APSP → all pair shortest path

Gli algoritmi studiati per il SSSP possono essere usati anche per APSP, ma hanno una complessità più alta

BF → $O(V^4)$

DIJ → $O(V^3 \log V)$

DAG → $O(V^3)$

Di seguito un problema di prog dinamica per ottenere meglio di Bellman-Ford

Risolveremo in modo ricorsivo

La dimensione del problema è identificata attraverso:

\# di archi che possono essere presenti in un cammino minimo

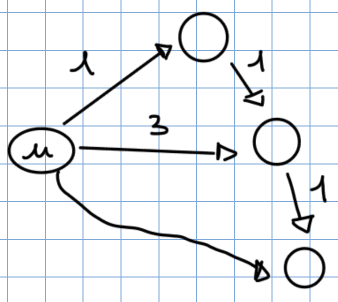
=

lunghezza massima di un cammino minimo

=

d → che sarebbe $V-1$

numero di archi possibili



$\delta^0(u,v) = +\infty$

$\delta^1(u,v) = 7$

$\delta^2(u,v) = 4$

$\delta^3(u,v) = 3$

$\delta^4(u,v) = 3$

$d = V-1$

$d = V-2$ } sottoproblemi più piccoli

$\vdots$

$d = 0$

W = matrice di adiacenza

Caso base: $d = 0$

$$\delta^0(u,v) = \begin{cases} 0 & \text{se } u = v \\ +\infty & \text{se } u \neq v \end{cases}$$

} non esiste nessun arco

Caso base

$d = 1$

$$\delta^1(u,v) = W[u,v]$$

Caso base

$$W[i;j] = \begin{cases} 0 & \text{se } i = j \\ peso(i,j) & \text{se } (i,j) \in E \\ +\infty & \text{se } (i,j) \notin,\ i \neq j \end{cases}$$

quale scegliamo?

$d = V-1$

$d = V-2$

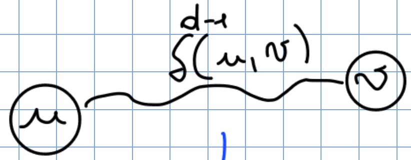
$\vdots$

$d = 1$

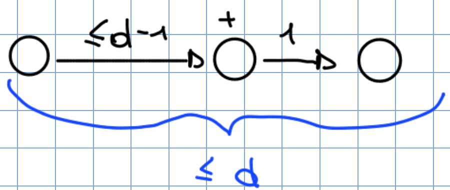
$d = 0$ → caso base

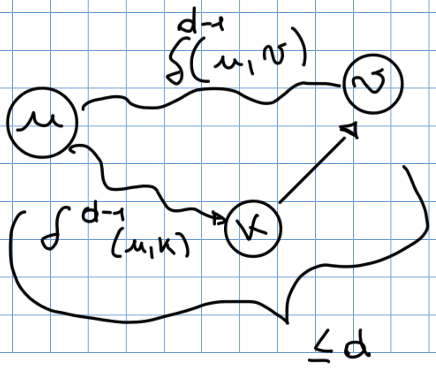
$\delta^{d-1}(u,v),\ \forall u,v \in V$

$\delta^{d}(u,v),\ \forall u,v \in V$



per avere d archi





Come faccio a capire quale cammino è migliore?

in questo modo

$$\text{if } \left(\delta^{d-1}(u,k) + u(k,v) < \delta^{d-1}(u,v)\right)$$
$$\text{then } \delta^{d}(u,v) = \delta^{d-1}(u,k) + w(k,v)$$
$$\text{else } \delta^{d}(u,v) = \delta^{d-1}(u,v)$$

Come individuiamo k?

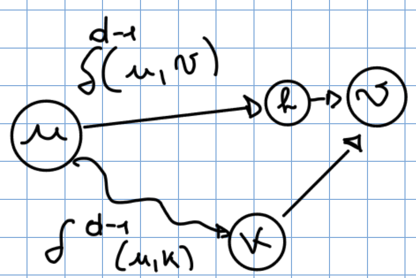
Lo facciamo sempre $\forall k \in V$

Data la matrice

$$l^{d}(i,j) = \begin{cases} w[i,j] & \text{se } d=1 \\ \min\limits_{1\le k\le \underbrace{n}_{|V|=n}} \left(l^{d-1}(i,j),\ l^{d-1}(i,k) + \overset{\text{peso}}{w(k,j)}\right) \end{cases}$$

il primo termine si può cancellare perché contenuto nel secondo

$$l^{d}(i,j) = \begin{cases} w[i,j] & \text{se } d=1 \\ \min\limits_{1\le k\le n} \left(l^{d-1}(i,k) + w(k,j)\right) \end{cases}$$



infatti se $h=k$ $\quad l^{d-1}(i,h) + \overset{\text{peso}}{w(h,j)} = l^{d-1}(u,v)$

matrice dei cammini — matrice archi

```
Extend-APSP(l^d, W)
    l^{d+1} ← new Matrix(n,n)
    for i = 1 to n do
        for j = to n do
            l^{d+1}[i,j] to n do
            for k ← 1 to n
                if l^d [i,k] + w[k,j] < l^{d+1}[i,j]
                then l^{d+1}[i,j] ← l^d[i,k] + w[k,j])
```

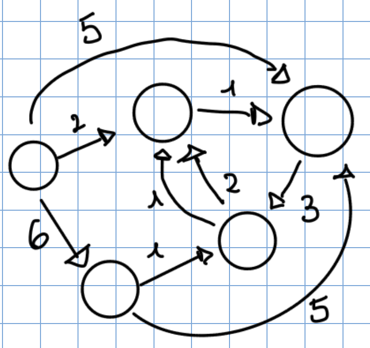
$O(V^3)$ per passare da d a $d+1$

$d=0 \to d=1 \to \dots \to d=V-1$

```
APSP(W)
    l^1 ← W
    for d = 2 to n-1 do
        l^d = Extend-APSP(l^{d-1}, W)
return l^{n-1}
```

$O(V^4)$

è uguale a Bellman-Ford



$W = l^1$

	1	2	3	4	5
1	0	∞	2	∞	1
2	2	0	∞	6	5
3	1	∞	0	∞	∞
4	∞	∞	1	0	5
5	∞	∞	3	∞	0

$W = l^2$

	1	2	3	4	5
1	0	∞	2	∞	1
2	2	0	4	6	3
3	1	∞	0	∞	2
4	2	∞	1	0	5
5	4	∞	3	∞	0

$W = l^3$

	1	2	3	4	5
1					
2					
3					
4					3
5					

ad ogni passaggio il numero di archi aumenta fino ad ottenere un grafo completo

È troppo lungo e lento, come lo facciamo meglio?

A e B sono $n \times n$

```
Matrix-multiply(A,B)
    C = new Matrix(n·n)
    for i=1 to n do
        for j = 1 to n do
            C[i,j] ← 0
            for k = 1 to n do
            C[i,j] ← C[i,j] + A[i,k] · B[k,j]
```

$A^p = A \cdot A \cdot A \dots A$

$L^p = \underbrace{L^1 \, L^1 \, L^1 \, \dots \, L^1}_{p}$

$A^1 = A$

$A^2 = A \times A$

$A^3 = A^2 \cdot A$

$A^4 = A^2 \cdot A^2$

$A^8 = A^4 \times A^4$

$A^{16} = A^8 \times A^8$

non devo fare per forza tutti i passaggi ma salto alcuni usando gli step precedenti

nel nostro caso diventa

$L^1 = W$

$L^2 = W \cdot W$

$L^3 = L^2 \cdot W$

$L^4 = L^2 \cdot L^2$

il nostro algoritmo quindi può diventare:

$$L^d(i,j) = \begin{cases} W[i,j] & \text{se } d = 1 \\ \min\limits_{1 \le k \le n} \left(L^{d/2}(i,k) + L^{d/2}(i,k) \right) \end{cases}$$

Fast - APSP(W)

$l^1 \leftarrow W$

$d = 1$

while $d < n-1$ do $\quad O(\log V)$

$\quad l^{2d} = \text{extend-APSP}(l^d, l^d)$

$\quad d = 2d$

return l^d

$O(V^3 \log V)$

se mettiamo un iterazione in più capiamo se ci sono cicli negativi