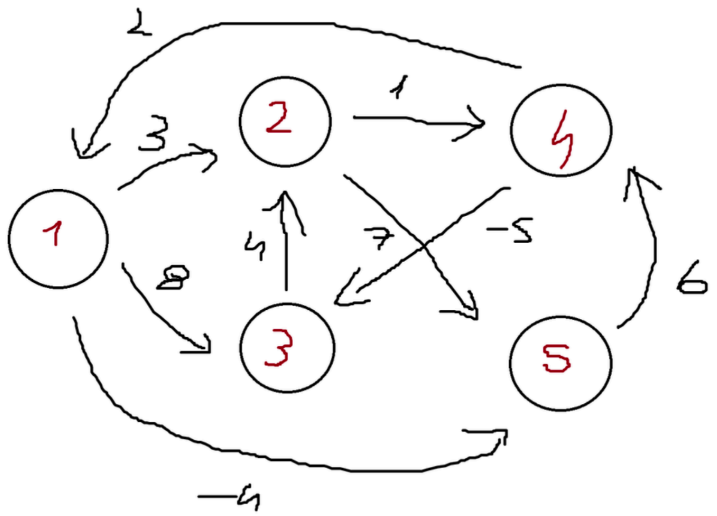
Grafi

Friday, 17 November 2023 15:27

- Prima di iniziare, noi ragioneremo con le seguenti assunzioni:
  - Il grafo non ha cicli negativi
  - Il grafo non ha cappi
  - Noi lavoreremo con grafi pesati
    Che sono:
    - Un insieme di vertici $V = \{1, \dots, n\}$
    - Una relazione $E \subseteq V * V$
      -> Le frecce
    - Una funzione $w: E \to \mathbb{R}$
      -> Il costo delle frecce

  $G = (V, E, w)$

  Esempio di grafo accettato:

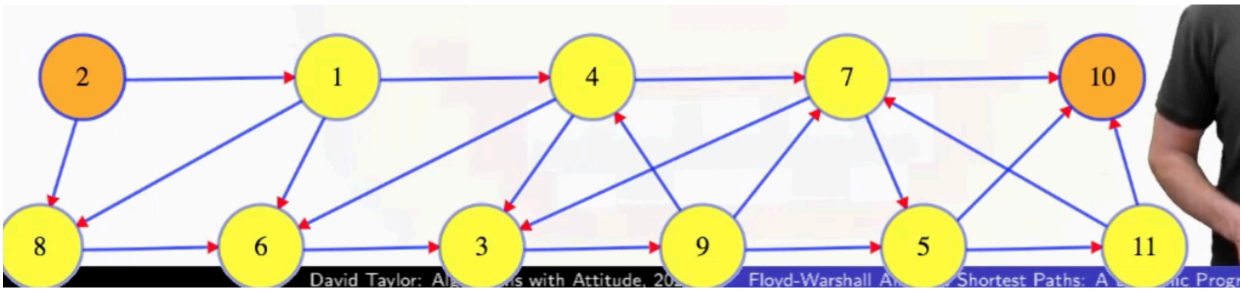


  Scritto attraverso la matrice di adiacenza:

	1	2	3	4	5
1	0	3	8	Inf	-4
2	Inf	0	Inf	1	7
3	Inf	4	0	Inf	Inf
4	2	Inf	-5	0	Inf
5	Inf	Inf	Inf	6	0

- Problema: trovare il cammino più breve di costo minore tra 2 nodi

- Spiegazione:



Supponiamo che vogliamo raggiungere il percorso più corto tra 2 a 10
L'algoritmo utilizza un valore k che ci dice "noi possiamo usare i vertici che sono minore uguali di k"
Esempio:

  - Usiamo k=9?
    - SI, allora il risultato sarà
      Il percorso più corto tra 2 e 9 + il percorso più corto tra 9 e 10

    E lo confrontiamo con k=k-1=8
  - Usiamo k=8?
    - Si, allora il risultato sarà
      Il percorso più corto tra 2 e 8 + il percorso più corto tra 8 e 10
      Correzione per dopo: No siccome, da 8 a 10 passa anche per 9, che però è 9>=k

    E lo mettiamo con il minimo di k-1:
  - Usiamo k=k-1?
    - Si, allora il risultato sarà
      Il percorso più corto tra i e k + il percorso più corto tra k e j

    E così via fino a k=0

Da notare questi casi:

  - Quando k=0 e i=j, allora stiamo chiedendo "come arrivare allo stesso nodo senza muoverci"
    Costo = 0
  - Se k=0 e (i, j) esiste nella nostra matrice della adiacenze (E)
    E quindi esiste una relazione tra i e j
    Allora dobbiamo tornare il peso tra i e j
    Costo = w_{ij}
  - In caso contrario, se non abbiamo una relazione vuol dire che non c'è un arco diretto tra i e j
    Quindi diciamo che non è raggiungibile, che è la stessa cosa di dire che il tempo è infinito
    $Costo = \infty$
  - Quando invece k>0, noi dobbiamo fare tutto ciò che abbiamo detto sopra,
    Il minimo tra:
      Il percorso più corto tra i e j non utilizzando il nodo k

        - Il percorso più corto tra i e j non utilizzando il nodo k
          d_{ij}^{k-1}
        - Il percorso più corto tra i e j utilizzando il nodo k
          E quindi, k è il nostro nodo intermedio, e quindi dobbiamo fare la somma di
          Ciò che c'è prima di k e ciò che c'è dopo
          $d_{ik}^{k-1} + d_{kj}^{k-1}$
- Ricorsione:

$$d_{i,j}^{k} = \begin{cases} 0 & k = 0 \wedge i = j \\ w_{ij} & k = 0 \wedge (i,j) \in E \\ \infty & k = 0 \wedge (i,j) \notin E \\ MIN\left(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right) & else \end{cases}$$

Sottostruttura ottima:

$$d_{i,j}^{k} = \begin{cases} 0 & k = 0 \wedge i = j \\ w_{ij} & k = 0 \wedge (i,j) \in E \\ \infty & k = 0 \wedge (i,j) \notin E \\ d_{ij}^{k-1} & k \notin Cammino \\ d_{ik}^{k-1} + d^{k-1} & k \in Cammino \end{cases}$$

```
ShortPath(i, j, k, W):
        If k==0:
                If i==j:
                        Return 0
                Return W[i, j]
        Return min(ShortPath(i, j, k-1, W), ShortPath(i, k, k-1, W) + ShortPAth(k, j, k-1, W))
```

- Iterativa:

CM(V, E, W, n):

$D_0 = W$

$P_0 = [n, n] of\ NIL$

```
        For i=1 to n:
                For j=1 to n:
                        If i != j and w[i, j] != inf:
```

$P_0[i,j] = i$

```
        For k=1 to n:
                For i=1 to n:
                        For j=1 to n:
```

$D_k[i,j] = D_{k-1}[i,j]$

$P_k[i,j] = P_{k-1}[i,j]$