* Cammini minimi da sorgente unica.

$G = (V, E)$ $w: E \to \mathbb{R}$ $s \in V$ è il vertice sorgente. → da cui voglio partire e di cui voglio conoscere ogni collegamento minimo con gli atri.

RISOLVONO LE RICHIESTE IN TEMPO E SPAZIO DI FLOYD-WARSHAU.

↳ Se voglio informazioni su un solo nodo → non mi serve tutta la computazione della matrice. (MIGLIORO IN TERMINI DI EFFICENZA).

= "Voglio sapere il modo più veloce per andare da Milano a Roma?

2 Algoritmi principali

- Bellman Ford → (più generale)
- Dijkstra → (solo con pesi $\mathbb{R}^+$)

- Basati sul concetto di sottostruttura ottima di cammini minimi:

$u \longrightarrow v \longrightarrow z$

(u,v) minimo + (v,z) minimo

* INFORMAZIONI AGGIUNTIVE PER GLI ALGORITMI:

$\Pi(v)$ → predecessore di un generico v_i in un cammino minimo.

$d(v)$ → distanza tra il vertice v_i al vertice sorgente.

* Tecnica di rilassamento (concetto chiave dei 2 algoritmi):

→ Determina $d(v)$

• Algoritmo di rilassamento:

// inizializzazione algoritmo

Init_source (G, s) $G = (V, E)$

For $v \in V$:

$d(v) = \infty$;
$\Pi(v) = nil$;

* Logica della tecnica di rilassamento:

È migliore la conoscenza accumulata fino ad ora, o il nuovo peso del lato?

$d(v)$ ~ $d(u) + d(v)$?

$u \xrightarrow{w} v$

50 100

$\pi(s) = nil$;
$d(v) = 0$;

Sapendo il valore di w, posso scegliere se "rilassare" v oppure no.

// Rilassamento

Relax (u, v, w) : $\in O(1)$

if $d(v) > d(u) + w(u,v)$:

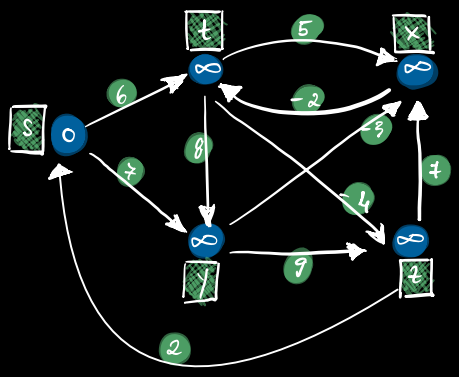
$d(v) = d(u) + w(u,v)$
$\pi(v) = u$;

$w = 20?$ → arrivo a v da u, con peso 70

$w = 60?$ → tengo il vecchio cammino di v

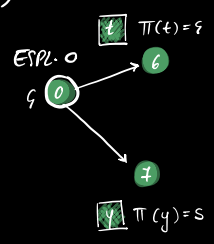
Bellman-Ford: RESTITUISCE
- **T** → POSSO CREARE OGNI $d(s)$
- **F** → NON POSSO FARLO

Fa un rilassamento su tutti i lati del grafo, espandendo la conoscienza sulle distanze minime tra i v.

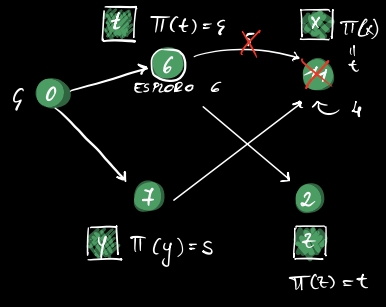


Procedo seguendo l'algoritmo di rilass:

1)

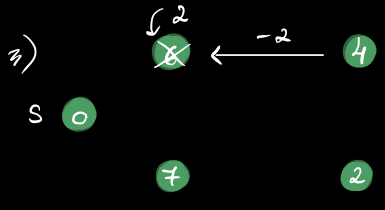


2)

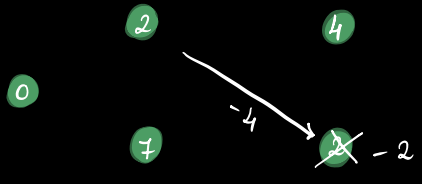


Ad ogni passo → rianalizzo ogni vertice.

3)



4)



5) Dopo un po' → non miglioro più, HO ANALIZZATO TUTTI I CAMMINI.

* Una volta fatti (n° vertici -1) passi → ho tutti i costi di esplorat. ↦ rilass.

del grafo, partendo da un sorgente.

* **ALGORITMO BELLMAN-FORD:**

```
BellmanFord (G, w, s):                          Θ(V)
    init_source (G, s);

For i=1 to |V| - 1 :                            Θ(|V| · |E|)
    For all (u,v) ∈ E :
        Relax (u, v, w);                        Θ(1)

For all (u,v) ∈ E :   // Controllo che l'algoritmo non rimanga bloccato in rilassamenti (di cicli)
    IF d(v) > d(u) + W(u,v) :                   O(|E|)
        return FALSE;
Return TRUE;
```

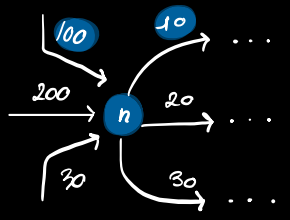
* Risulta: $T(n,m) = \Theta(n \cdot m)$

Algoritmo Dijkstra:

- **REQUISITO**: non ho pesi negativi.

Cosa significa? Che dato un qualsiasi nodo:

* **OTTIMO CORRENTE +**:



* La "migliore uscita", sarà l'arco di peso minore.

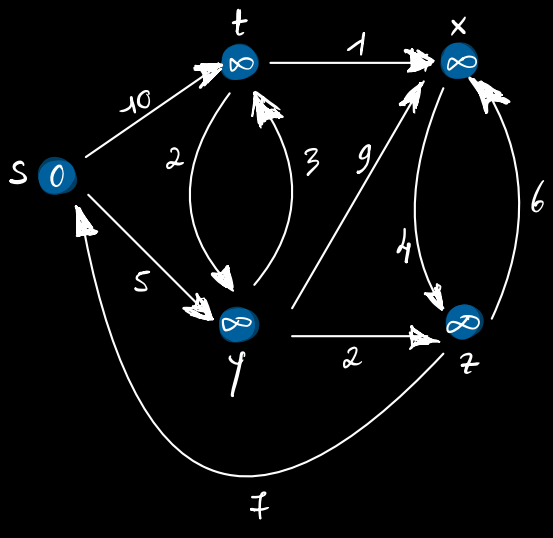
* Ed il discorso vale anche per "la migliore entrata".

Cosa che, con Bellman Ford non possiamo garantire gli archi negativi "rovinano" l'OTTIMO "GLOBALE" PER 1 NODO.

* **loop**:

1. Parto da un $n \in V$
2. Guardo cosa c'è vicino → setto le distanze dei vertici vicini
3. Prendo quello con distanza minima, perché so CHE NON È ULTERIORMENTE MIGLIORABILE (ho solo archi positivi).

esempio:



IT # 0)
$d(s) = 0$
$d(V) = \infty$

IT # 1)
$d(s) = 0$
$d(t) = 10$
$d(y) = 5$
↓
sicuro!

IT # 2)
$d(s) = 0$
$d(y) = 5$
$d(t) = \cancel{10} = 8$
$d(x) = 14$
$d(z) = 7$ → sicuro

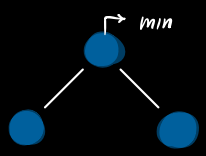
IT # 3)
$d(s) = 0$
$d(y) = 5$
$d(t) = 8$
$d(x) = \cancel{14} = 8$
$d(z) = 7$

IT # 4)
$d(s) = 0$
$d(y) = 5$
$d(t) = 8$
$d(x) = 9$
$d(z) = 7$

* Non riguardo ogni volta tutti i lati (Bellman-Ford).

→ Selezione nodo minimo

HEAP → array trattato come albero binario, vale ∀u:



* ogni u ha → left(u) > u, right(u) > u
* Facilito l'estrazione del minimo.
* Ho una heapsize[H]

Algoritmo Dijkstra:

Dijkstra (G, w, s) $\Theta(|V|)$

init_source (G, s);

S = ∅ ; // insieme soluzioni

H = V ; // nello heap ci mettiamo tutti i vertici.

While H ≠ ∅ : // Finché ci sono d(v) = ∞ (non proprio vuoto)

- u = extract-min (H)
- S = S ∪ {u}
- For all v ∈ Adj {u} // aggiorno le distanze degli adiacenti (RILASSAMENTO)
  - Relax (u, v, w) ; $O(\log(u))$

Aggiorna ogni lista di adiacenza di ogni vertice, facendo un rilassamento.

↳ $O[(|V| + |E|)\log(u)]$

↓

più o meno = T(u) alg.

CONFRONTO ALGORITHI:

	T(u)	S(u)	Requisiti
Floyd_Warshall	$\Theta(|V|^3)$	$\Theta\|V^2\|$	$w[(i,j) \in E] \in \mathbb{R}$

Bellman-Ford	$O(\|V\|^2)$	$\Theta(\|V\|)$	$w[(i,j) \in E] \in \mathbb{R}$
Dijkstra	$O(\|V\| \cdot \log\|V\|)$	$\Theta(\|V\|)$	$w[(i,j) \in E] \in \mathbb{R}^+$