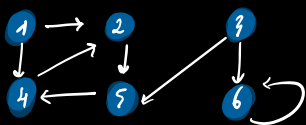


1) SCRIVERE LA MATRICE DI ADIACENZA DEL SEGUENTE GRAFO:

19/11

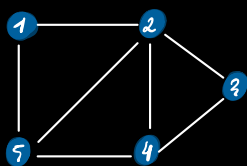


Variable utilizzata: $\forall (i,j) \in V \times V$

$$a_{ij} = \begin{cases} 1 & \text{se } (i,j) \in E \\ 0 & \text{se } (i,j) \notin E \end{cases}$$

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

2) SCRIVERE LA LISTA DI ADIACENZA DEL GRAFO NON ORIENTATO:



- 1 • → 2 → 5
- 2 • → 1 → 5 → 4 → 3
- 3 • → 2 → 4
- 4 • → 3 → 2 → 5
- 5 • → 4 → 2 → 1

* SE FACCIAMO LA MATRICE DI ADIACENZA ESCE COSÌ?

SIMMETRICA RISPETTO ALLA DIAGONALE.

ESPLORAZIONE IN AMPIEZZA

BFS (Breadth-First search): esplora un grafo a partire da un v. sorgente. Esplora prima i "vicini" e poi i "vicini dei figli".

* COSA FA L'ALGORITMO:

1) SCOPRE I VERTICI CHE SONO RAGGIUNGIBILI DA S.

RAGGIUNGIBILI = v è raggiungibile da u se \exists cammino da u a v.

CAMMINO = è una sequenza finita di vertici u_0, u_1, \dots, u_k dove $u_0 = u$, $u_k = v$ e $\forall i \in \{0, \dots, k-1\}$ $(u_i, u_{i+1}) \in E$.
 insieme archi

2) Calcola $\forall v \in V$, la distanza di quel vertice DALLA SORGENTE. (numero minimo di archi).

DISTANZA: # di archi su un cammino minimo da u a v.

- PRINCIPIO DI FUNZIONAMENTO:

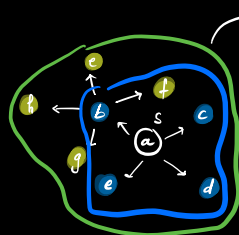
3) Amplia la frontiera tra vertici scoperti e non scoperti in modo tale che prima scopre i vertici a distanza 0 da S, poi:

scopre i vertici a distanza 1 da S.

" " " 2 da S.

" " " 3 da S.

.....



FRONTIERA CHE SI ALLARGA AD OGNI NUOVA SCOPERTA.

* ESPLORO PRIMA I VICINI, POI UNO ALLA VOLTA.

"VICINI DEI VICINI".
 ↳ uno alla volta.

4) Genera un albero con una radice S contenente tutti i vertici raggiungibili da S, tramite i valori π al termine dell'esecuzione.

→ LOGICA DI ESPANSIONE DEI GRAFI IN AMPIEZZA:

ISTANZA: $G(V, E)$, $s \in V$

- * grafo orientato o non
- * un suo vertice

* nodi scoperti direttamente da $s \rightarrow d=1, \pi=s$

* nodi figli dei figli diretti di $s \rightarrow d=2, \pi=(s).next(c)$

* figli dei figli di $s \rightarrow d=n, \pi=(s).next(c).next(c) \dots (u)$

↳ con liste / matrici di adiacenza diventa molto semplice.

* ALGORITMO:

BFS (G, s): $G(V, E)$ $s \in V$

for $u \in V \setminus \{s\}$

$u.col = "white"$
 $u.d = \infty$
 $u.\pi = NIL$

initialization!

Enqueue (G, s):

$s.col = "gray"$
 $s.d = 0$
 $s.\pi = \emptyset$

initialization del nodo sorgente.

→ INFORMAZIONI AGGIUNTIVE:

c → colore, dello "stato di esplorazione" del nodo.

d → distanza dalla sorgente (come definita prima)

π → "predecessore", da "chi era stato scoperto."

* è il "parent" del vertice corrente.

Occhio! I non collegati al nodo s (componenti indipendenti)

NON VENGONO ESPORATI.

while $Q \neq \emptyset$ \triangleright coda dei vertici del grafo a distanza d

$u = \text{Dequeue}(Q)$ // Estraggo 1 vertice dalla frontiera corrente.

for $v \in \text{Adj}[u]$ // Espando, per quel vertice, di 1 la distanza di frontiera.

IF ($v.col == "white"$):

$v.col = "gray"$

$v.\pi = u$

$v.d = u.d + 1$

Aggiorno il vertice appena scoperto.

Enqueue (G, v) → lo metto in coda! Cioè, "è grigio" e ne devo ancora esplorare gli adiacenti.

$u.col = "black"$

↳ Ho esplorato TOTALMENTE l'ampiezza di un nodo.

ALBERO GENERATO DA s :

$G(V, E)$ se V ALBERO → grafo non orientato connesso e aciclico

ALBERO BFS:

$G_\pi = (V_\pi, E_\pi)$ $V_\pi = \{u \in V \mid u.d \neq \infty\} \cup \{s\}$ → POSSO SCRIVERLA IN ALTRI 2 MODI → * con colore. $\{n \in V \mid n.col = "black"\}$
* con parent. $\{...\}$

$E_\pi = \{(u.\pi, u) \mid u \in V_\pi \setminus \{s\}\}$

ALGORITMO FUNZIONA BENE ANCHE SENZA i colori! (uso $0 \dots \infty$, per le distanze).

For $u \in V \setminus \{s\}$

$u.d = \infty$

$s.d = 0$

$Q := \emptyset$

ENQUEUE(Q, s)

While $Q \neq \emptyset$

$u = \text{DEQUEUE}(Q)$

For $u \in \text{Adj}[u]$

If $(u.d = \infty)$

ENQUEUE(Q, u)

$u.d = u.d + 1$

Con conosco i vertici raggiungibili con cammino minimo.
Conosco la distanza.

Riassumendo:

* Inizializzazione \rightarrow Ogni nodo in più, aggiunge $\begin{cases} \text{colore} \\ \text{parent} \\ \text{distanza dalla sorgente} \end{cases}$
 \rightarrow Viene utilizzata una queue come struttura dati di supporto.
Contiene \rightarrow "vertici da visitare" (in ampiezza)