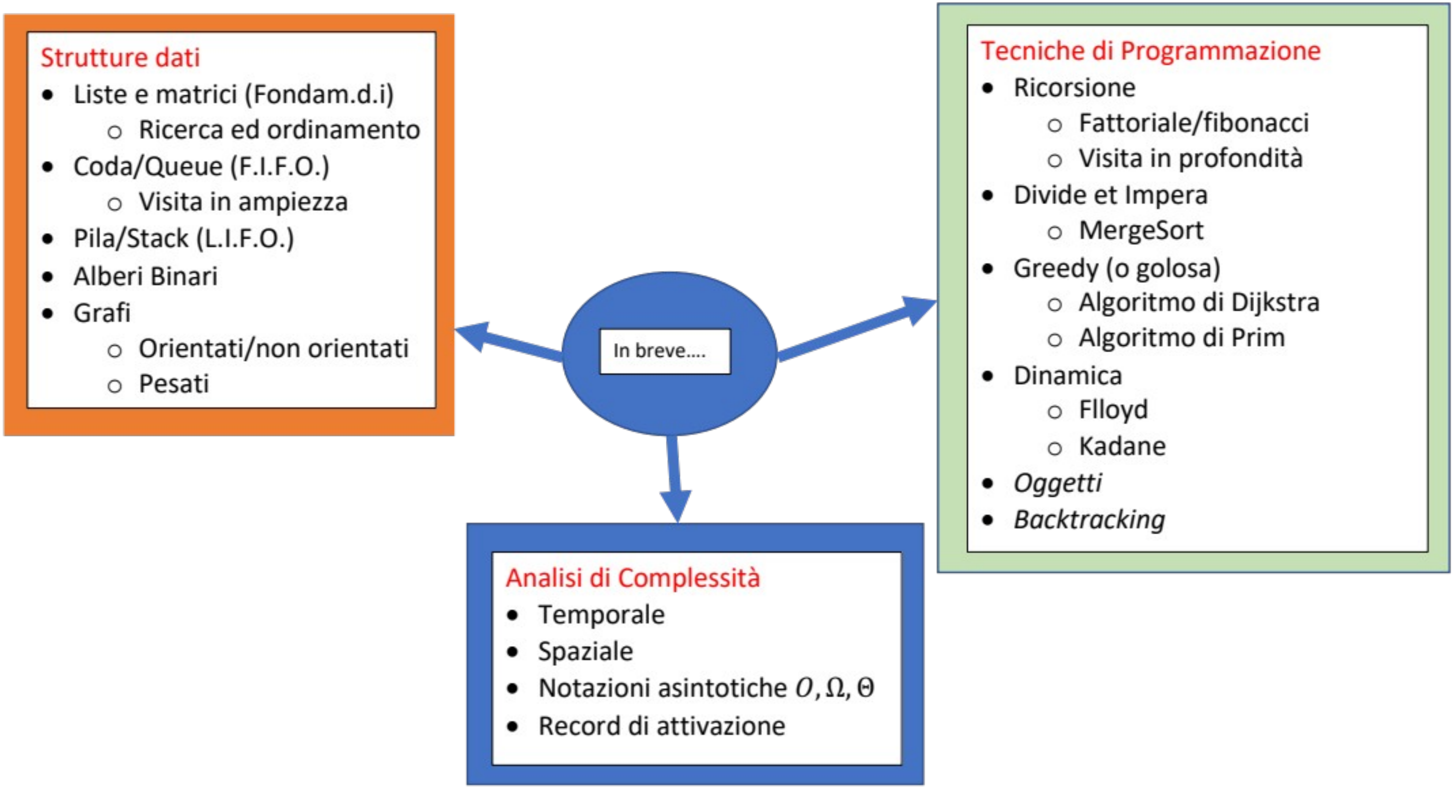
Medicina e Tecnologie TD
Lezione 7

Pierangelo Veltri
pierangelo.veltri@unical.it

Overview


Strutture dati
Liste e matrici (Fondam.d.i)
Ricerca ed ordinamento
Coda/Queue (F.I.F.O.)
Visita in ampiezza
Pila/Stack (L.I.F.O.)
Alberi Binari
Grafi
Orientati/non orientati
Pesati
In breve….
Tecniche di Programmazione
Ricorsione
Fattoriale/fibonacci
Visita in profondità
Divide et Impera
MergeSort
Greedy (o golosa)
Algoritmo di Dijkstra
Algoritmo di Prim
Dinamica
Flloyd
Kadane
Oggetti
Backtracking
Analisi di Complessità
Temporale
Spaziale
Notazioni asintotiche O, Ω, Θ
Record di attivazione

Problemi ed algoritmi su grafi

- Visita dei grafi
  - BFV(Breadth-first visit), DFV (deep-first-visit)
- Cammino minimo (a partire da un nodo) – Single-source shortest-path:
  - algoritmo di **Djakstra**
  - Bellman-ford algoritmo (anche per pesi negativi)
- Cammino minimo per ogni coppia di nodi
  - **Floyd-Warshall**
- Minumum Spanning Tree
  - Algo **Kruskal**
- Chiusura transitiva
  - Algoritmo di **Floyd-Warshall**
- Ordinamenti topologici (esempio per scheduling: se non ha terminato un evento non può partire il successivo)

Definizioni fondamentali

Sia G un grafo orientato. Se esiste un arco (v,w) allora diremo che w è ***adiacente*** a v e che l'arco *esce* da v ed *entra* in w.

Il *grado di entrata* di un nodo v è il numero di archi entranti in esso e il *grado d'uscita* è il numero di archi uscenti, cioè il numero di nodi adiacenti a v.

Definizioni fondamentali

Sia G un grafo non orientato.

Se esiste un arco(v,w) allora diremo che v e w sono adiacenti e che quest'arco è lo stesso dell'arco (w,v).

Il *grado* di un nodo v è il numero di nodi adiacenti a v.

Definizione di cammino

Un *cammino* da *v* a *w* in un grafo (orientato o non) è una sequenza di *k*-1 archi distinti

$$((v_1, v_2), (v_2, v_3), \ldots, (v_{\{k-1\}}, v_k)) \text{ tale che}$$

$$v = v_1 \;\; e \quad\;\; w = v_k$$

Per convenzione si assume che esiste sempre un cammino da un nodo a se stesso di lunghezza 0 (cammino *nullo*).

Definizione di cammino (continua)

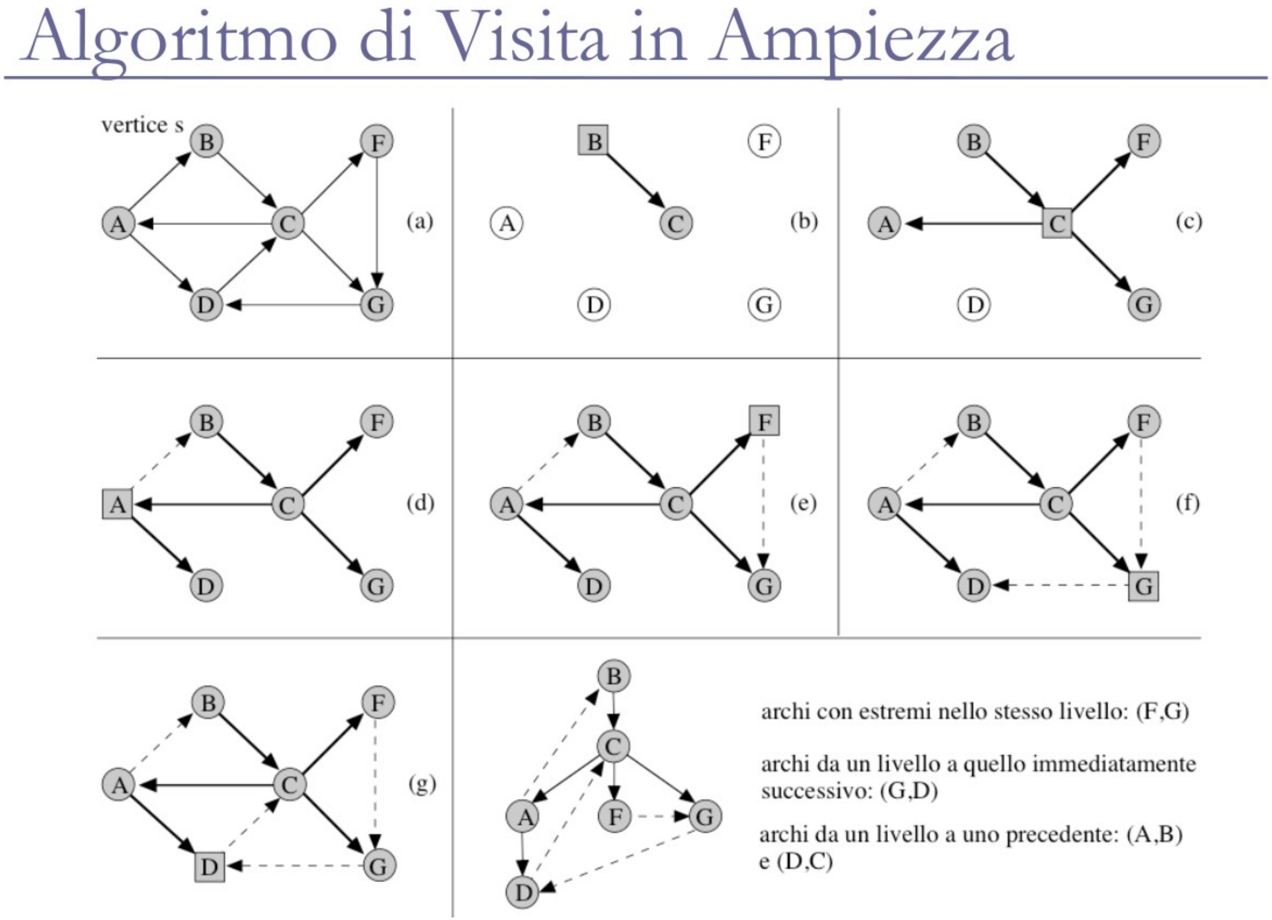
Un cammino è detto *semplice* se tutti i nodi

$v_1, v_2, \dots, v_k$ *s*ono distinti tranne eventualmente v_1 e v_k che possono coincidere (in questo caso il cammino è detto *ciclo*).

Algoritmi di Visita

- Una visita di un grafo G permette di esaminare i nodi e gli archi di G in modo sistematico
- Problema di base in molte applicazioni
- Esistono vari tipi di visite con diverse proprietà: in particolare:
  - visita in ampiezza (BFV=breadth first visit) e
  - visita in profondità (DFV=depth first visit)

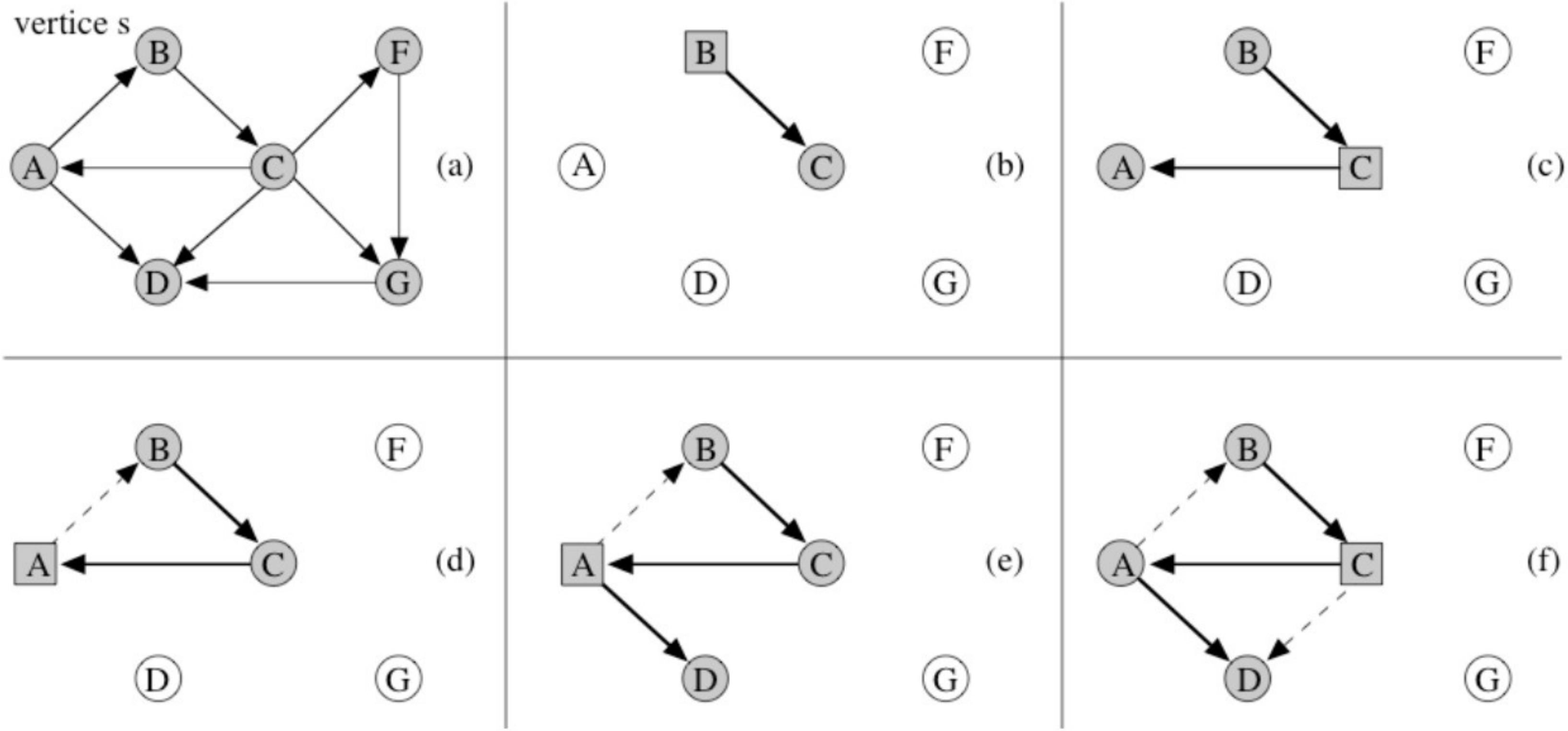
Vediamo bene le visite ...



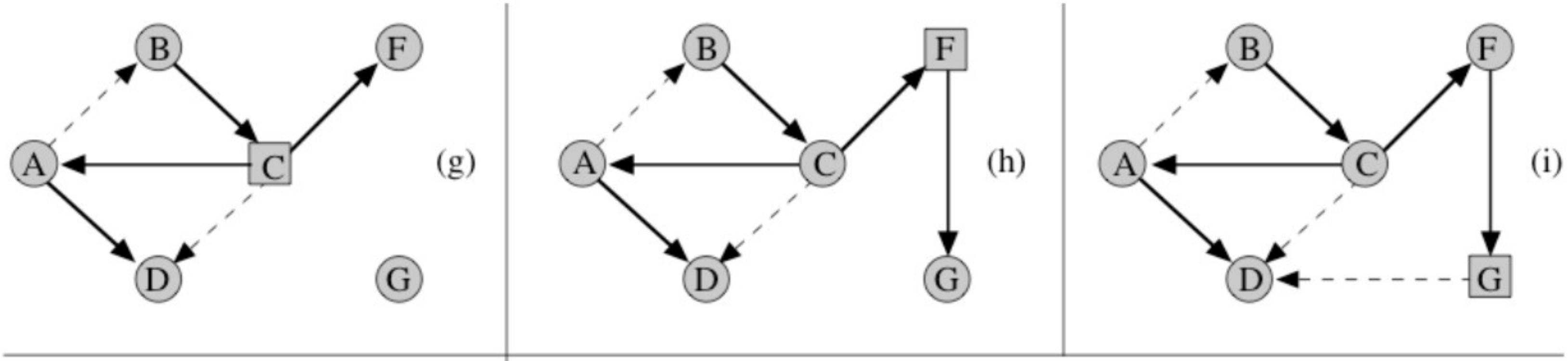
Algoritmo di Visita in Ampiezza

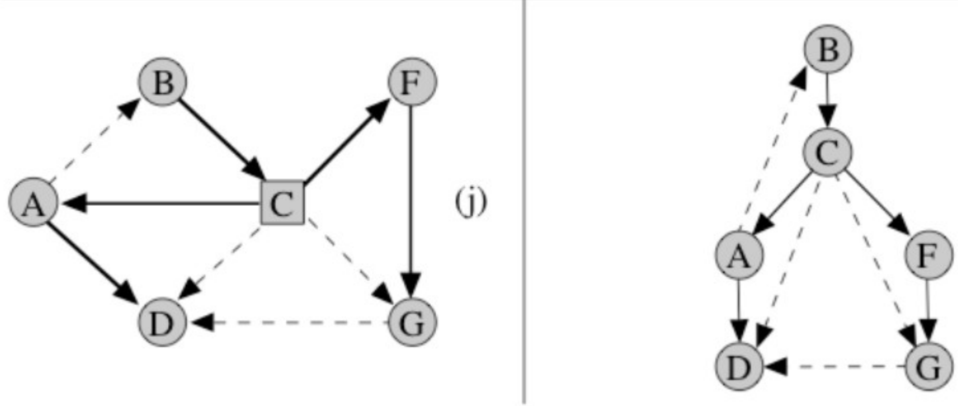
```
algoritmo visitaBFS(vertice s) → albero
    rendi tutti i vertici non marcati
    T ← albero formato da un solo nodo s
    Coda F
    marca il vertice s
    F.enqueue(s)
    while ( not F.isempty() ) do
        u ← F.dequeue()
        for each ( arco (u, v) in G ) do
            if ( v non è ancora marcato ) then
                F.enqueue(v)
                marca il vertice v
                rendi u padre di v in T
    return T
```

Algoritmo di Visita in Profondità



Algoritmo di Visita in Profondità





archi in avanti: (C,D) e (C,G)

archi all'indietro: (A,B)

archi trasversali a sinistra: (G,D)

Algoritmo di Visita in Profondità

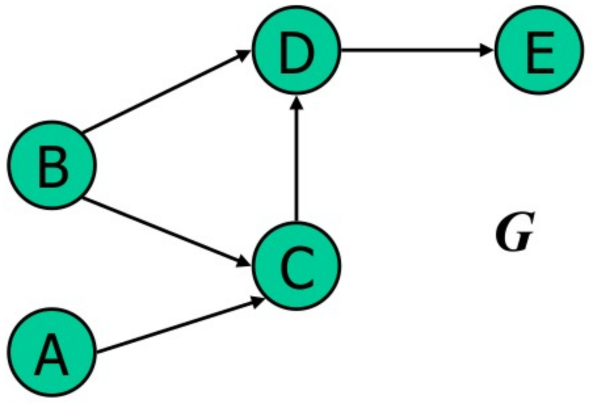
```
procedura visitaDFSRicorsiva(vertice v, albero T)
    marca e visita il vertice v
    for each ( arco (v, w) ) do
        if ( w non è marcato ) then
            aggiungi l'arco (v, w) all'albero T
            visitaDFSRicorsiva(w, T)

algoritmo visitaDFS(vertice s) → albero
    T ← albero vuoto
    visitaDFSRicorsiva(s, T)
    return T
```

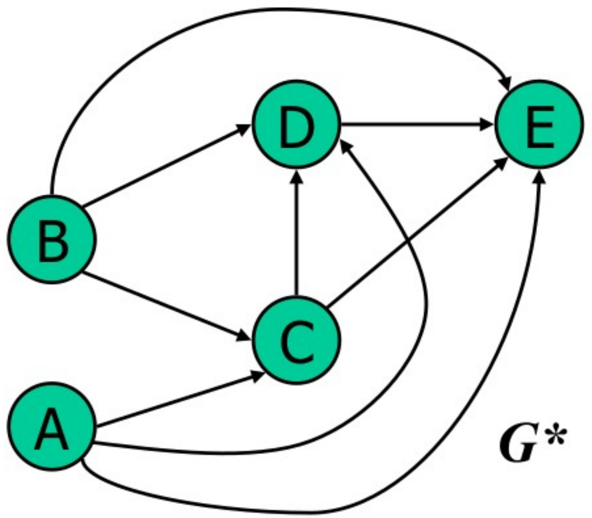
- Esempio di implementazione in Pyhton (Grafi.ipynb) – credits Prof. Alfano
-

Chiusura Transitiva

Dato un grafo orientato G, la chiusura transitiva calcola un grafo ***G* tale che***

- ***G**** ha gli stessi nodi di ***G***
- se ***G*** ha un cammino da u a v ($u \neq v$), ***G**** ha arco da u a v





Chiusura transitiva

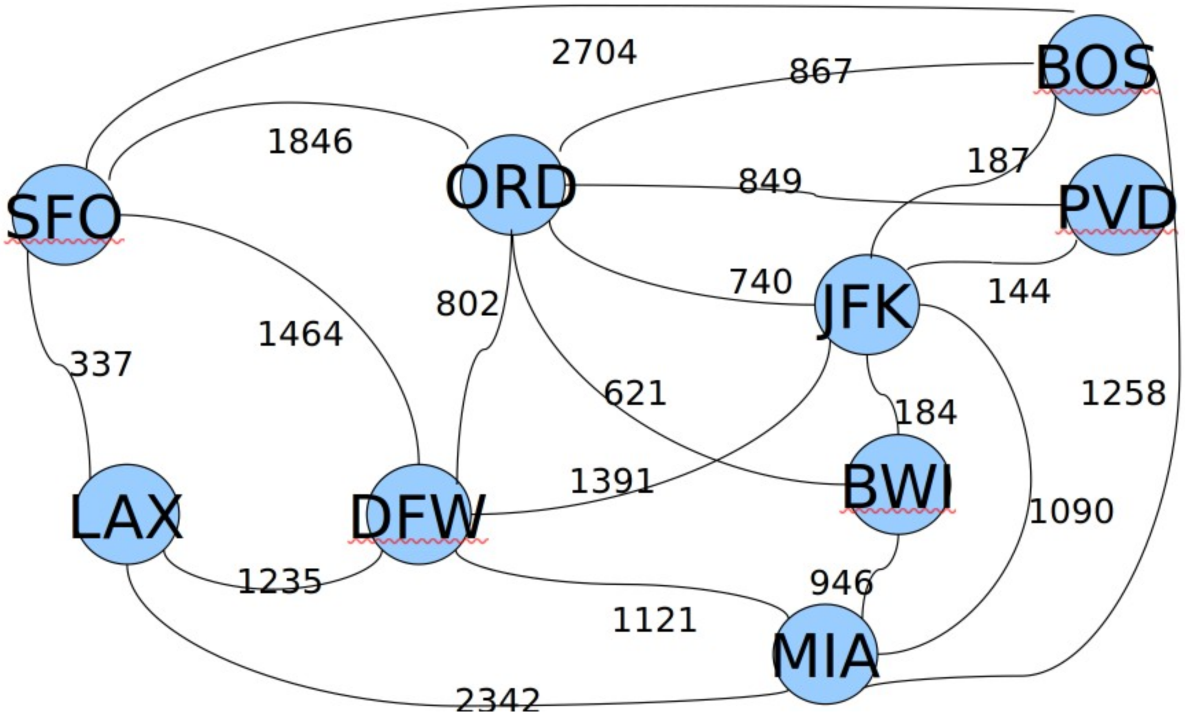
Idea: se esiste l'arco (i,k) e l'arco (k,j) allora deve esistere l'arco (i,j)

```
def closure (G) :
    for k in nodes(G):
      for i in nodes(G):
        for j in nodes(G):
          if is_edge(G,i,k) and is_edge(G,k,j):
            insert_edge(G, i , j )
```

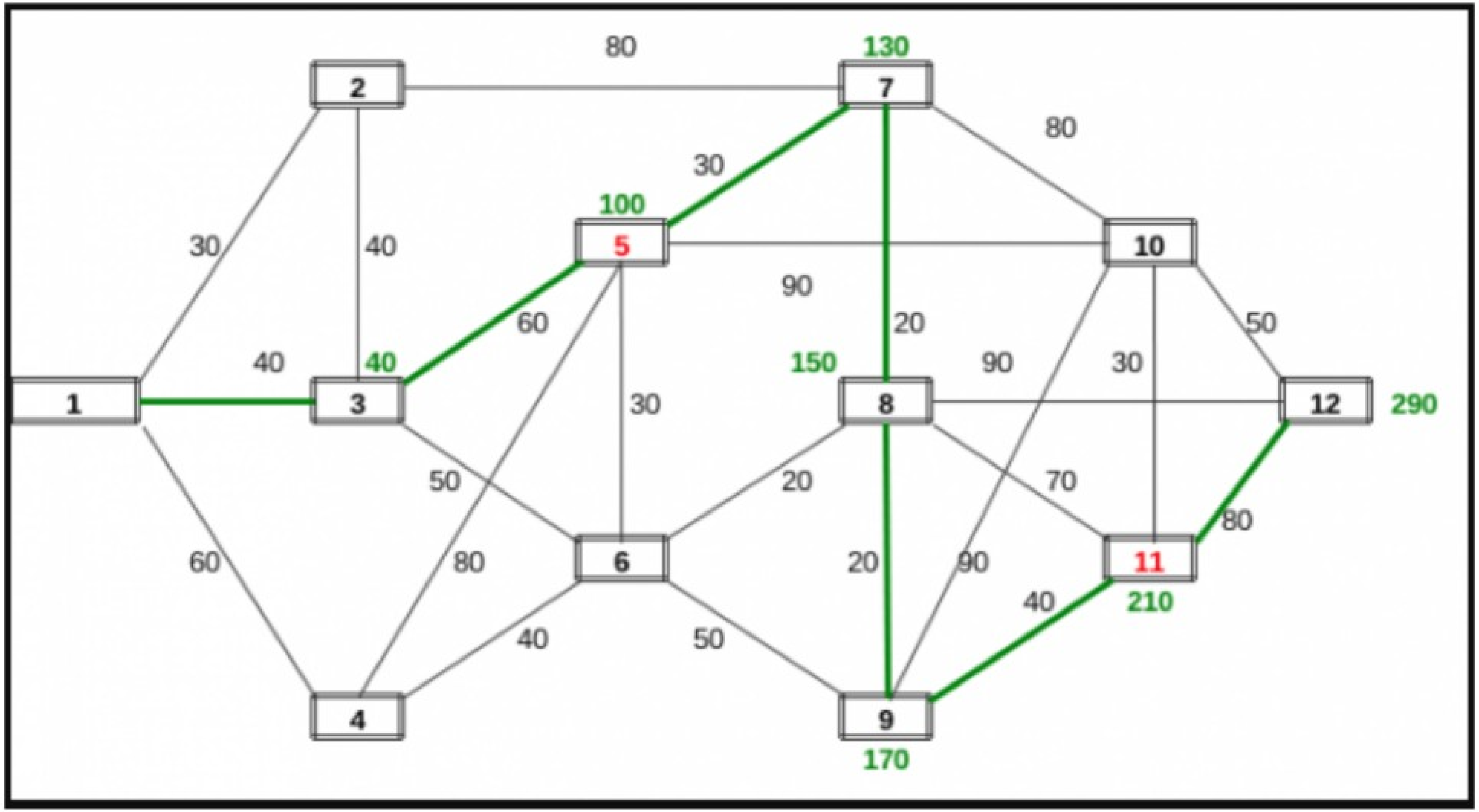
Attenzione: fare prima una copia di G!!!

Es: Cammino minimo partendo da un nodo

Grafo dei collegamenti aeroportuali Archi costi (es. miglia o euro) Calcolare il cammino a costo minimo tra JFK e SFO



Cammini Minimi



Cammini Minimi

Sia $G=(V,E)$ un grafo orientato con costi $w(v_i,v_j)$ sugli archi. Il costo di un cammino $\pi=\langle v_0,v_1,v_2,\ldots,v_k\rangle$ è dato da:

$$w(\pi) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Un cammino minimo tra una coppia di vertici (x,y) è un cammino di costo minore o uguale a quello di ogni altro cammino tra gli stessi vertici.

Distanza (minima)tra due nodi

- La distanza d_{xy} tra due vertici x e y è il costo di un cammino minimo tra da x a y, o $+\infty$ se i due vertici non sono connessi