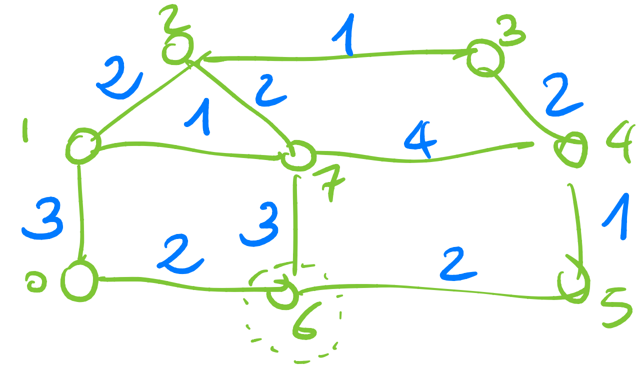
GRAFI PESATI

$G=(V,E,W)$ $\quad W: E \to \mathbb{R}$

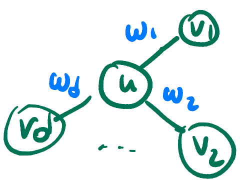


matrice adiacenza $A_{ij} = \begin{cases} 1 & (i,j)\in E \\ 0 & \text{altrimenti} \end{cases}$

$$A_{ij} = \begin{cases} W(i,j) & \text{se } (i,j)\in E \\ \text{None} & \text{altrimenti} \end{cases}$$

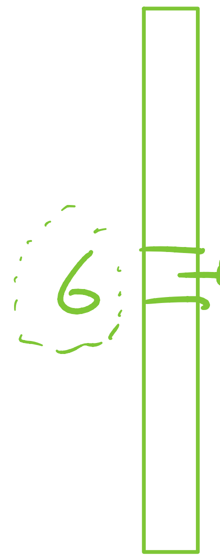
liste di adiacenza

adj

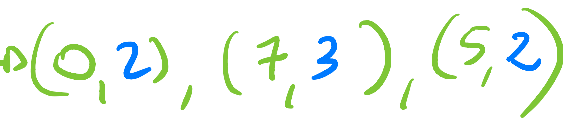


u → $(v_1, w_1), (v_2, w_2) \cdots (v_d, w_d)$

pair<nodo, peso>



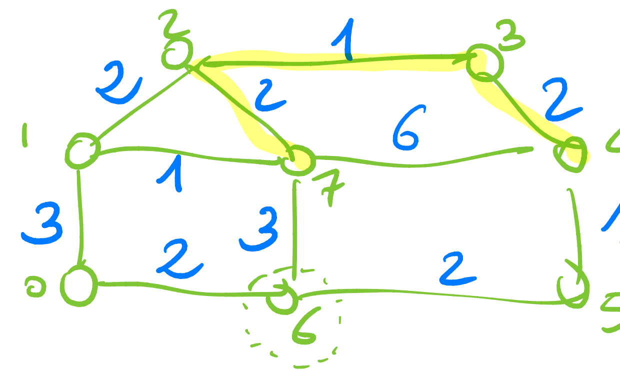
6 → (0,2), (7,3), (5,2)



Cammino pesato

sequenza di nodi $u_1 \; u_2 \dots u_k$

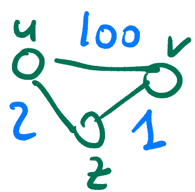
peso = $\sum_{i=1}^{k-1} w(u_i, u_{i+1})$



$d(u,v)$ = il peso minimo di un cammino tra u e v

$d(7,4) = 5$

cammino di peso minimo $\neq$ cammino di lunghezza minima



es. tariffe dei voli

quindi non possiamo usare la BFS per trovare i cammini minimi

Greedy: Algoritmo di Dijkstra per i cammini minimi

Hp: $W : V \to \mathbb{R}^+$

Proprietà invariante mantenute dall'algoritmo:

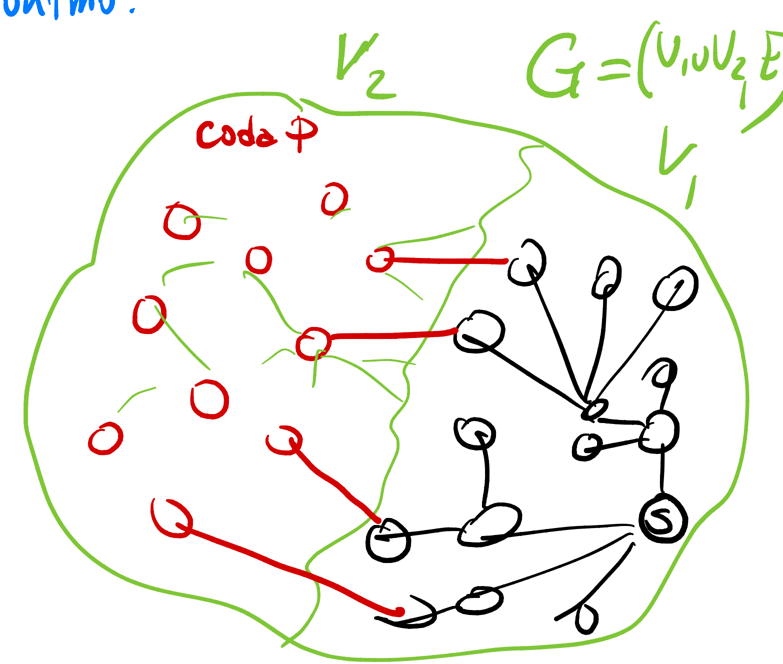
nodo s di partenza:

$d(s,u)$ $\forall u \in V = V_1 \cup V_2$

coda P: estrae l'elemento minimo ogni volta

ogni nodo $v \in V_2$ mantiene (se esiste) un arco verso l'albero in V_1 t.c. minimizza la sua distanza "potenziale" con S

"BFS" che tiene conto del peso degli archi



albero dei cammini minimi

Inizialmente: $V_1 = \{s\}$, $V_2 = V - \{s\}$

$dist[u]$ = peso del miglior cammino trovato finora usando soltanto nodi in V_1 e archi da V_2 a V_1

$dist[s] = 0$, $dist[u] = +\infty$ per $u \neq s$ e vicini di s

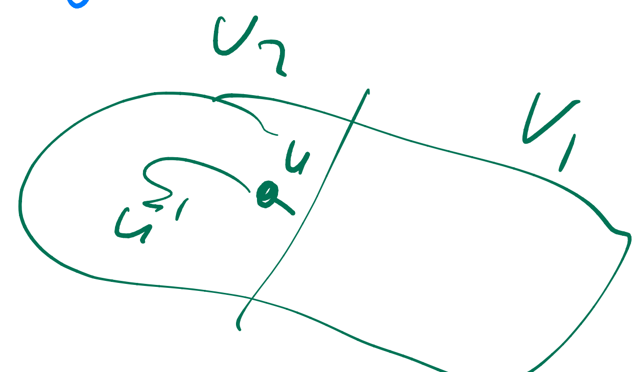
$dist[v] = w(s,v)$ se $(s,v) \in E$

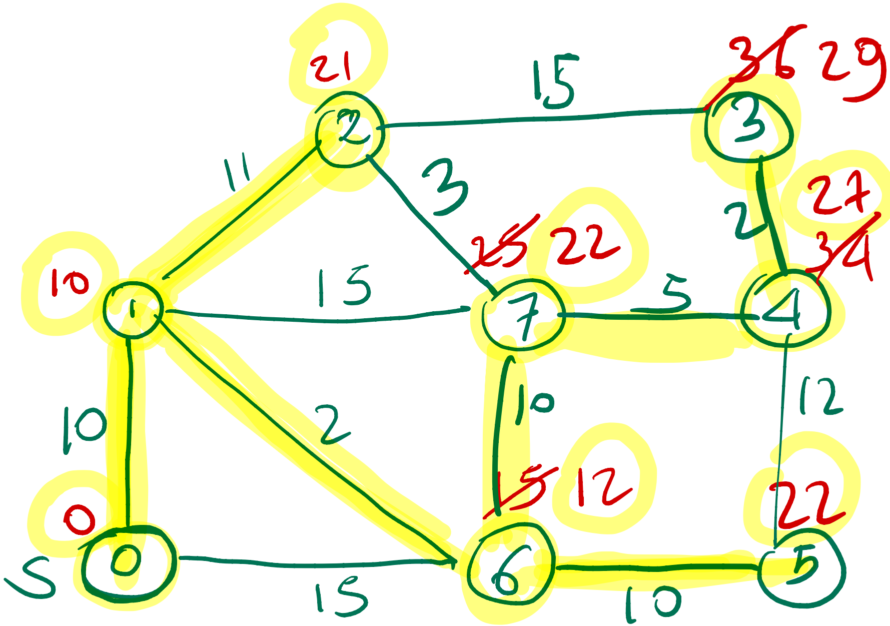
Passo generico: prende $u \in V_2$ t.c. $dist[u] \leq dist[u']$ per $u' \in V_2$, $u' \neq u$

CLAIM $dist[u] = d(s,u)$ per tale nodo

Idea: non è migliorabile utilizzando gli altri nodi perché i pesi sono positivi

Il cammino minimo $u, u_1, \dots, u_k, s$ è tale che $u_1, u_2, \dots, u_k \in V_1$





$d(s,u) \; \forall u$

IMPORTANTE: PESI > 0

Fatto: Minima distanza candidata, tra quelle disponibili, diventa la distanza finale, perché le altre non possono diminuirla ulteriormente

regola di rilassamento: $\forall v \in N(u)$

$$d(s,u) + w(u,v) < d'(s,v)$$

$$\Rightarrow \underbrace{d'(s,v)}_{\text{distanza candidata}} = \underbrace{d(s,u)}_{\text{distanza finale}} + w(u,v)$$

(0, 1, 10) (6, 7, 10) (7, 4, 5)

(1, 2, 11) (6, 5, 10) (4, 3, 2)

(1, 6, 2)

$\boxed{\infty} = \sum \text{pesi} + 1$

P = coda di priorità

$S = 0 \qquad d(S,S) = 0$

$$\text{Costo} = O\Big(|E| \cdot \underbrace{\text{costo}(\text{pop/push})}_{\lg|V| \text{ usando alberi bilanciati}}\Big) = O\big(|E| \cdot \lg|V|\big)$$

(si può migliorare con gli heap di Fibonacci)
$O(|V| \lg|V| + |E|)$

$d(s,u)$: $s \in V$, $\forall u \in V$

$\forall s,u \in V$: $d(s,v)$: eseguire Dijkstra $\forall s \in V$

for $s = 0, 1, \ldots, n-1$:
  esegui Dijkstra(s)

la distanza tra ogni coppia di nodi

$$\text{diametro}(G) = \max_{u,v \in V} d(u,v) = \max_{s \in V} \Big(\underbrace{\max_{u \in V} d(s,u)}_{\text{Dijkstra}(s)}\Big) \qquad O(|V| \cdot |E| \cdot \lg|V|)$$