*Punto a)* Si sceglie l'agoritmo di Bellman-Ford in quanto esistono archi con costo negativo. Bellman-Ford è l'unico algoritmo visto in grado di garantire convergenza alla soluzione ottima del problema dei cammini minimi in presenza di archi di costo negativo, sebbene mediamente meno efficiente dell'algoritmo di Dijkstra, che non garantisce di trovare la soluzione al problema dei cammini minimi se esistono archi di costo negativo.

*Punto b)*

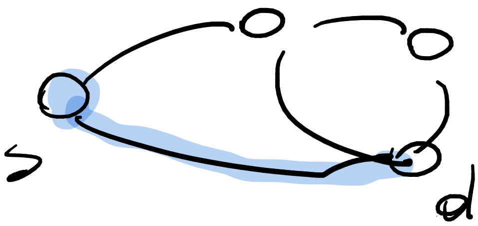
| *Iter.* | A | B | C | D | E | F | G | *Aggiornati* |
|---|---|---|---|---|---|---|---|---|
| *h=0* | 0(-) | +∞(-) | +∞(-) | +∞(-) | +∞(-) | +∞(-) | +∞(-) | A |
| *h=1* | 0(-) | 5(A) | –1(A) | 3(A) | +∞(-) | +∞ (-) | +∞(-) | B, C, D |
| *h=2* | 0(-) | 4(D) | –1(A) | 0(C) | 8(B) | 3(C) | +∞(-) | B, D, E, F |
| *h=3* | 0(-) | 1(D) | –1(A) | 0(C) | ~~7(B)~~ 5(F) | 2(D) | 4(F) | B, E, F, G |
| *h=4* | 0(-) | 1(D) | –1(A) | 0(C) | ~~4(B)~~ 3(G) | 2(D) | 3(F) | E, G |
| *h=5* | 0(-) | 1(D) | –1(A) | –1(E) | 2(G) | 2(D) | 3(F) | D, E |
| *h=6* | 0(-) | 0(D) | –1(A) | –2(E) | 2(G) | 1(D) | 3(F) | B, D, F |
| *h=7* | 0(-) | –1(D) | –1(A) | –2(E) | 2(G) | 0(D) | 2(F) | B, F, G |

GRAFI

BELLMAN-FORD

DIJKSTRA

BELLMAN-FORD =>




O(n^3)

G = (V, E)

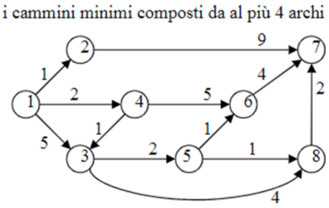
S = SORGENTE

D = DESTINAZIONE

V = NODI

E = EDGES = ARCHI

3. Si vogliono determinare i cammini minimi composti da al più 4 archi sul seguente grafo:



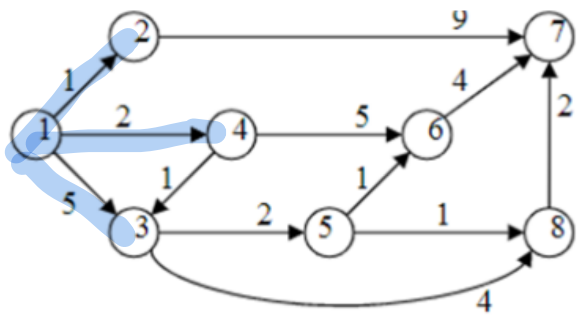

-> DIJKSTRA!

[ - si scelga un algoritmo appropriato e si motivi la scelta; ]
- si calcolino i cammini minimi con al più quattro archi dal nodo 1 verso tutti gli altri nodi (i passi dell'algoritmo vanno riportati in una tabella e giustificati);
- si ricavi un cammino minimo di al più quattro archi da 1 a 7, descrivendo il procedimento adottato.

O con MAX. ITER.

BF -> PESI NEGATIVI (SE ∃ ! PESO NEGATIVO)

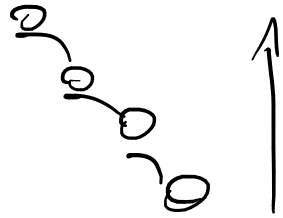
[D -> PESI POSITIVI (TUTTI) = PIÙ SEMPLICEMENTE ]



| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $0$ | $1_1$ | $5_1$ | $2_1$ | $\infty$ | $\infty$ | $\infty$ |

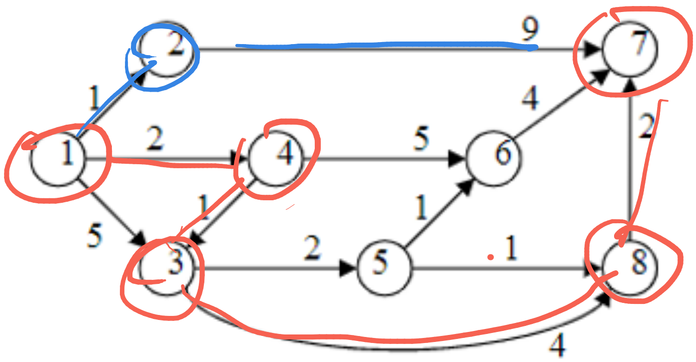
$h=1$

$h=2$



PREDEC.

| Iterazione | Nodo 1 | Nodo 2 | Nodo 3 | Nodo 4 | Nodo 5 | Nodo 6 | Nodo 7 | Nodo 8 | Aggiornamenti |
|---|---|---|---|---|---|---|---|---|---|
| *Inizio* | $0_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | 1 |
| $h=1$ | $0_\Lambda$ | $1_1$ | $5_1$ | $2_1$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | $+\infty_\Lambda$ | 2,3,4 |
| $h=2$ | $0_\Lambda$ | $1_1$ | $3_4$ | $2_1$ | $7_3$ | $7_4$ | $10_2$ | $9_3$ | 3,5,6,7,8 |
| $h=3$ | $0_\Lambda$ | $1_1$ | $3_4$ | $2_1$ | $5_3$ | $7_4$ | $10_2$ | $7_3$ | 5,8 |
| $h=4$ | $0_\Lambda$ | $1_1$ | $3_4$ | $2_1$ | $5_3$ | $6_5$ | $9_8$ | $6_5$ | 6,7,8 |



S - D

PREDEC. ALL' ULTIMO NODO!

CAMMINO MINIMO

DEFINIZIONE

ALBERO! ⇒ TUTTI I PERCORSI!

La tabella riporta al riga 0 di inizializzazione e una riga per ogni iterazione. All'iterazione $h$ si controllano gli archi $(i,j)$ uscenti da ciascun nodo $i$ nella colonna *Aggiornati* alla riga $h$–1, e si aggiornano i costi e i predecessori del nodo $j$ all'iterazione $h$ qualora l'etichetta del nodo $i$ all'iterazione $h$–1 più il costo dell'arco $(i,j)$ sia strettamente minore dell'etichetta corrente del nodo $j$.

L'algoritmo si ferma qualora la lista dei nodi aggiornati sia vuota (convergenza delle etichette ai costi dei cammini minimi da A verso gli altri nodi) o, come in questo caso, venga completata l'iterazione con $h$ uguale al numero di nodi avendo dei nodi aggiornati (presenza di un ciclo negativo).