UNITELMA SAPIENZA

SAPIENZA
Università di Roma
Dipartimento di Informatica

Progettazione di algoritmi

Angelo Monti

Lezione 7:
Visita di grafi in ampiezza (BFS).

Lezione 7: Sommario

- Il problema delle distanze
- La visita in ampiezza (BFS)
- Correttezza ed efficienza della BFS

Abbiamo un puzzle meccanico formato da 9 tessere, numerate da 1 a 9, incasellate in un quadrato 3x3. La configurazione iniziale è la seguente:

1	2	3
4	5	6
7	8	9

La meccanica del puzzle permette di fare solo due tipi di mosse:

- Rotazione di una posizione a destra di una della righe
- Rotazione di una posizione in alto di una delle colonne

Ad esempio:

- Se alla configurazione iniziale applichiamo la rotazione alla seconda riga otteniamo:

1	2	3
4	5	6
7	8	9

$\Rightarrow$

1	2	3
6	4	5
7	8	9

- Se alla configurazione iniziale applichiamo la rotazione alla seconda colonna otteniamo:

1	2	3
4	5	6
7	8	9

$\Rightarrow$

1	5	3
4	8	6
7	2	9

Data una qualsiasi configurazione mescolata vogliamo trovare, se esiste, una sequenza di mosse che la riporta nella configurazione iniziale.

Ad esempio, qual'è una sequenza minima di mosse per la seguente configurazione?

7	5	9
2	3	1
4	8	6

Possiamo modellare il problema tramite un grafo diretto.

I nodi del grafo sono tutte le possibili configurazioni (anche quelle non raggiungibili tramite le mosse permesse) e c'è un arco che va nodi **u** a **v** solo se con una mossa si può passare da **u** a **v**.

Il grafo risultante avrà:

- n = 9! = 362.880
- m = 6x 362.880 = 2.177.280 (ogni nodo avrà grado uscente pari a 6)

Per risolvere il problema basterà

1) Verificare che dal nodo **u** che rappresenta la configurazione mescolata è possibile raggiungere il nodo **v** che rappresenta la configurazione iniziale.
2) Tra tutti i cammini dal nodo **u** al nodo **v** trovare quello che attraversa il numero minimo di archi.

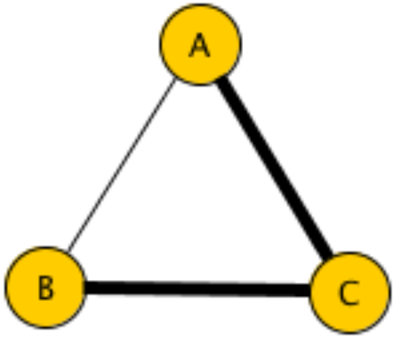
DEF. In un grafo la lunghezza di un cammino è data dal numero di archi che lo compongono.

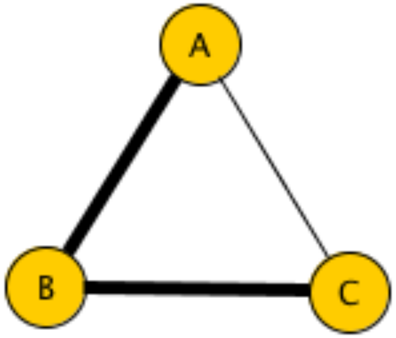
In termini generali il problema visto prima è il seguente:
Dato un grafo (diretto o non diretto) a dati due suoi nodi **u** e **v** si vuole trovare un ***cammino di lunghezza minima*** da u a v, se ne esiste almeno uno.

Una visita DFS a partire da **u** ci permette di scoprire se il nodo **v** è raggiungibile da **u** e in questo caso il vettore dei padri permette di ricavare il cammino che da **u** porta a **v** nell'albero DFS.

Per determinare i cammini di lunghezza minima la visita DFS non è adatta perché i cammini che trova non sono in genere di lunghezza minima.

Ad esempio, nel grafo qui sotto sono marcati i due possibili alberi della DFS a partire dal nodo **A**

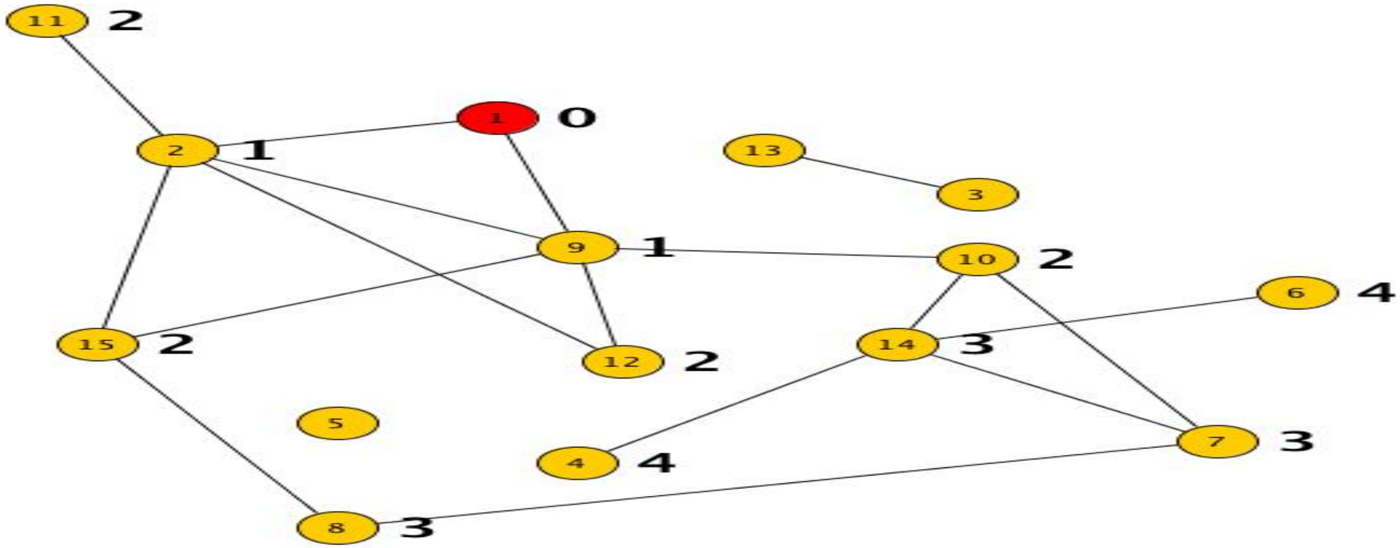




In entrambi i casi, per uno dei due nodi **B** o **C** il cammino trovato non è di lunghezza minima.

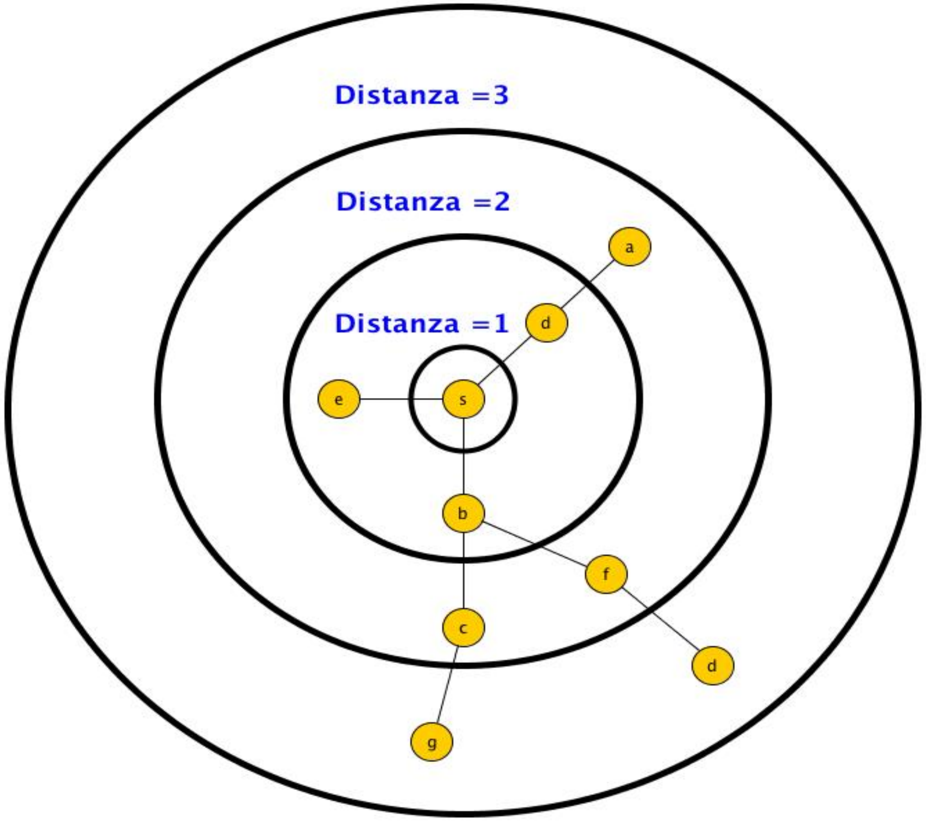
Per determinare i cammini minimi occorre un altro tipo di visita.

DEF. Dati due nodi s ed u di un grafo G, definiamo distanza (minima) in G di u da s il numero minimo di archi che bisogna attraversare per raggiungere u a partire da s. La distanza è posta a $+\infty$ se u non è raggiungibile partendo da s.



Accanto ad ogni nodo raggiungibile a partire dal nodo rosso sono riportate le distanze.

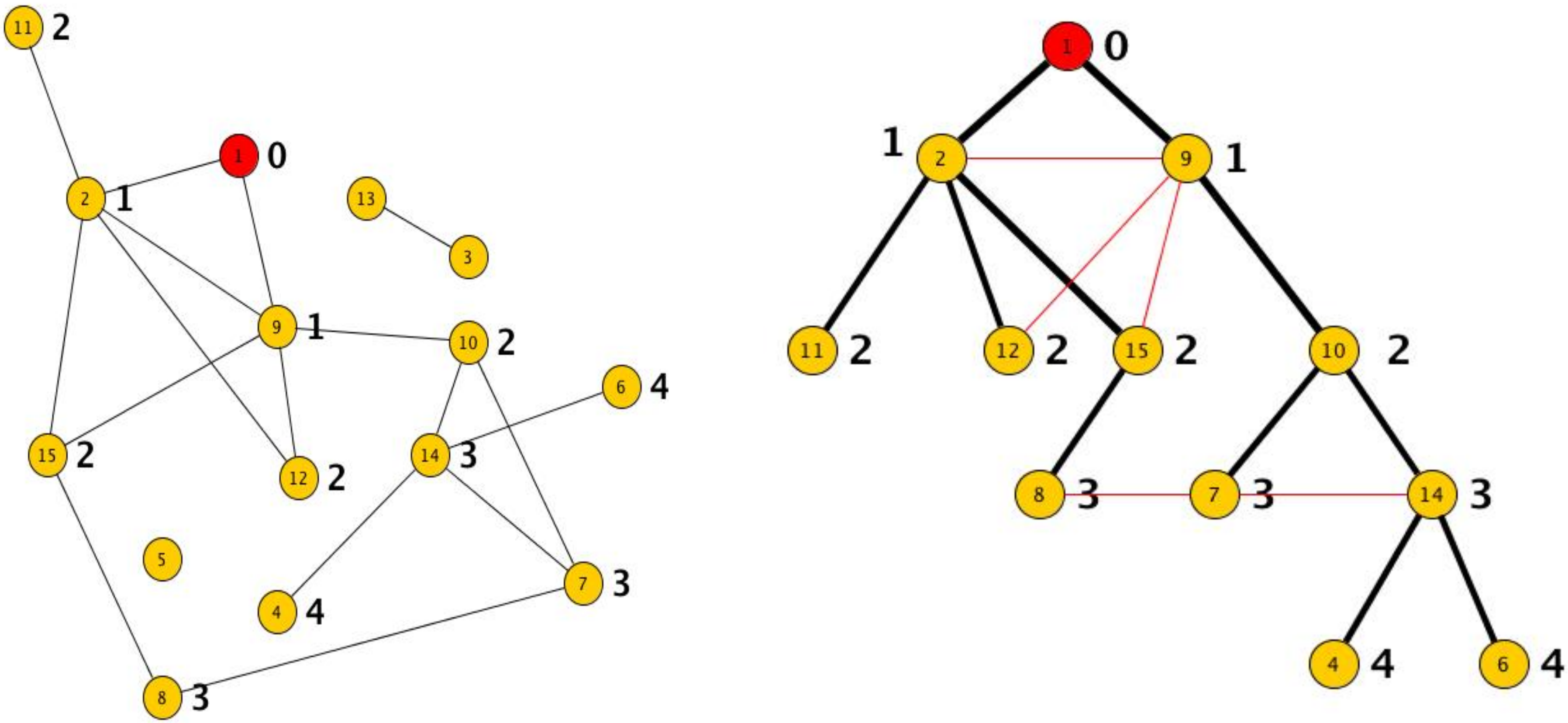
La visita in ampiezza (Breadth First Search) esplora i nodi del grafo partendo da quelli a distanza **1** da **s**. Poi visita quelli a distanza **2** e così via.
L'algoritmo visita tutti i vertici a livello ***k*** prima di passare a quelli a livello ***k+1***.



- Si genera un albero detto albero BFS
- Si visitano nodi sempre più distanti dalla radice

Per effettuare questo tipo di visita manteniamo in una coda i nodi visitati i cui adiacenti non sono stati ancora esaminati.

Ad ogni passo, preleviamo il primo nodo dalla coda, esaminiamo i suoi adiacenti e se scopriamo un nuovo nodo lo visitiamo e lo aggiungiamo alla coda.



Un grafo e l’albero BFS risultante. In rosso gli archi incontrati nel corso della visita che portano a nodi già visitati.

$P[1 \ldots n]$ **vettore dei padri inizializzato a zero**
$Dist[1 \ldots n]$ **vettore delle distanze inizializzato a** $+\infty$
$P[s] \leftarrow s$
$Dist[s] \leftarrow 0$
$Q \leftarrow$ **coda vuota**
inserisci s **in coda** Q**.**
WHILE $Q \neq \emptyset$ **DO**
    $u \leftarrow$ **estrai nodo dalla coda Q**
    FOR ogni $v \in adj(u)$ **DO**
        IF $(P[v] = 0)$ **THEN**
            $P[v] \leftarrow u$
            $Dist[v] \leftarrow Dist[u] + 1$
            inserisci v **in coda** Q
    ENDFOR
ENDWHILE

- L'algoritmo calcola le distanze dei nodi raggiungibili da **s** e produce il vettore dei padri dell'albero BFS risultante.

$P[1 \dots n]$ vettore dei padri inizializzato a zero
$Dist[1 \dots n]$ vettore delle distanze inizializzato a $+\infty$
$P[s] \leftarrow s$
$Dist[s] \leftarrow 0$
$Q \leftarrow$ coda vuota
inserisci s in coda Q.
WHILE $Q \neq \emptyset$ DO
    $u \leftarrow$ estrai nodo dalla coda Q
    FOR ogni $v \in adj(u)$ DO
        IF $(P[v] = 0)$ THEN
            $P[v] \leftarrow u$
            $Dist[v] \leftarrow Dist[u] + 1$
            inserisci v in coda Q
    ENDFOR
ENDWHILE

- Le inizializzazioni dei vettori P e $Dist$ richiedono tempo $O(n)$.
- La coda può essere implementata in modo tale che le operazioni di inserzione ed estrazione richiedano $O(1)$.
- Il WHILE esegue al più n iterazioni (perché ogni nodo viene inserito in coda al più una volta e ad ogni iterazione un nodo è estratto dalla coda).
- Ad ogni iterazione del WHILE il FOR interno fa un numero di iterazioni pari al numero degli adiacenti del nodo u estratto dalla coda in quell'iterazione del WHILE. Quindi il numero *totale* di iterazioni del FOR è la somma dei gradi (per il grafo diretto, i gradi uscenti) dei nodi e quindi $O(m)$.

La complessità è dunque O(n+m)

Correttezza dell'algoritmo per la visita BFS.

Una semplice prova per induzione su k permette di dimostrare quanto segue:

- **Per ogni $k = 0, 1, 2 \ldots$ c'è un passo dell'algoritmo della BFS in cui:**
  - **Risultano visitati tutti e soli i nodi che distano al più k da s e la loro distanza minima è stata calcolata correttamente nel vettore $Dist$**
  - **Per tutti gli altri nodi $Dist$ contiene ancora il valore $+\infty$ assegnato all'inizio.**
  - **La coda Q contiene tutti e soli i nodi che distano esattamente k da u.**

Nel passo induttivo della prova si sfrutta il fatto che i nodi che distano esattamente **k+1** da **s** sono adiacenti a nodi che distano esattamente **k** da **s** e che i valori di **Dist** una volta calcolati non vengono più modificati.

- **Per ogni $k = 0, 1, 2 \ldots$ c'è un passo dell'algoritmo della BFS in cui:**
  - **Risultano visitati tutti e soli i nodi che distano al più k da s e la loro distanza minima è stata calcolata correttamente nel vettore $Dist$**
  - **Per tutti gli altri nodi $Dist$ contiene ancora il valore $+\infty$ assegnato all'inizio.**
  - **La coda Q contiene tutti e soli i nodi che distano esattamente k da u.**

Sia d la distanza massima cui possono trovarsi nodi di G raggiungibili da s. L'affermazione in rosso ci dice che, nel corso dell'esecuzione dell'algoritmo, c'è un passo in cui:

- tutti i nodi a distanza al più d (vale a dire tutti i nodi raggiungibili da s) avranno la loro distanza calcolata correttamente in $Dist$
- tutti i nodi non raggiungibili da s avranno in $dist$ la loro distanza posta correttamente a $+\infty$.
- nella coda saranno presenti solo i nodi a distanza esattamente d.

Nei passi seguenti dell'algoritmo uno dopo l'altro tutti i nodi in coda verranno estratti, nessuno di questi porterà ad un nodo non ancora visitato da inserire in coda, di conseguenza la coda finirà per svuotarsi e l'algoritmo terminerà. A quel punto tutti i nodi avranno in $Dist$ il valore corretto o perché visitati o perché non raggiungibili a partire da s.