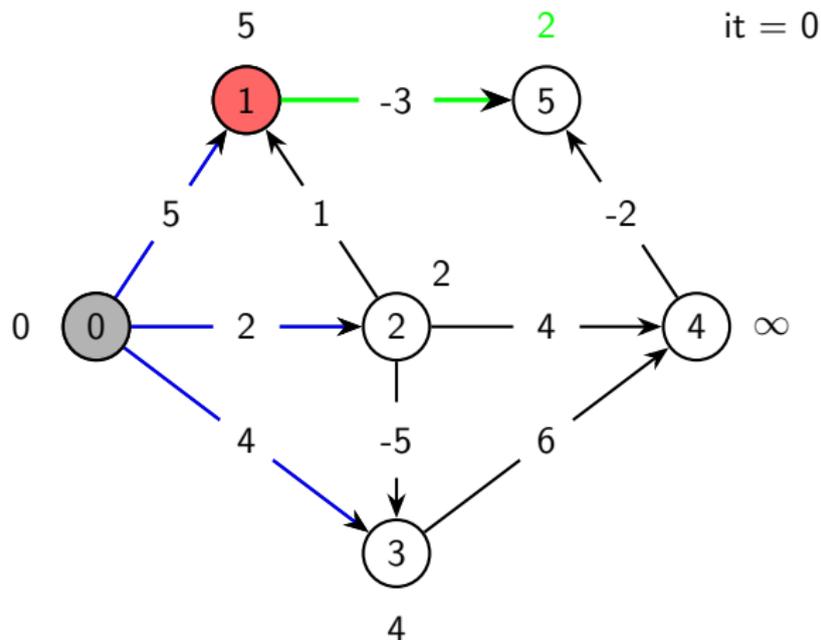
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

Algoritmo

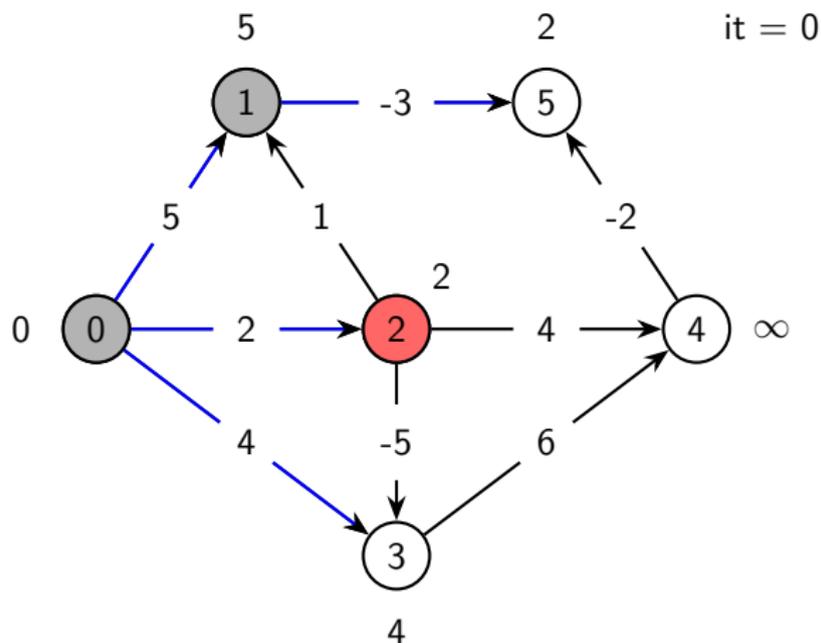
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

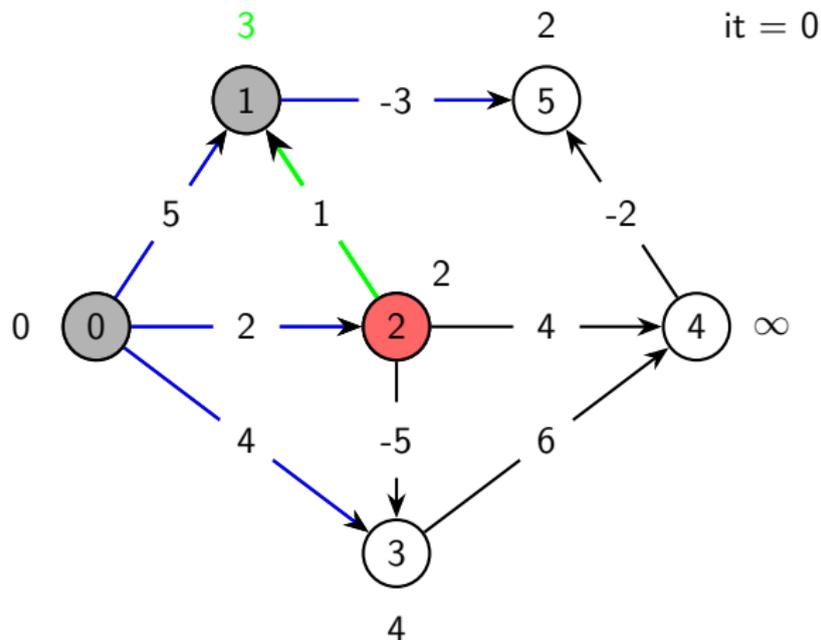
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

Algoritmo

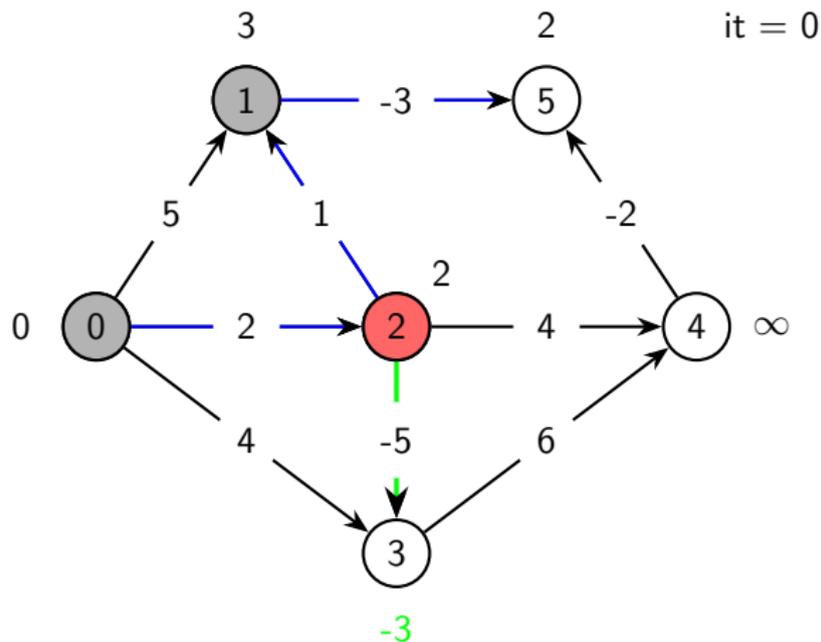
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

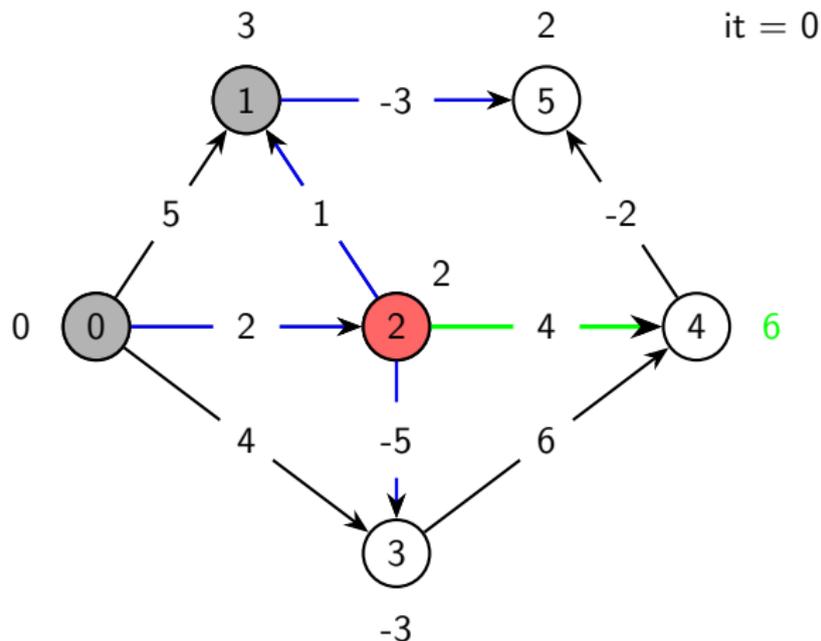
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

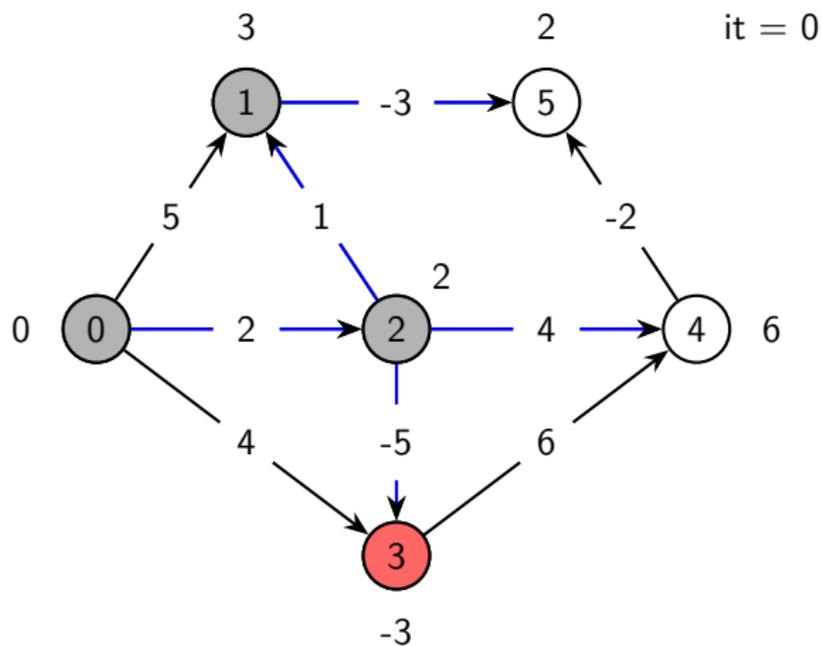
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

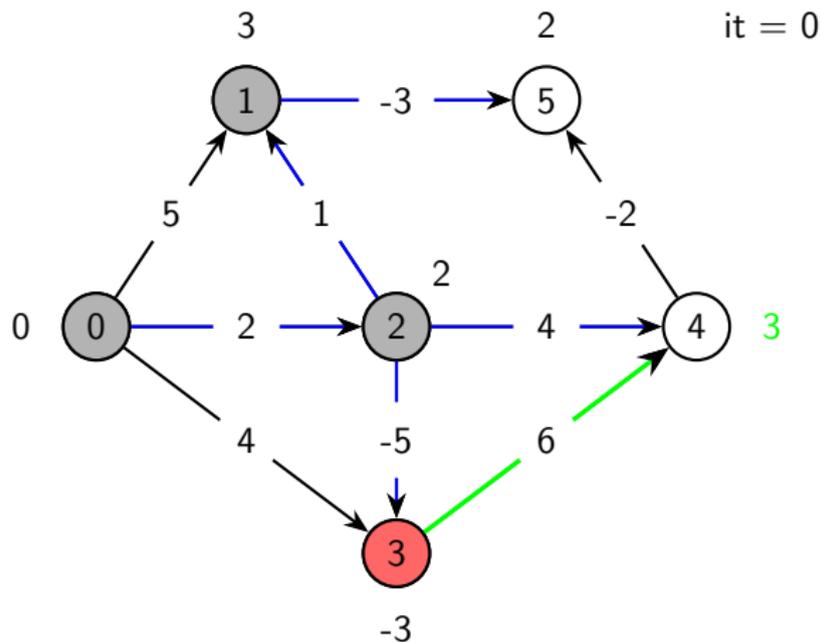
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

Algoritmo

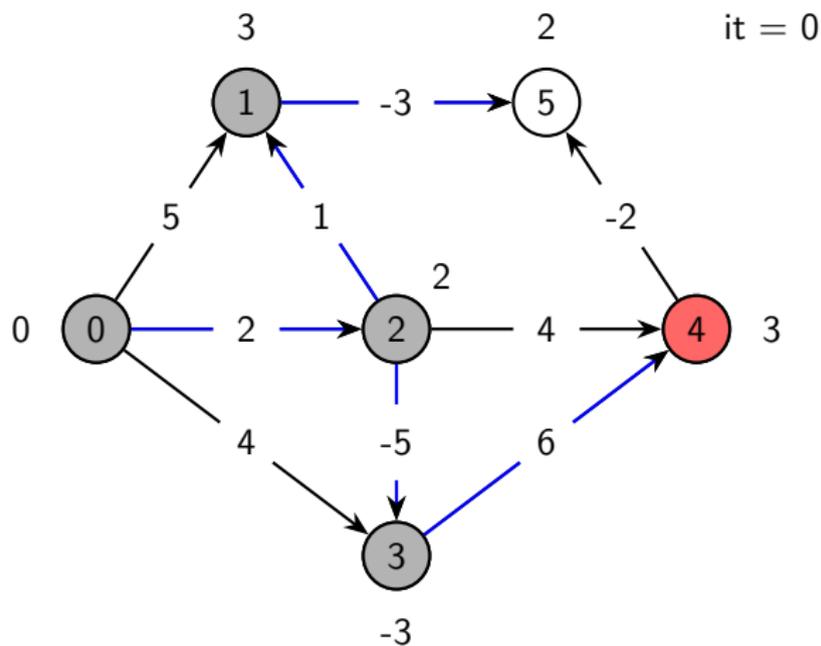
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

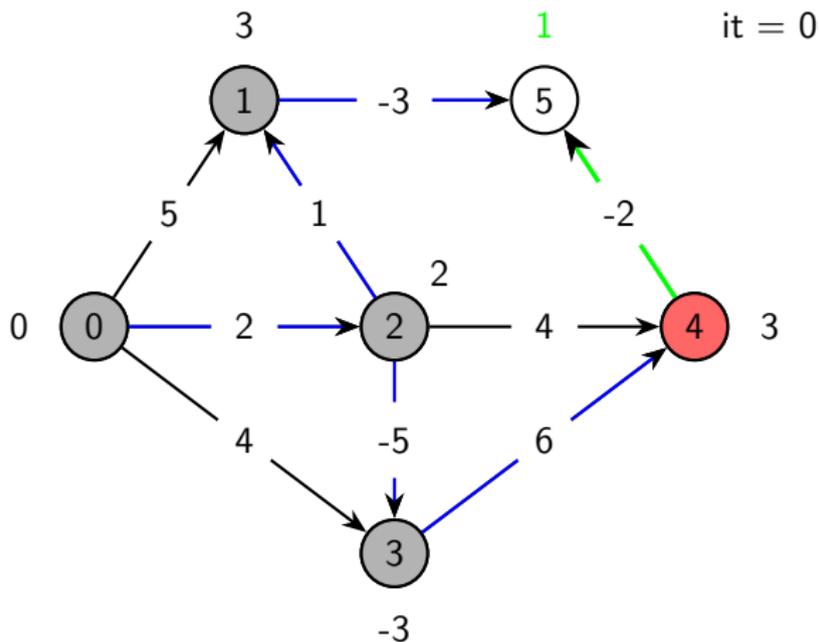
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

Algoritmo

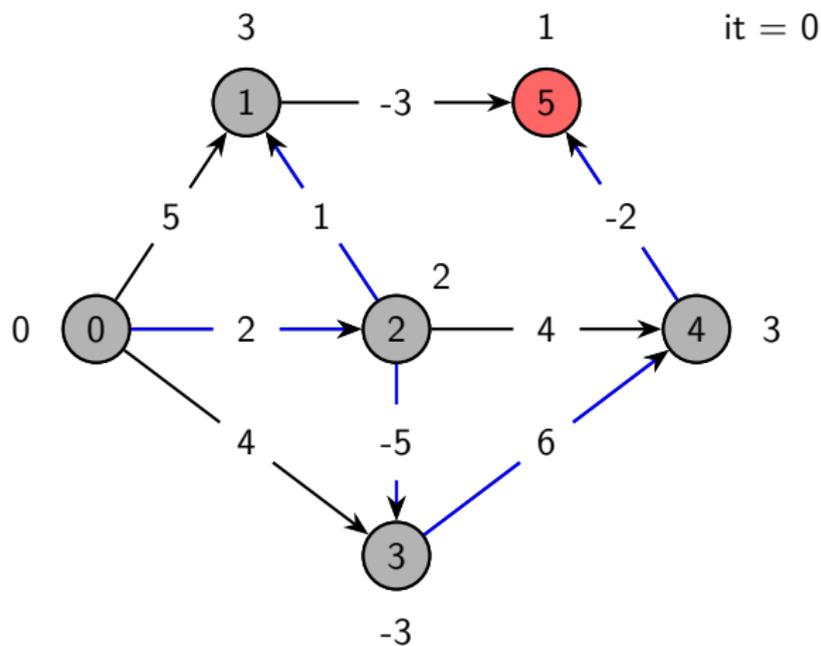
Espandiamo come nelle altre visite ogni arco, aggiornando la distanza se necessario.



Algoritmo di Bellman-Ford

Algoritmo

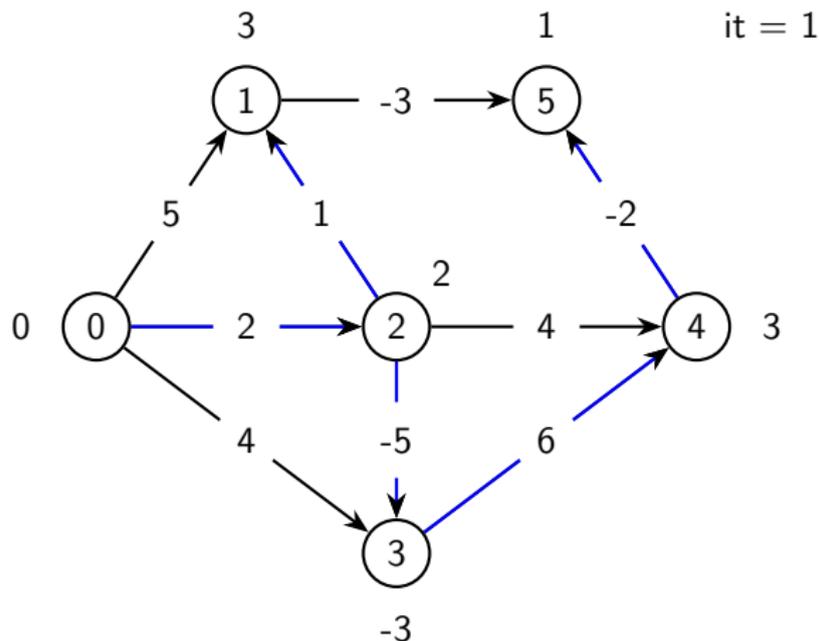
Processiamo in un qualunque ordine tutti i nodi



Algoritmo di Bellman-Ford

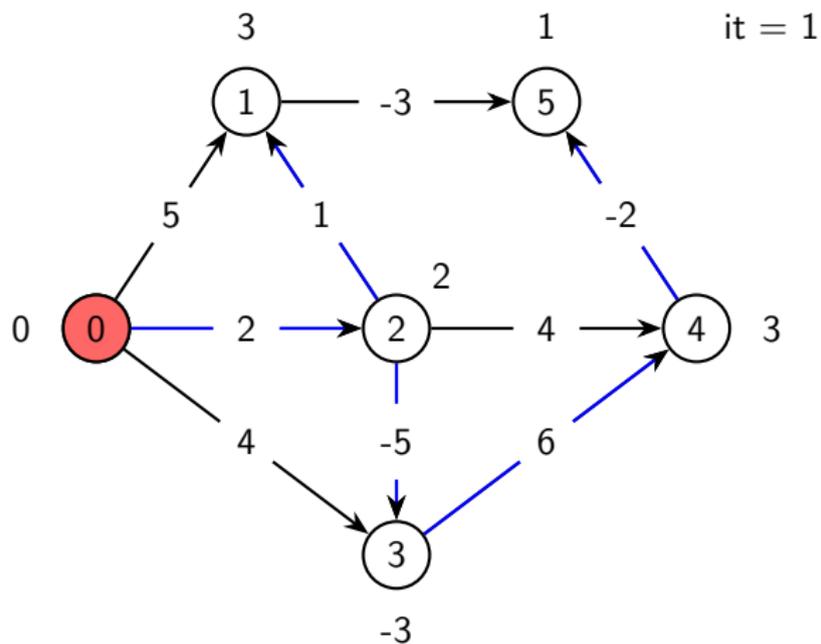
Algoritmo

La prima iterazione si è conclusa. Ora si riparte dal primo nodo e si ripete la visita di tutti i nodi.



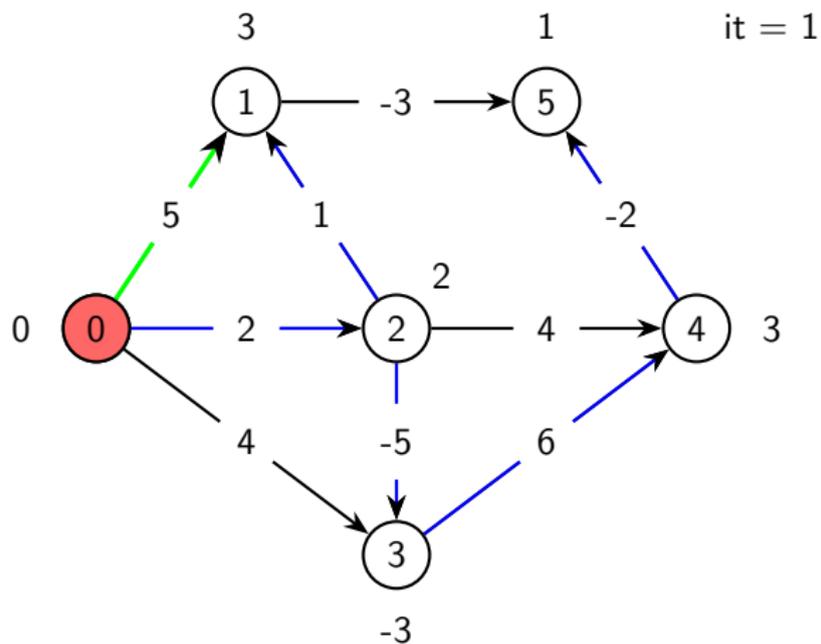
Algoritmo di Bellman-Ford

Algoritmo



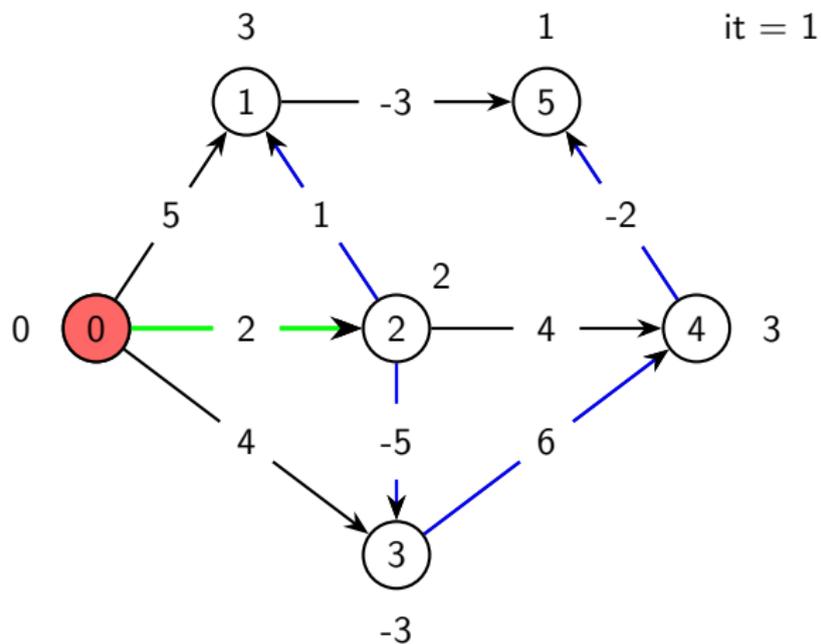
Algoritmo di Bellman-Ford

Algoritmo



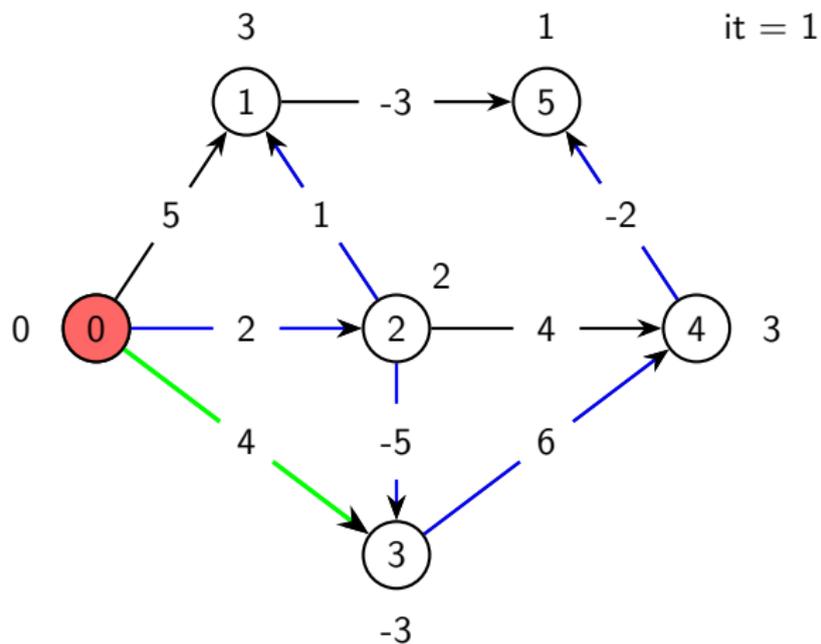
Algoritmo di Bellman-Ford

Algoritmo



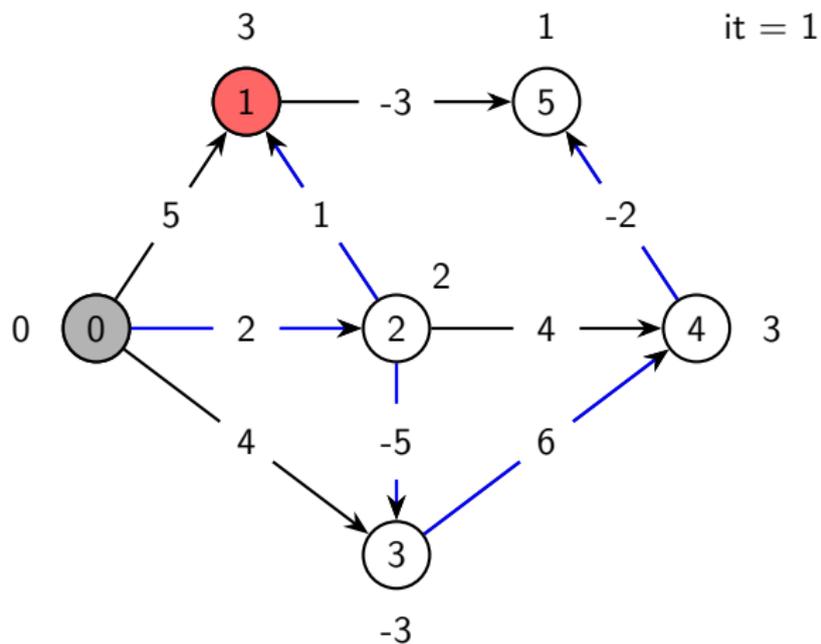
Algoritmo di Bellman-Ford

Algoritmo



Algoritmo di Bellman-Ford

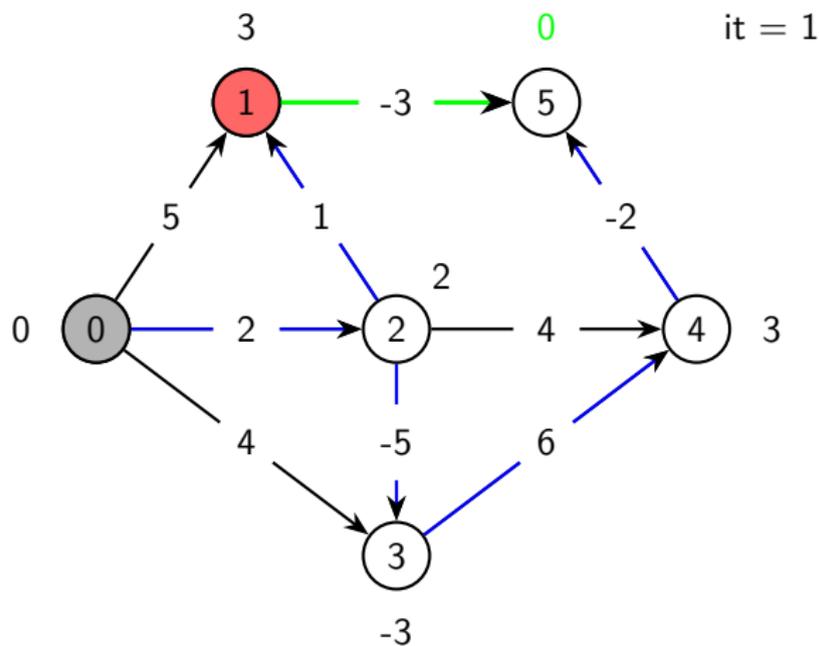
Algoritmo



Algoritmo di Bellman-Ford

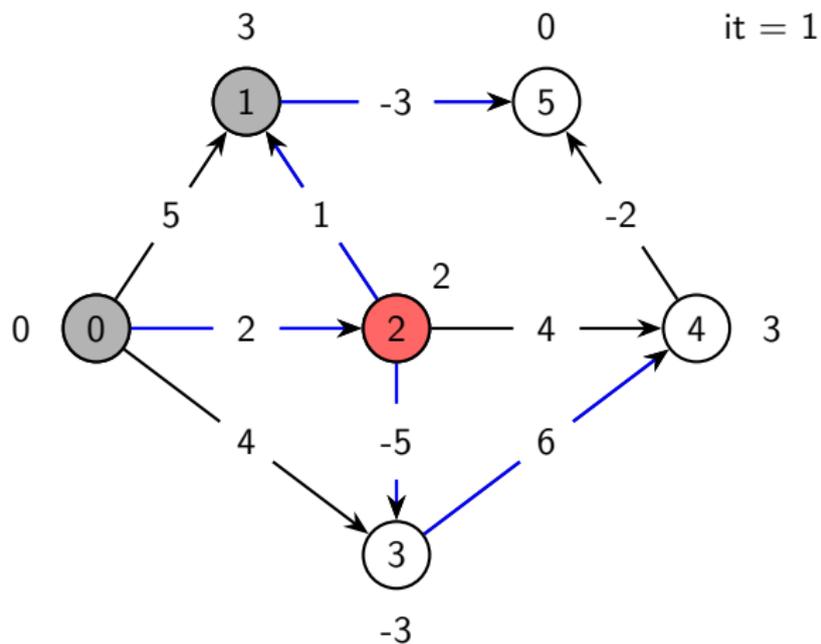
Algoritmo

Alcune distanze potrebbero cambiare.



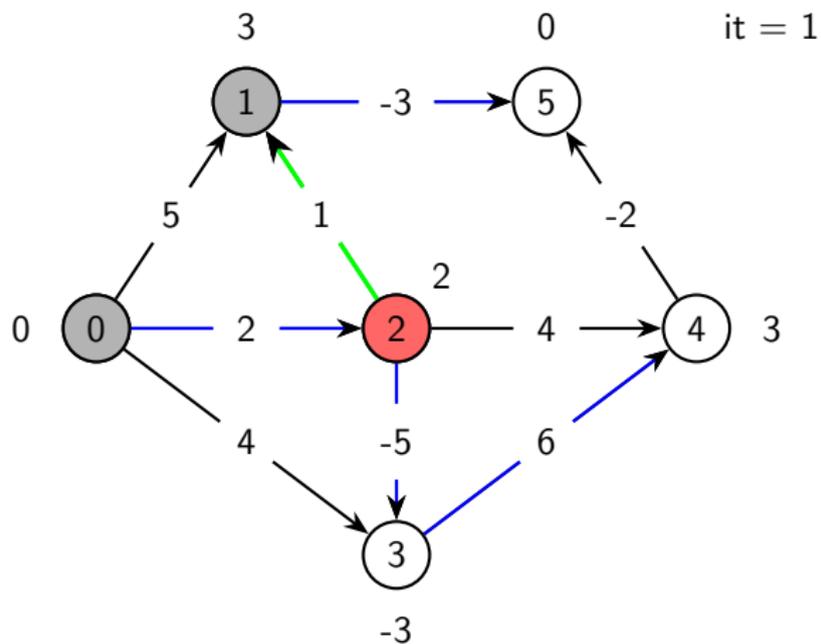
Algoritmo di Bellman-Ford

Algoritmo



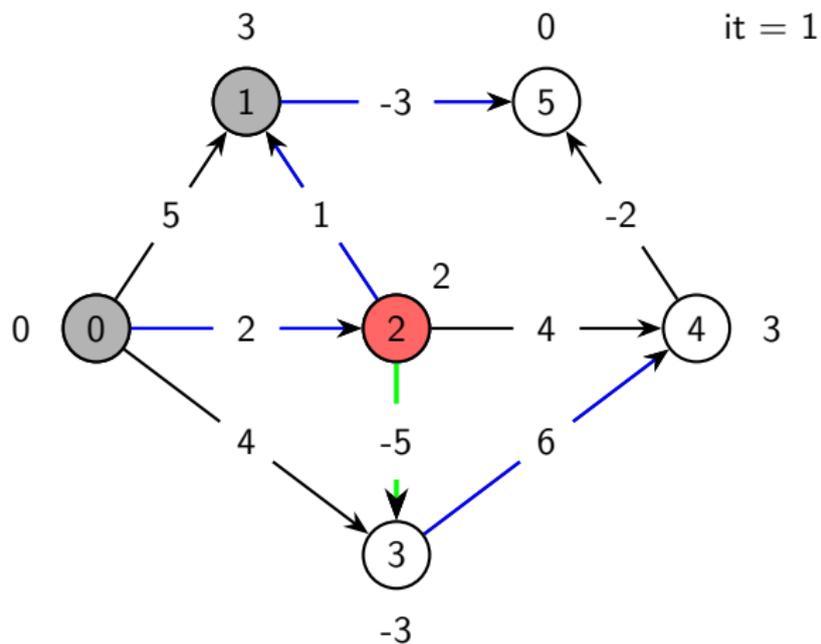
Algoritmo di Bellman-Ford

Algoritmo



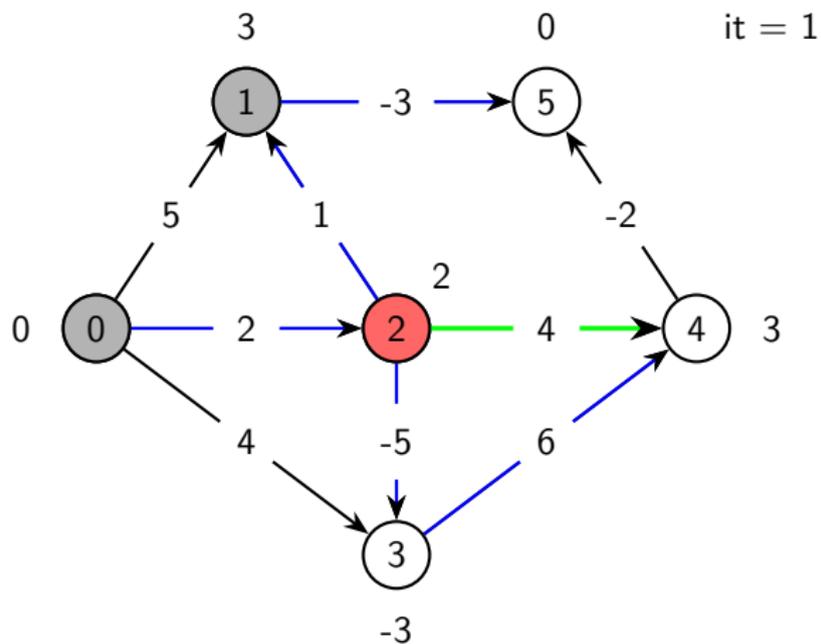
Algoritmo di Bellman-Ford

Algoritmo



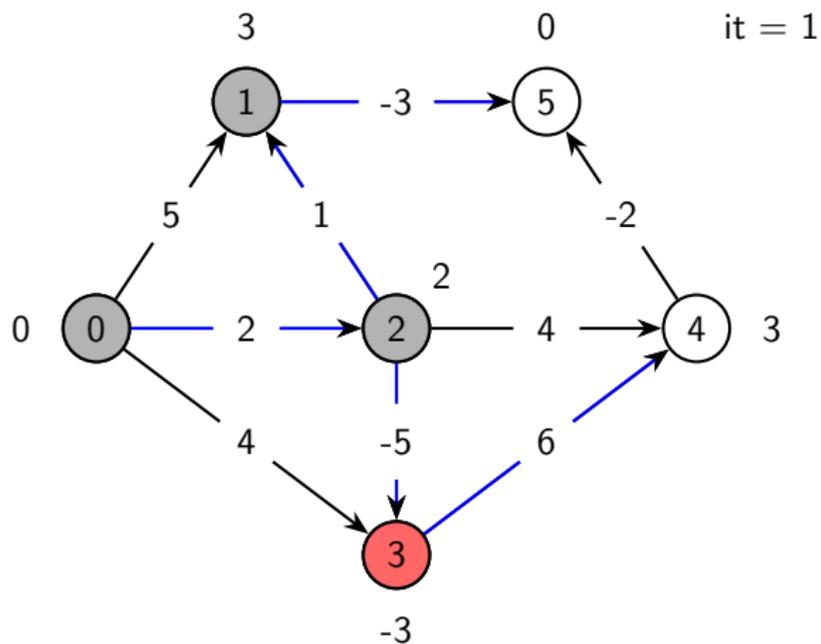
Algoritmo di Bellman-Ford

Algoritmo



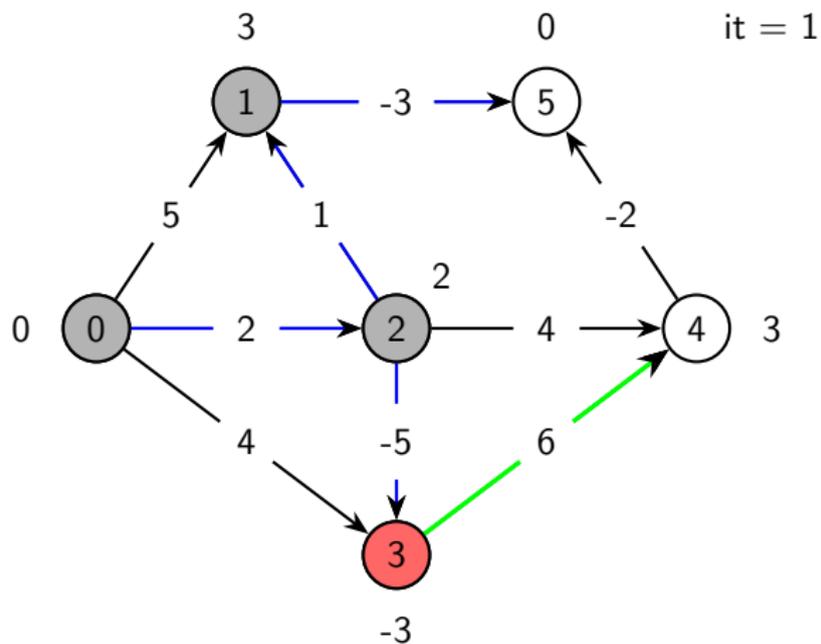
Algoritmo di Bellman-Ford

Algoritmo



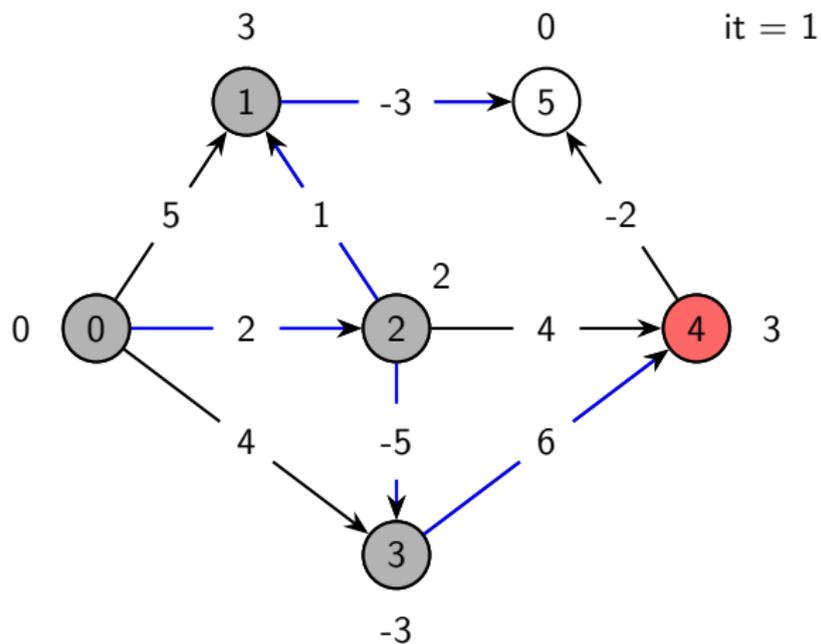
Algoritmo di Bellman-Ford

Algoritmo



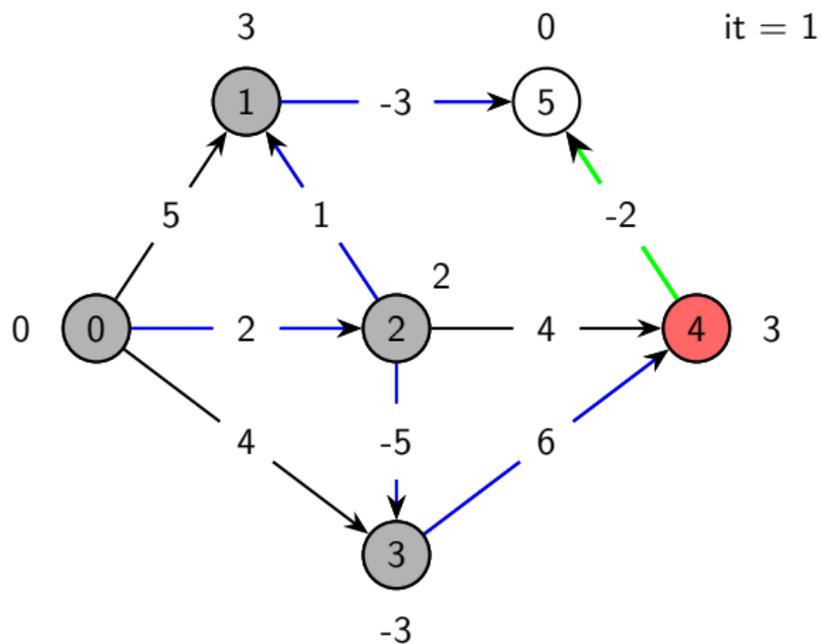
Algoritmo di Bellman-Ford

Algoritmo



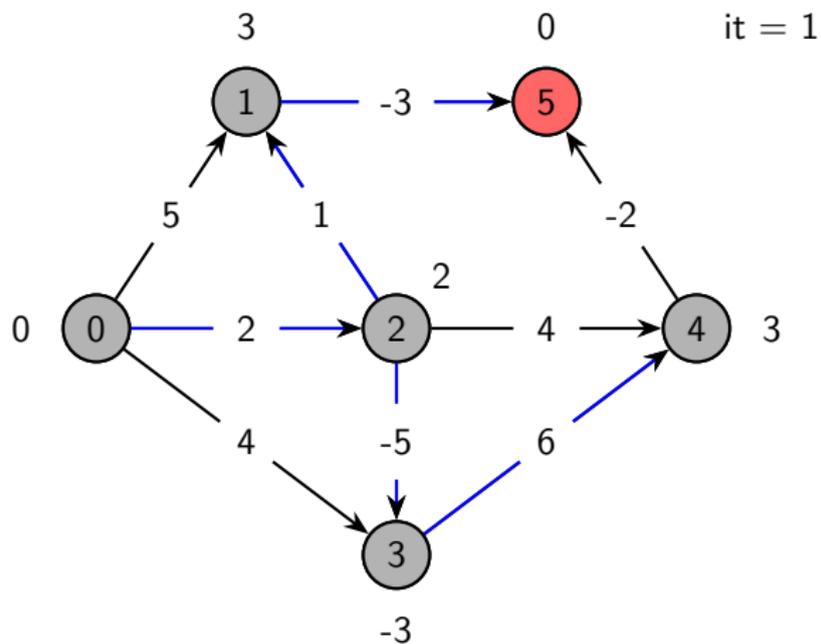
Algoritmo di Bellman-Ford

Algoritmo



Algoritmo di Bellman-Ford

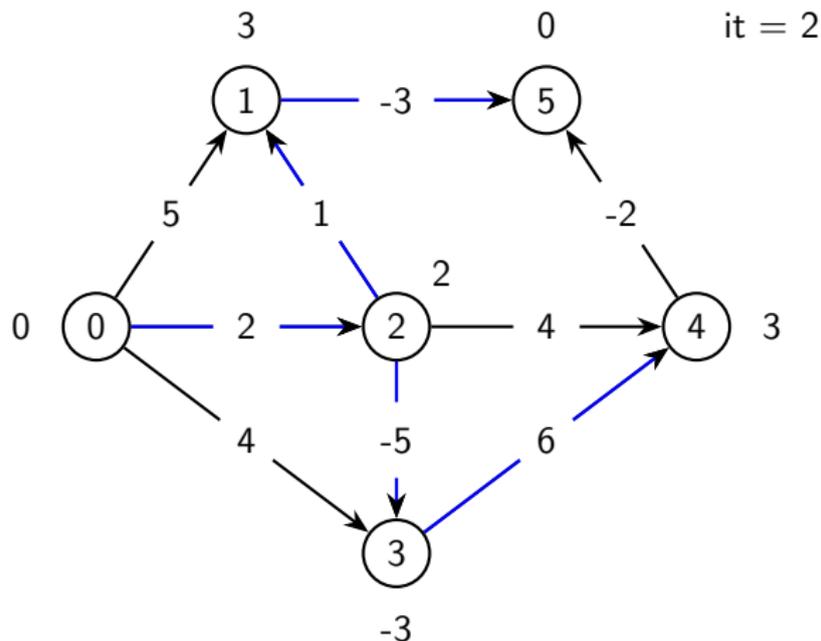
Algoritmo



Algoritmo di Bellman-Ford

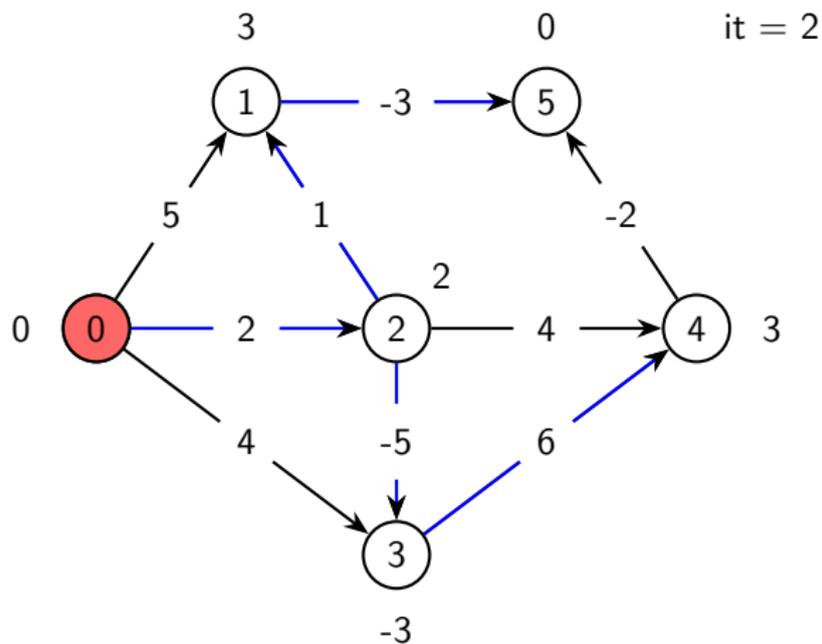
Algoritmo

Visto che ci sono stati dei cambiamenti è necessario ripetere la visita di tutti i nodi.



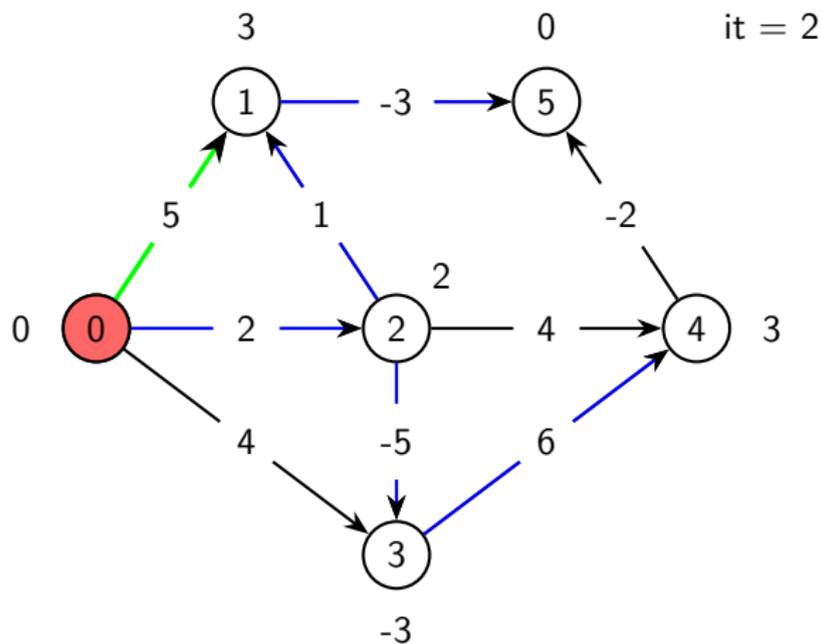
Algoritmo di Bellman-Ford

Algoritmo



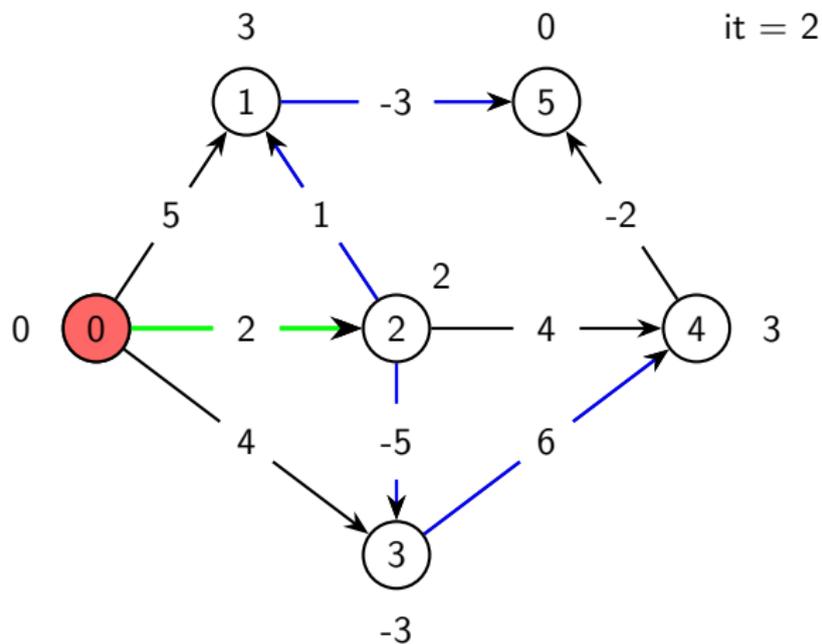
Algoritmo di Bellman-Ford

Algoritmo



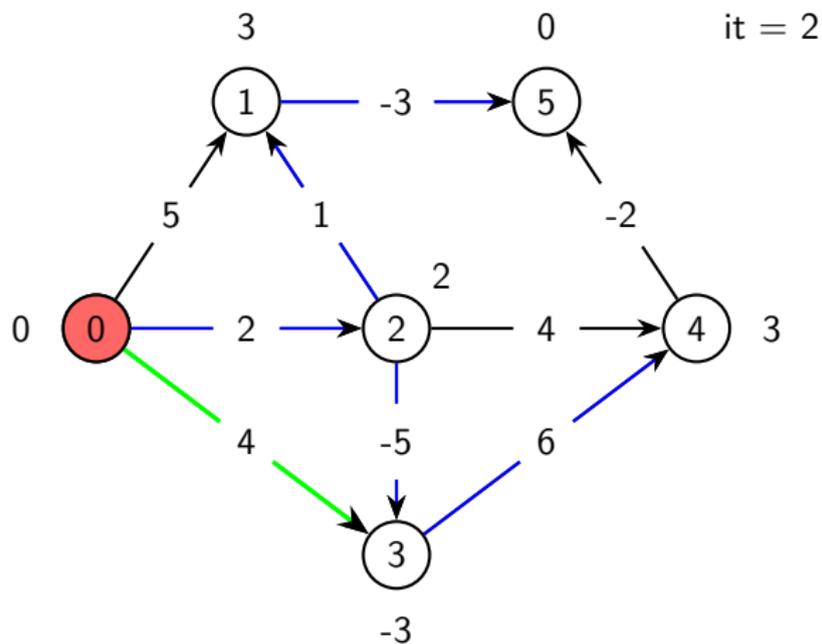
Algoritmo di Bellman-Ford

Algoritmo



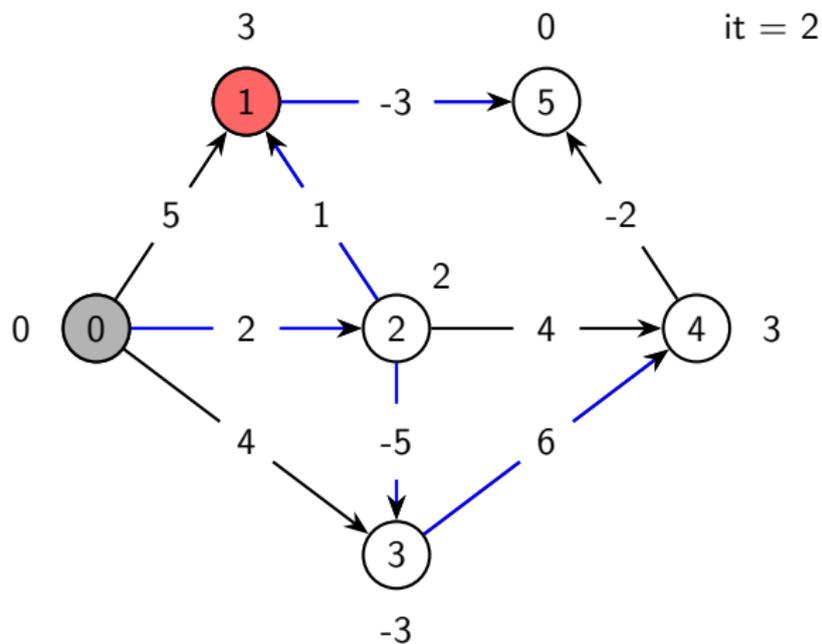
Algoritmo di Bellman-Ford

Algoritmo



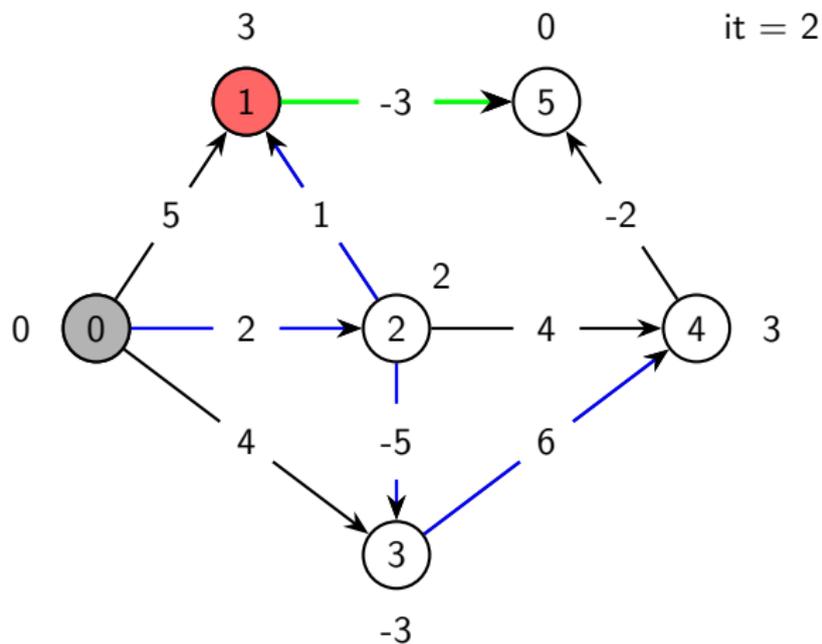
Algoritmo di Bellman-Ford

Algoritmo



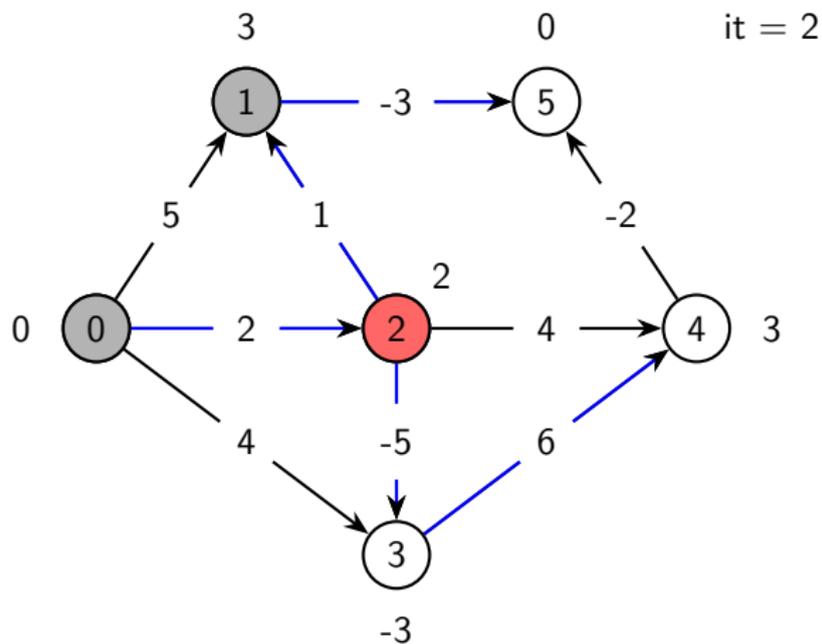
Algoritmo di Bellman-Ford

Algoritmo



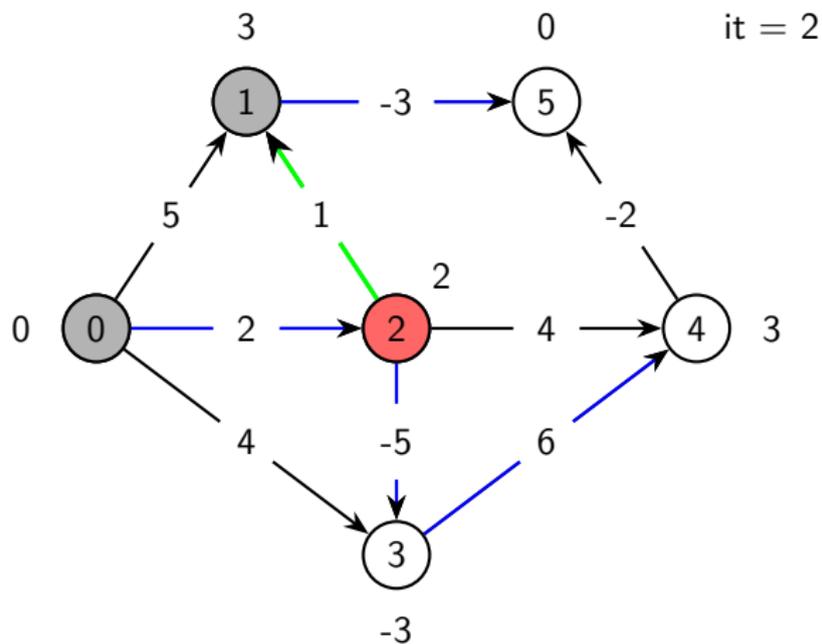
Algoritmo di Bellman-Ford

Algoritmo



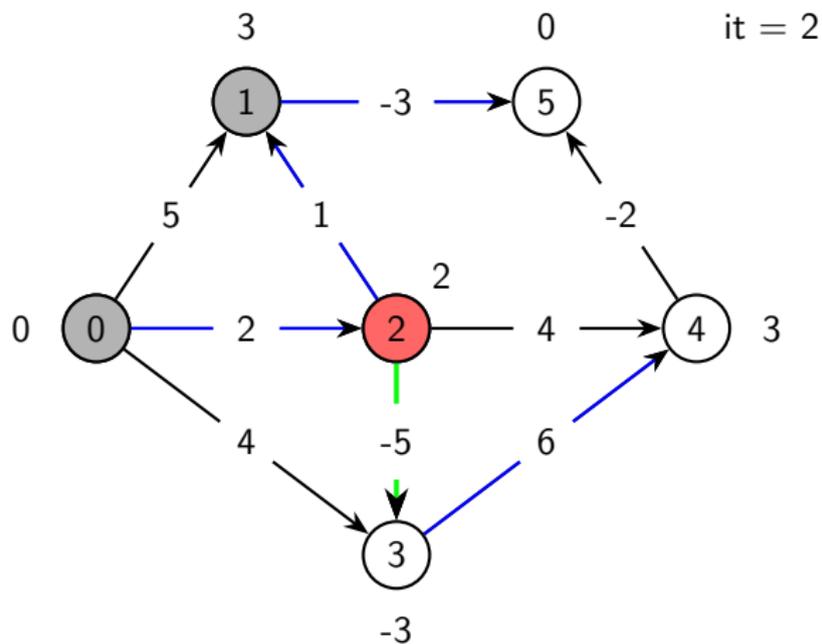
Algoritmo di Bellman-Ford

Algoritmo



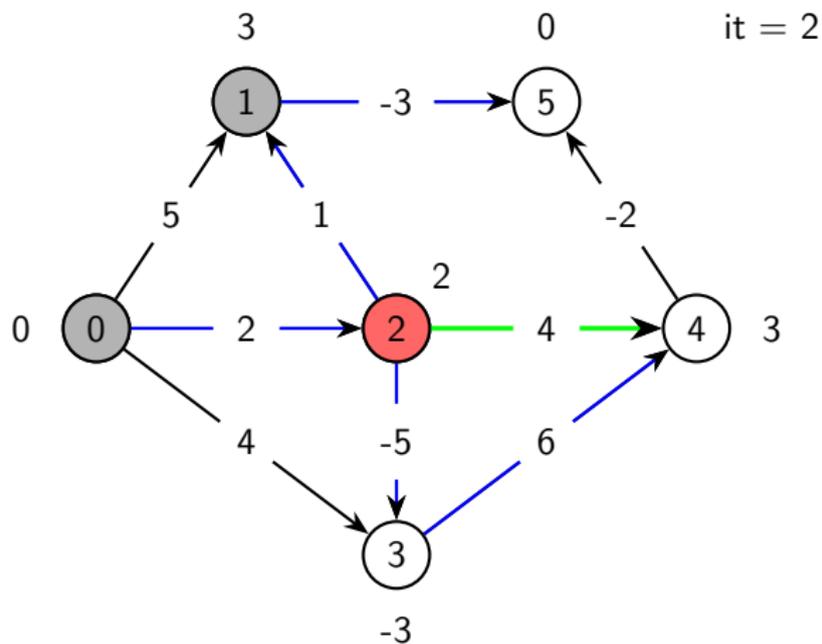
Algoritmo di Bellman-Ford

Algoritmo



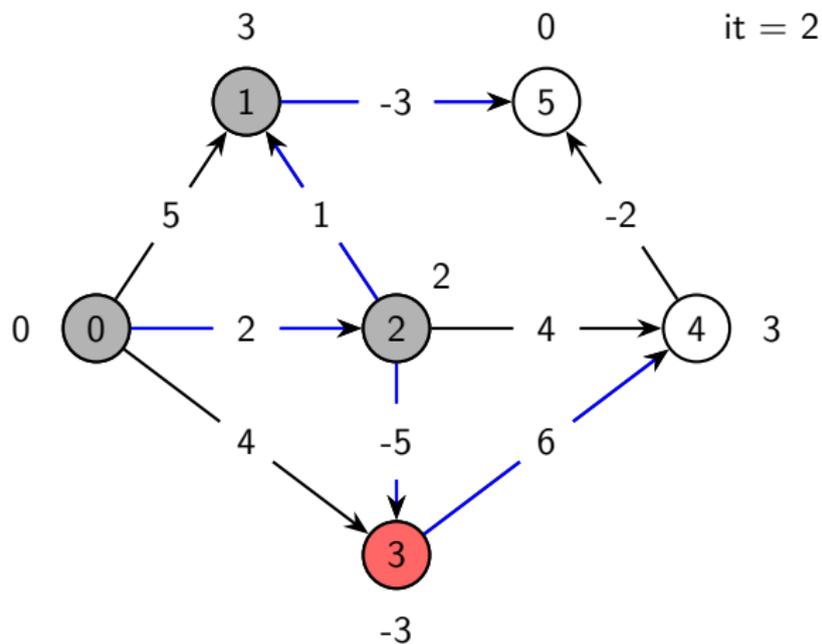
Algoritmo di Bellman-Ford

Algoritmo



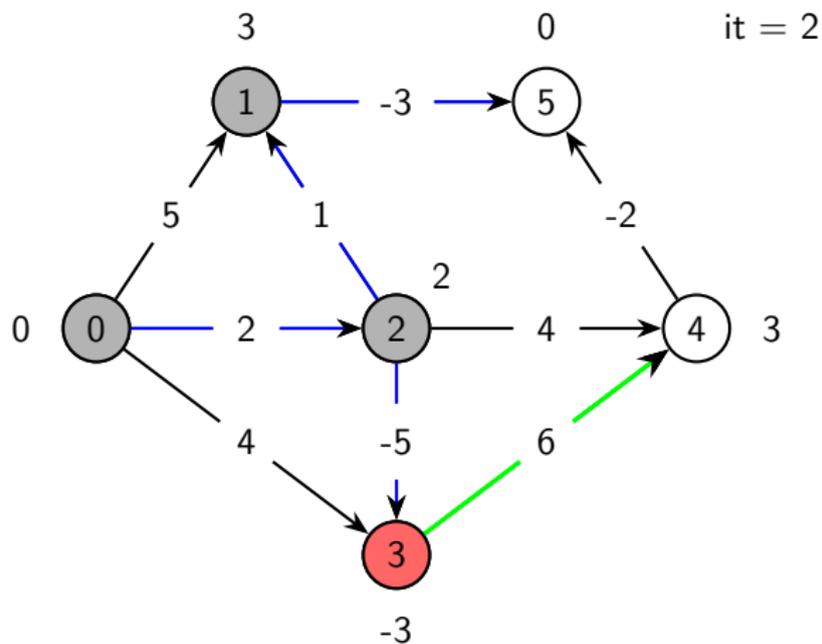
Algoritmo di Bellman-Ford

Algoritmo



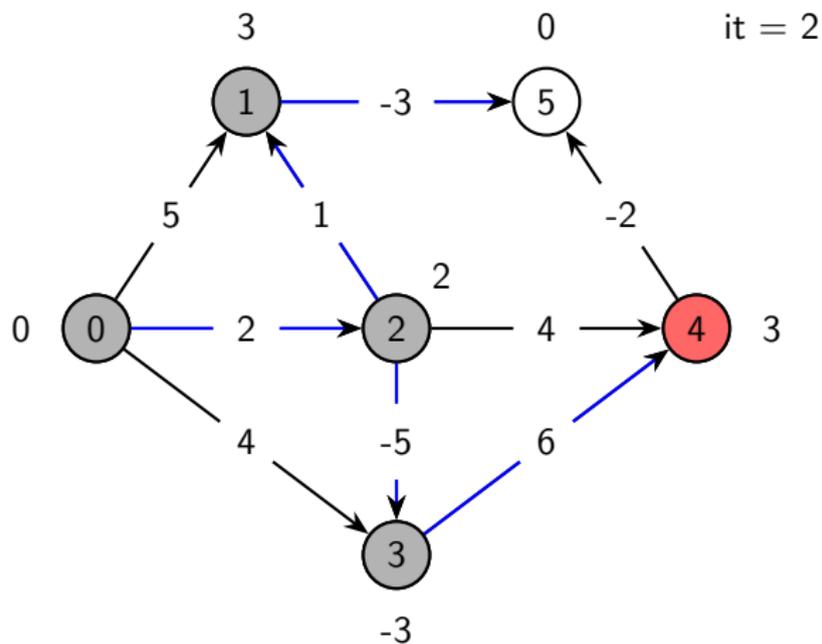
Algoritmo di Bellman-Ford

Algoritmo



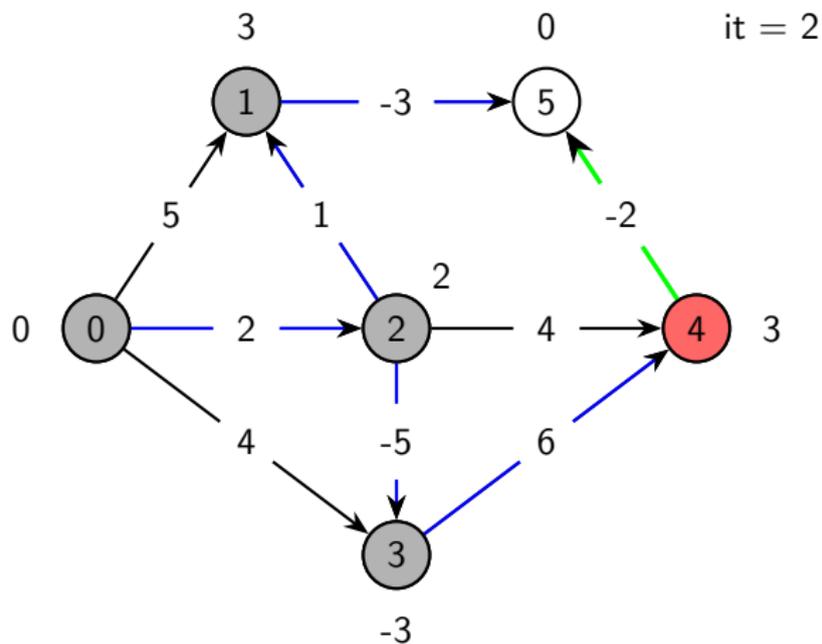
Algoritmo di Bellman-Ford

Algoritmo



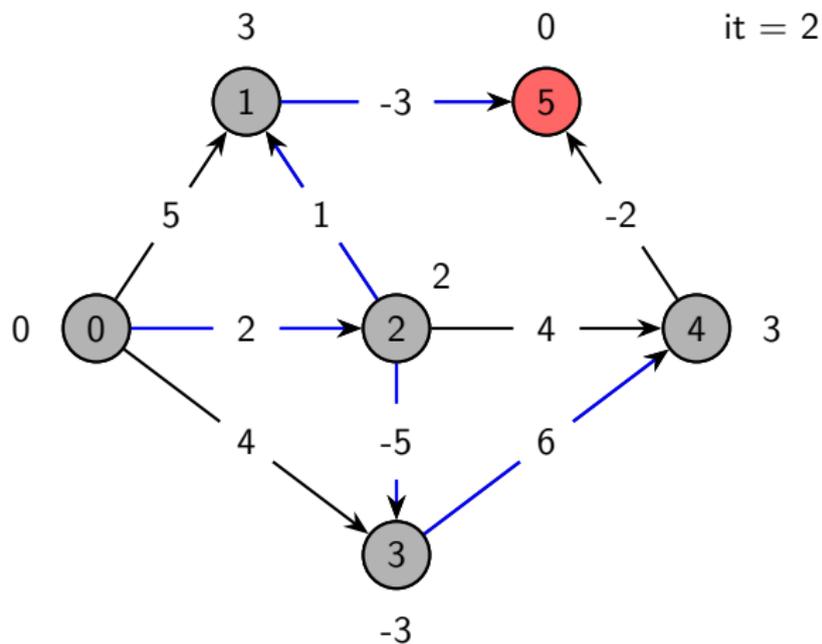
Algoritmo di Bellman-Ford

Algoritmo



Algoritmo di Bellman-Ford

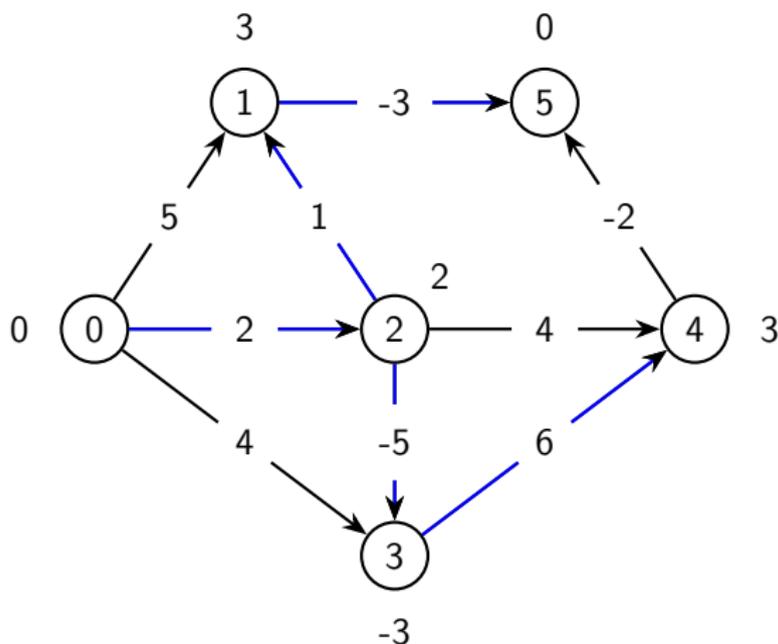
Algoritmo



Algoritmo di Bellman-Ford

Algoritmo

Durante questa iterazione non ci sono stati cambiamenti: l'algoritmo termina.



Algoritmo di Bellman-Ford

Complessità

L'algoritmo di Bellman-Ford termina quando in un'iterazione non ci sono stati aggiornamenti alle distanze.

Algoritmo di Bellman-Ford

Complessità

L'algoritmo di Bellman-Ford termina quando in un'iterazione non ci sono stati aggiornamenti alle distanze.

Se non ci sono cicli di lunghezza negativa questo succede sicuramente prima dell' N -esima iterazione, se cioè non accade vuol dire che siamo in presenza di un ciclo di lunghezza negativa e ci possiamo fermare perché ci saranno vertici a distanza $-\infty$.

Algoritmo di Bellman-Ford

Complessità

L'algoritmo di Bellman-Ford termina quando in un'iterazione non ci sono stati aggiornamenti alle distanze.

Se non ci sono cicli di lunghezza negativa questo succede sicuramente prima dell' N -esima iterazione, se cioè non accade vuol dire che siamo in presenza di un ciclo di lunghezza negativa e ci possiamo fermare perché ci saranno vertici a distanza $-\infty$.

La complessità è dunque $\mathcal{O}(NM)$, poiché ogni iterazione richiede tempo $\mathcal{O}(M)$.

Algoritmo di Floyd-Warshall

L'algoritmo di Floyd-Warshall risolve un problema più generale dei precedenti, cioè trovare le distanze fra *tutte* le coppie di nodi in un grafo senza cicli di lunghezza negativa.

Algoritmo di Floyd-Warshall

L'algoritmo di Floyd-Warshall risolve un problema più generale dei precedenti, cioè trovare le distanze fra *tutte* le coppie di nodi in un grafo senza cicli di lunghezza negativa.

Operazione triangolare

Per ogni nodo h , e per ogni coppia di nodi i, j controlla se il percorso $i \rightarrow h \rightarrow j$ conviene rispetto al percorso $i \rightarrow j$.

Per ricostruire la soluzione possiamo ricordarci, per ogni coppia di nodi, quale arco prendere.

Algoritmo di Floyd-Warshall

$$D_{ij} = \infty, N_{ij} = -1 \quad \forall i, j$$

foreach *edge* (i, j) **do**

$$| \quad D_{ij} = w_{ij}$$

$$| \quad N_{ij} = j$$

end

foreach *node* i **do**

$$| \quad D_{ii} = 0$$

end

Algoritmo di Floyd-Warshall

$$D_{ij} = \infty, N_{ij} = -1 \quad \forall i, j$$

foreach *edge* (*i*, *j*) **do**

| $D_{ij} = w_{ij}$

| $N_{ij} = j$

end

foreach *node* *i* **do**

| $D_{ii} = 0$

end

for $h \leftarrow 0$ **to** $N - 1$ **do**

|

end

Algoritmo di Floyd-Warshall

$$D_{ij} = \infty, N_{ij} = -1 \quad \forall i, j$$

foreach *edge* (i, j) **do**

| $D_{ij} = w_{ij}$

| $N_{ij} = j$

end

foreach *node* i **do**

| $D_{ii} = 0$

end

for $h \leftarrow 0$ **to** $N - 1$ **do**

| **for** $i \leftarrow 0$ **to** $N - 1$ **do**

| | **for** $j \leftarrow 0$ **to** $N - 1$ **do**

| | | **end**

| | **end**

end

Algoritmo di Floyd-Warshall

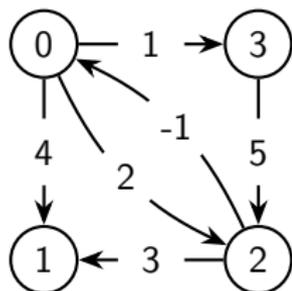
```

 $D_{ij} = \infty, N_{ij} = -1 \quad \forall i, j$ 
foreach edge  $(i, j)$  do
  |  $D_{ij} = w_{ij}$ 
  |  $N_{ij} = j$ 
end
foreach node  $i$  do
  |  $D_{ii} = 0$ 
end
for  $h \leftarrow 0$  to  $N - 1$  do
  | for  $i \leftarrow 0$  to  $N - 1$  do
  | | for  $j \leftarrow 0$  to  $N - 1$  do
  | | | if  $D_{ij} > D_{ih} + D_{hj}$  then
  | | | |  $D_{ij} = D_{ih} + D_{hj}$ 
  | | | |  $N_{ij} = N_{ih}$ 
  | | | end
  | | end
  | end
end

```

Algoritmo di Floyd-Warshall

Algoritmo

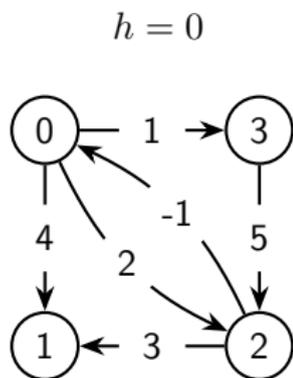


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

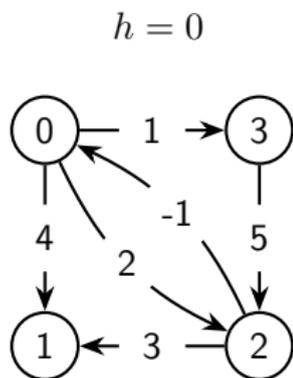


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

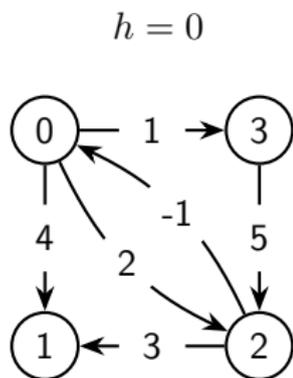


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

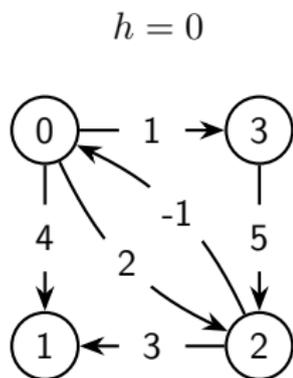


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

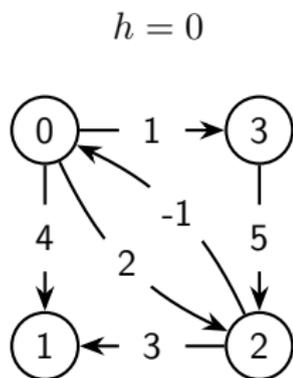


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

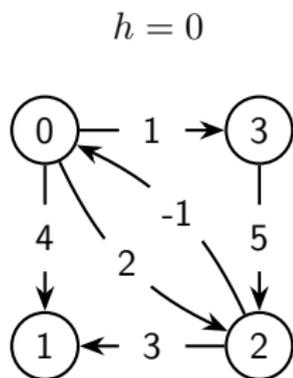


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

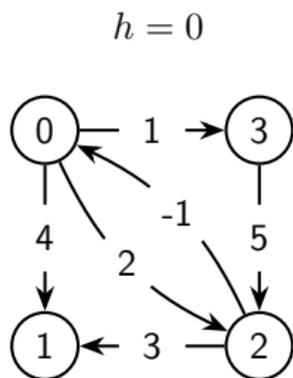


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	∞
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	-1
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

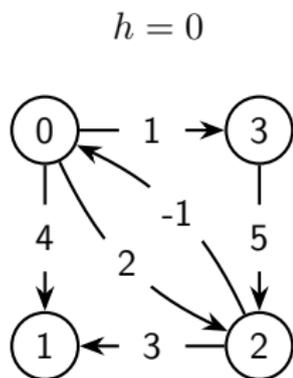


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

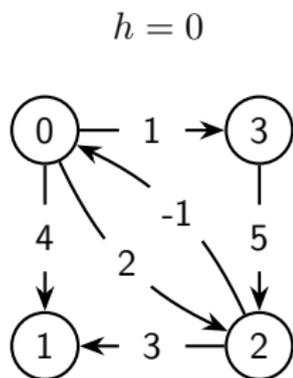


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

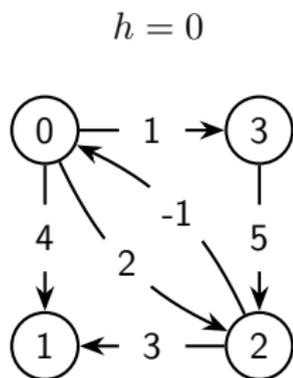


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

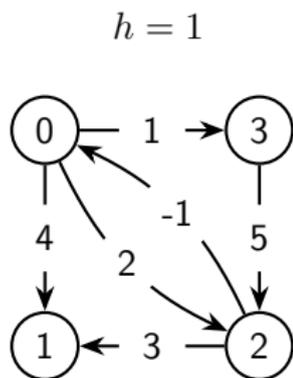


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

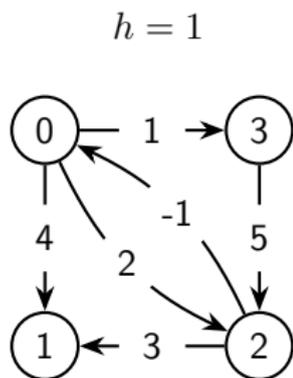


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

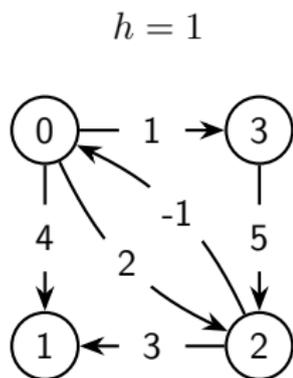


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

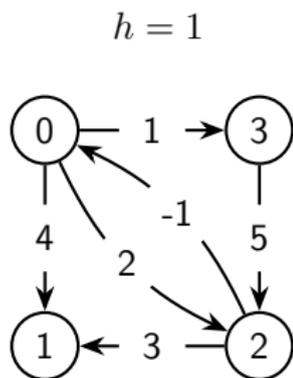


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

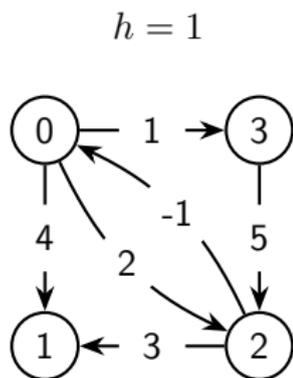


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

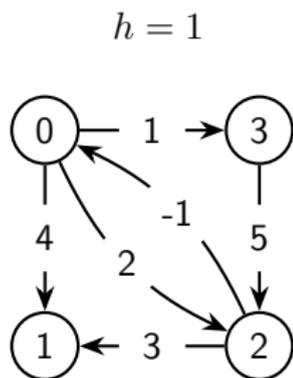


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

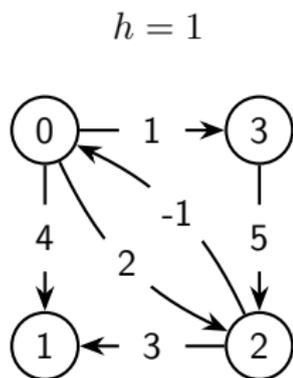


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

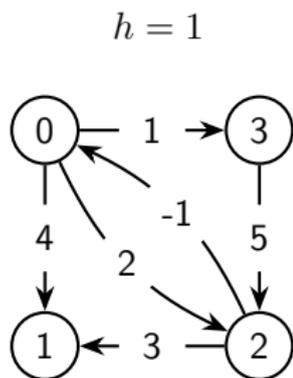


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

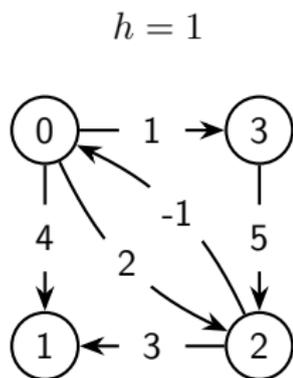


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

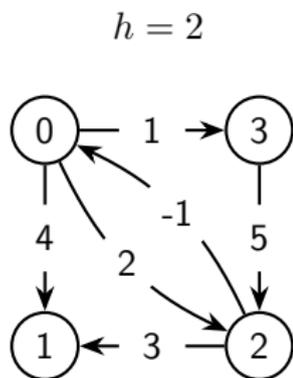


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

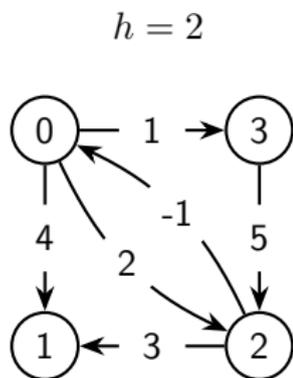


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

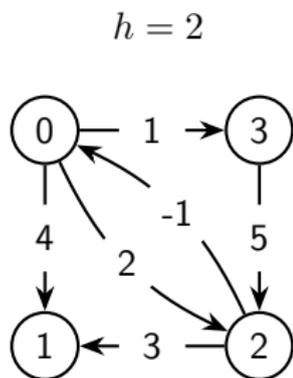


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

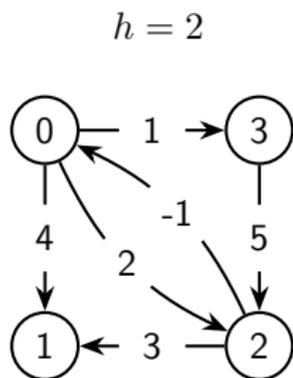


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

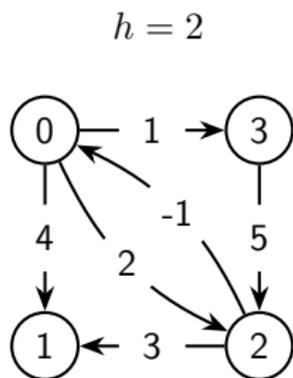


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

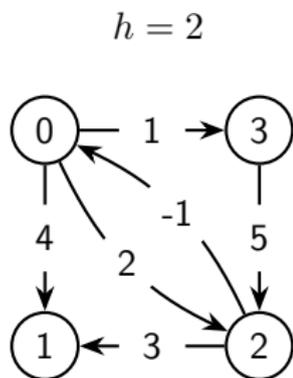


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

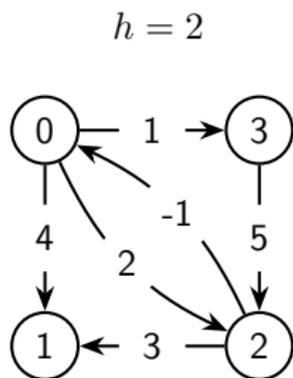


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

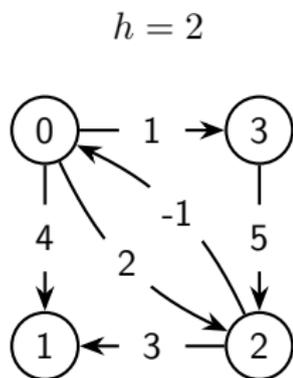


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	∞	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	-1	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

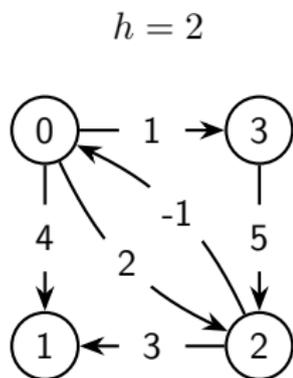


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

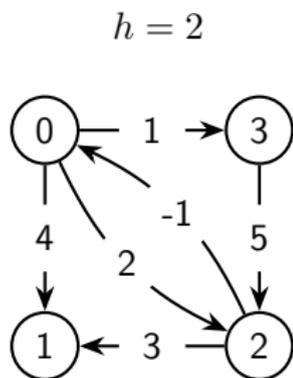


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	∞	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	-1	2	3

Algoritmo di Floyd-Warshall

Algoritmo

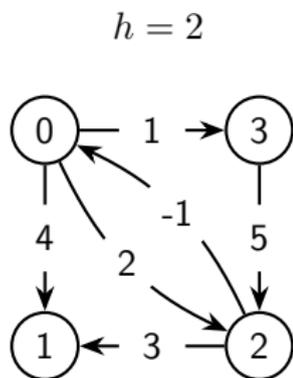


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

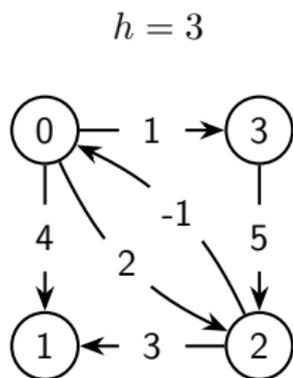


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

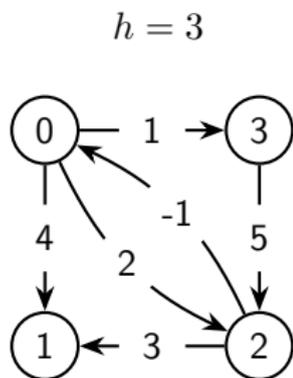


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

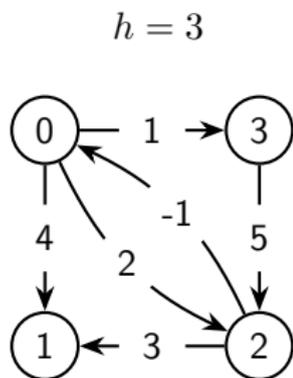


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

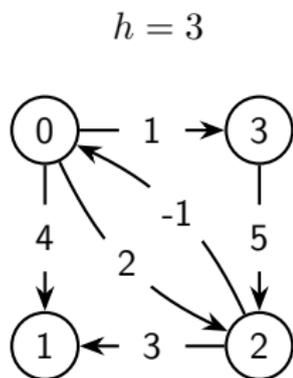


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

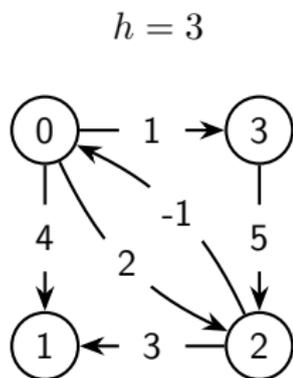


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

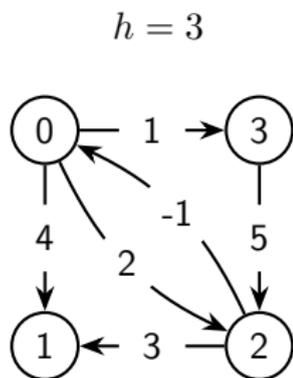


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

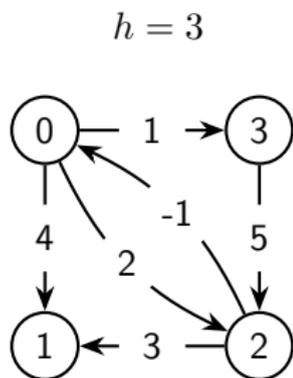


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

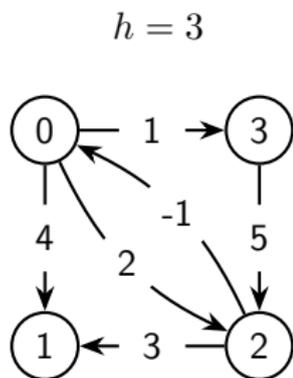


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

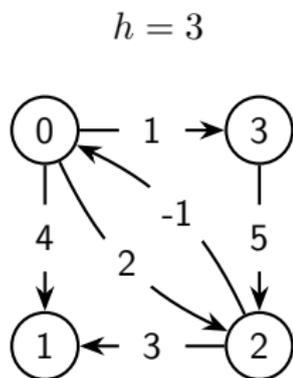


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo

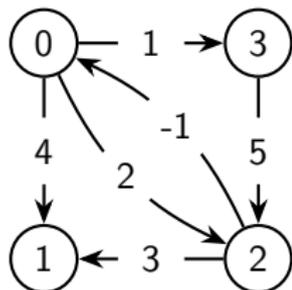


D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Algoritmo



D	0	1	2	3
0	0	4	2	1
1	∞	0	∞	∞
2	-1	3	0	0
3	4	8	5	0

N	0	1	2	3
0	0	1	2	3
1	-1	1	-1	-1
2	0	1	2	0
3	2	2	2	3

Algoritmo di Floyd-Warshall

Complessità

Per sua stessa costruzione, l'algoritmo di Floyd-Warshall ha complessità $\mathcal{O}(N^3)$.

Osservazione

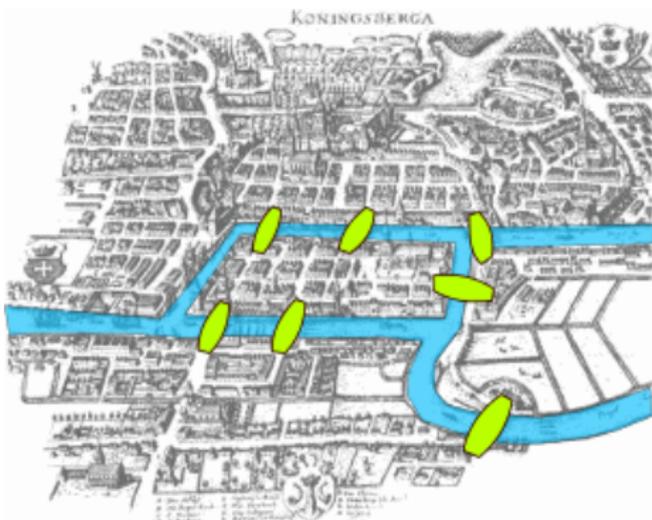
Dijkstra usando una priority queue ha complessità $\mathcal{O}(N \cdot M \log M)$, quindi Floyd-Warshall è più veloce su grafi *densi* ($M = \mathcal{O}(N^2)$).

Esercizio

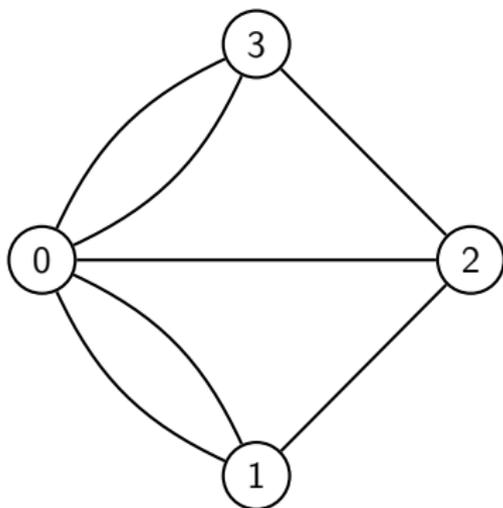
- *Algoritmo di Dijkstra in 10 minuti training – dijkstra*
- *Algoritmo di Bellman-Ford UVa 558 – Wormholes*
- *Algoritmo di Floyd-Warshall UVa 821 – Page Hopping*

Problema

I sette ponti di Königsberg



Fare una passeggiata attraversando tutti i ponti una ed una sola volta.

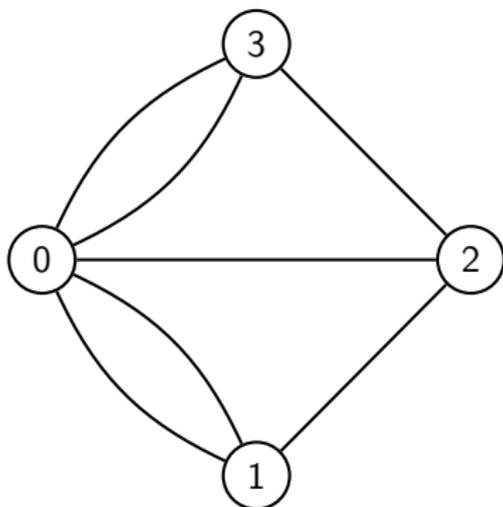


Trovare un cammino che percorre tutti gli archi una ed una sola volta.

Condizione

Esiste un cammino che percorre tutti gli archi una ed una sola volta se e solo se *tutti i vertici hanno grado pari oppure esistono esattamente due vertici di grado dispari*.

Tutti e 4 i vertici hanno grado dispari, quindi è impossibile percorrere tutti i ponti una ed una sola volta.



Algoritmo

Da quali nodi z può iniziare il cammino?

Algoritmo

Da quali nodi z può iniziare il cammino?

S : pila di nodi inizialmente vuota

aggiungi un possibile z ad S

while S non è vuota **do**

$x \leftarrow$ cima di S

 rimuovi x dalla cima di S

if grado di $x = 0$ **then**

 | aggiungi x alla soluzione

else

 | $y \leftarrow$ uno dei vicini di x

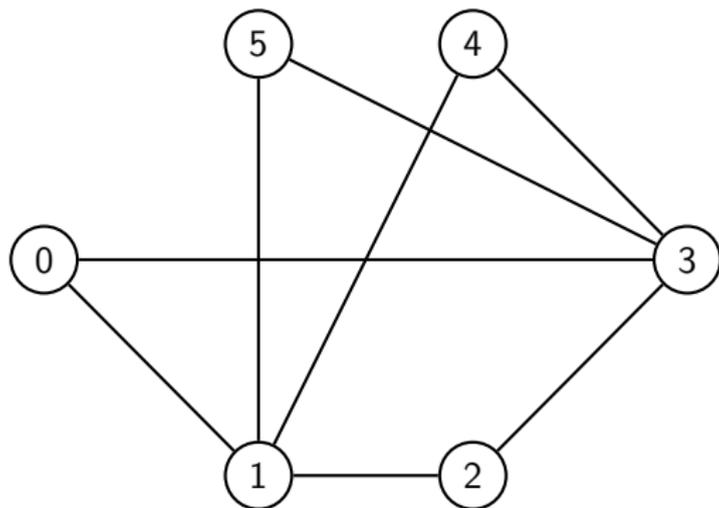
 | rimuovi quell'arco $x \rightarrow y$

 | aggiungi y ad S

end

end

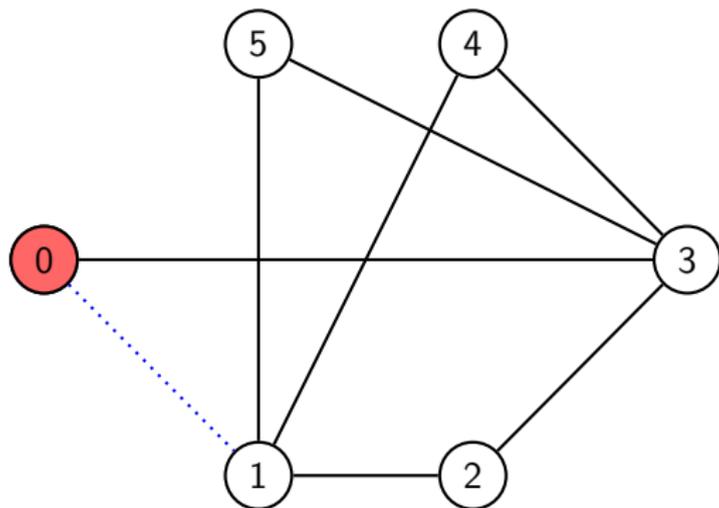
Algoritmo



stack =

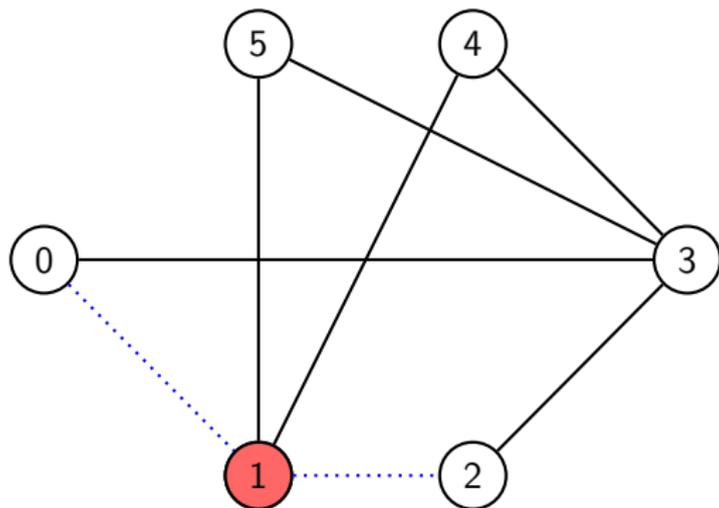
cammino =

Algoritmo



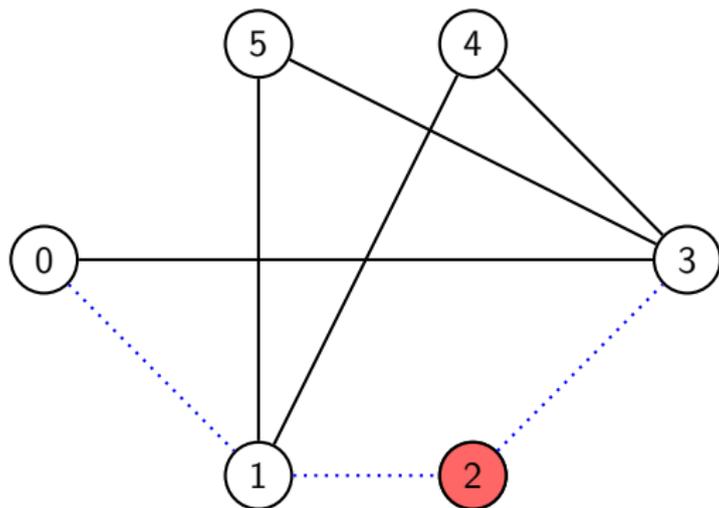
stack =
cammino =

Algoritmo



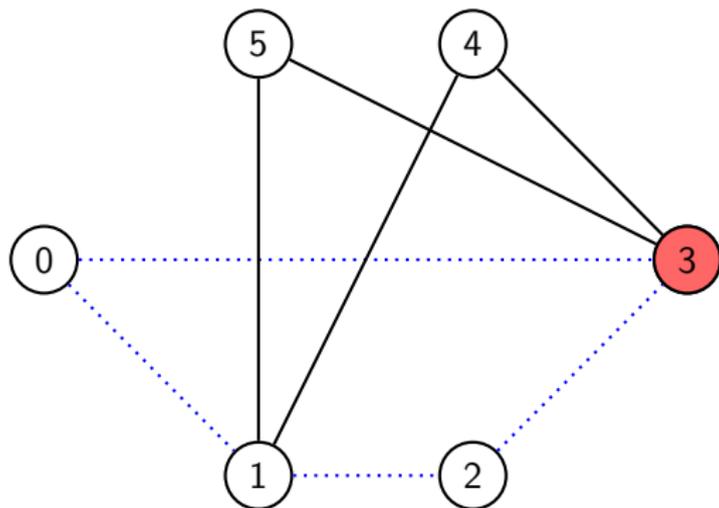
stack = 0
cammino =

Algoritmo



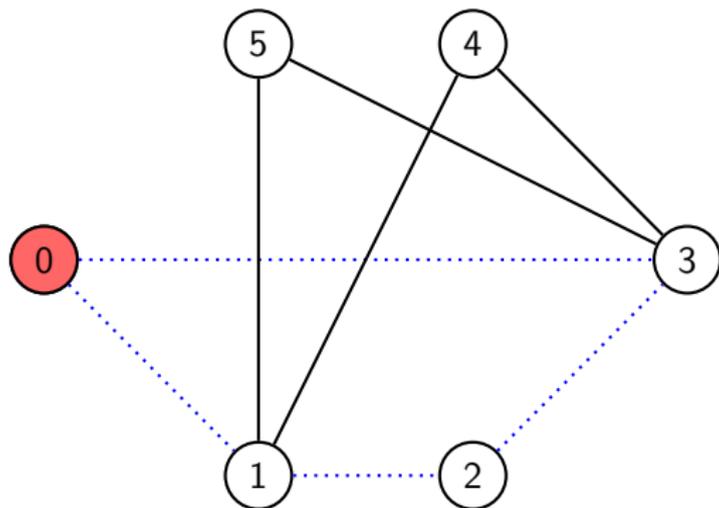
stack = 0 1
cammino =

Algoritmo



stack = 0 1 2
cammino =

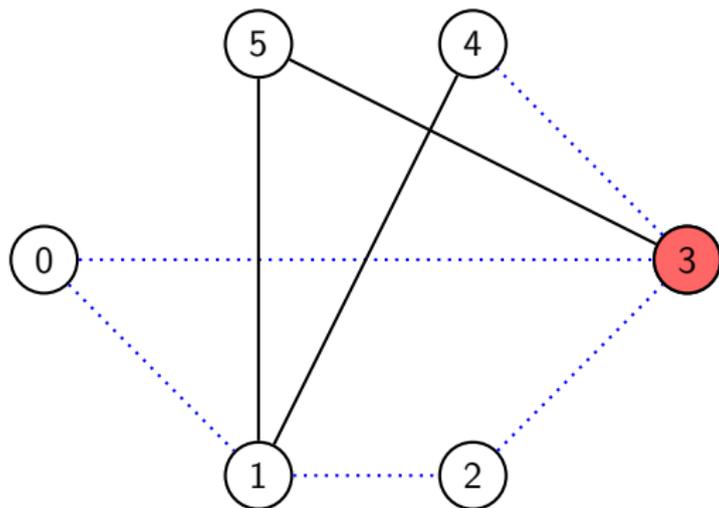
Algoritmo



stack = 0 1 2 3

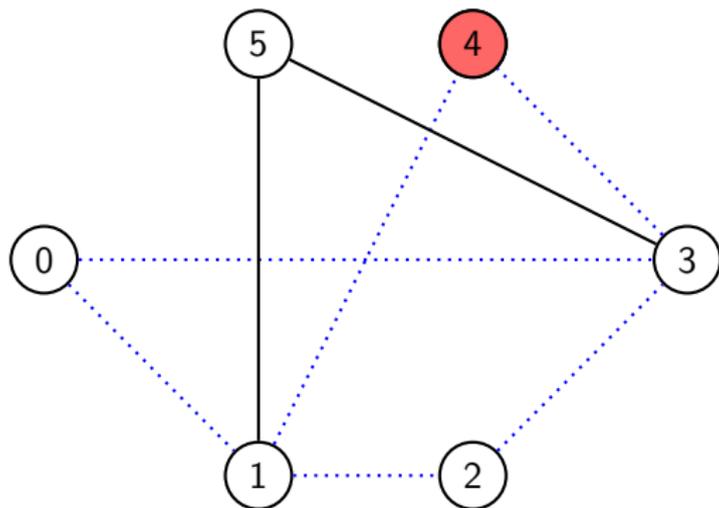
cammino =

Algoritmo



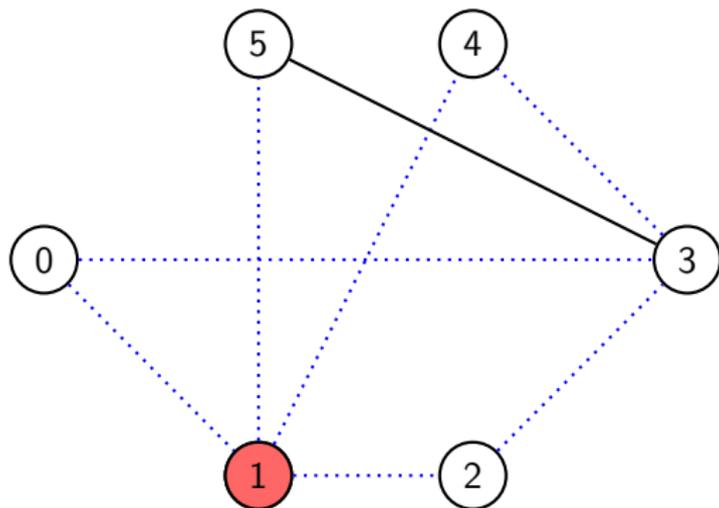
stack = 0 1 2
cammino = 0

Algoritmo



stack = 0 1 2 3
cammino = 0

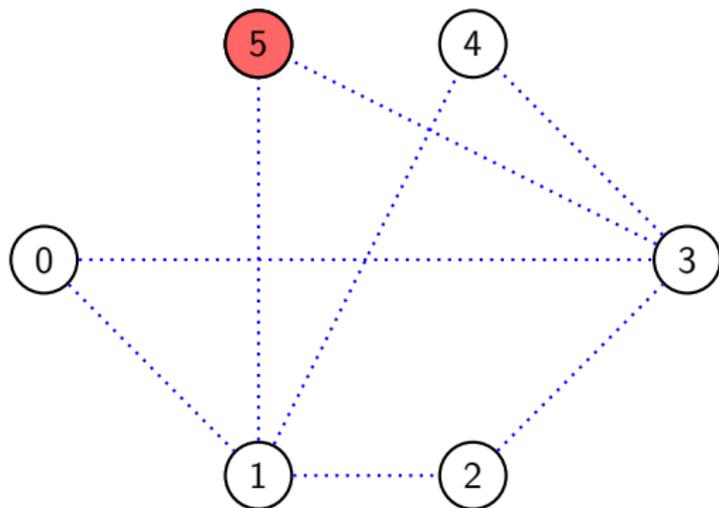
Algoritmo



stack = 0 1 2 3 4

cammino = 0

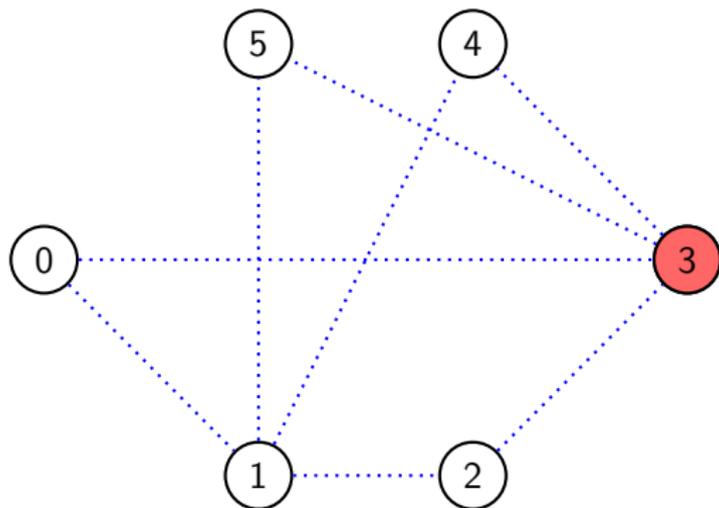
Algoritmo



stack = 0 1 2 3 4 1

cammino = 0

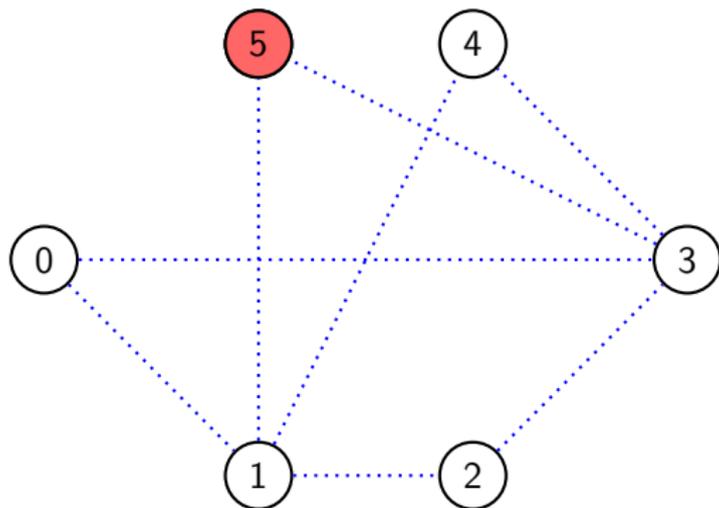
Algoritmo



stack = 0 1 2 3 4 1 5

cammino = 0

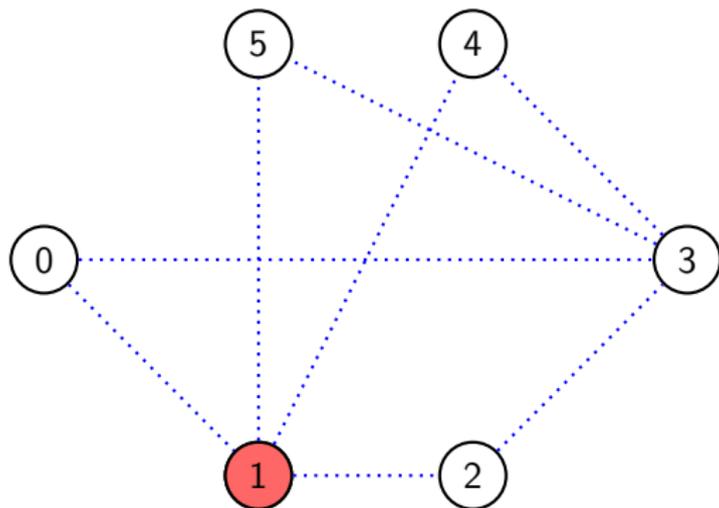
Algoritmo



stack = 0 1 2 3 4 1

cammino = 0 3

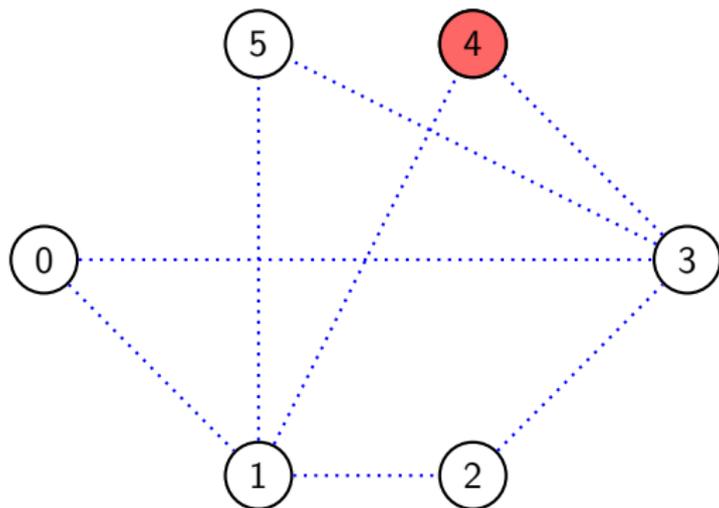
Algoritmo



stack = 0 1 2 3 4

cammino = 0 3 5

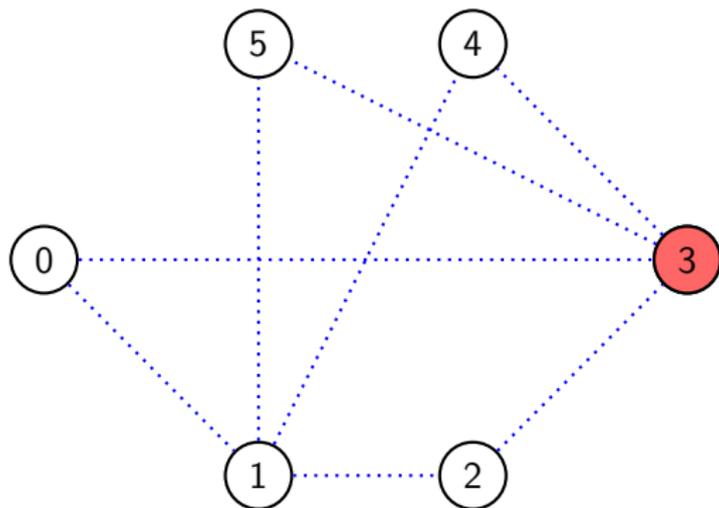
Algoritmo



stack = 0 1 2 3

cammino = 0 3 5 1

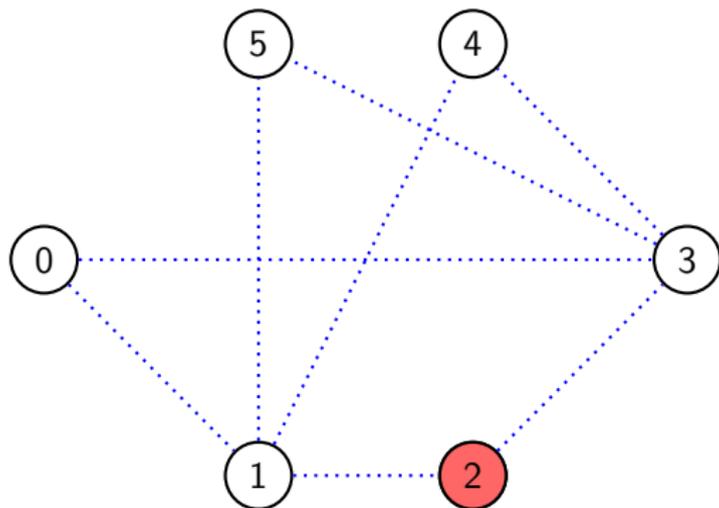
Algoritmo



stack = 0 1 2

cammino = 0 3 5 1 4

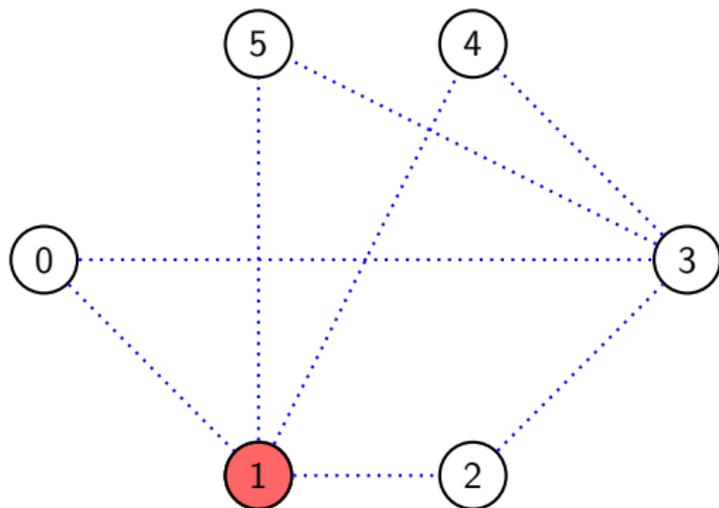
Algoritmo



stack = 0 1

cammino = 0 3 5 1 4 3

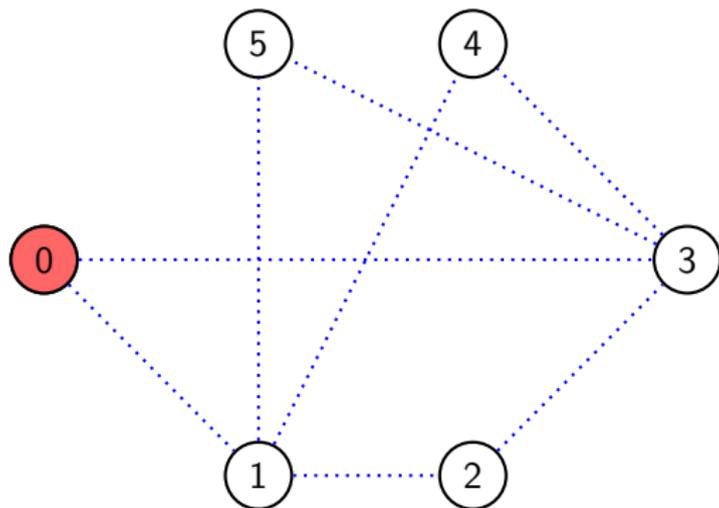
Algoritmo



stack = 0

cammino = 0 3 5 1 4 3 2

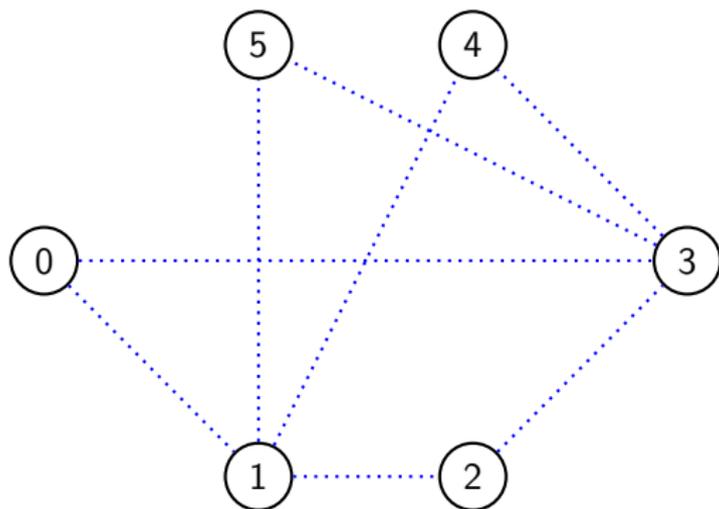
Algoritmo



stack =

cammino = 0 3 5 1 4 3 2 1

Algoritmo



stack =

cammino = 0 3 5 1 4 3 2 1 0

Esercizio

- *Cammino euleriano training – matita*