Algoritmi e Strutture Dati

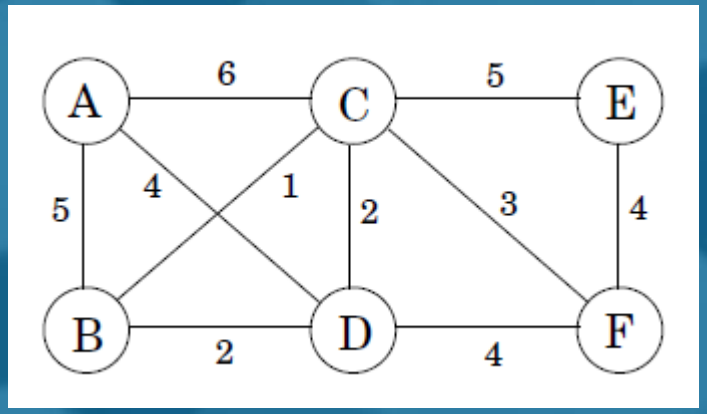
Capitolo 13

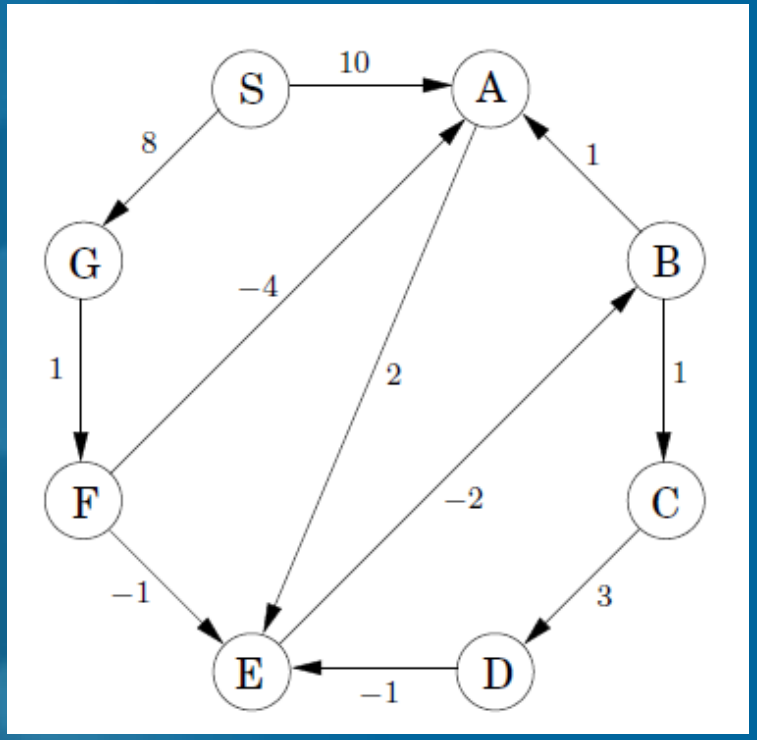
Cammini minimi:

Ordinamento topologico

Grafi pesati

- Grafo pesato: è un grafo G=(V,E,w) in cui ad ogni arco viene associato un valore definito dalla funzione peso w (definita su un opportuno insieme, di solito i reali).





Cammini minimi in grafi pesati

Sia G=(V,A,w) un grafo diretto e pesato con pesi reali sugli archi. Il **costo** (o anche peso o lunghezza, quest'ultimo termine è preferibilmente usato per grafi non pesati) di un cammino (non necessariamente semplice, ci possono cioè essere vertici ripetuti) orientato $\pi=<v_0,v_1,v_2,\ldots,v_k>$ (ovvero, la successione di archi diretti $(v_{i-1},v_i)\in A$ per $i=1,\ldots,k$) è dato da:

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

Un cammino minimo in G tra una coppia di vertici x e y è un cammino orientato in G tra x e y avente costo minore o uguale a quello di ogni altro cammino tra x e y. Tale costo viene detto la distanza in G tra x e y, e verrà denotato con d_{xy} (in particolare, scriveremo $d_{xy}=+\infty$ se i due vertici non sono connessi)

NOTA: Il cammino minimo tra due nodi ovviamente non è necessariamente unico.

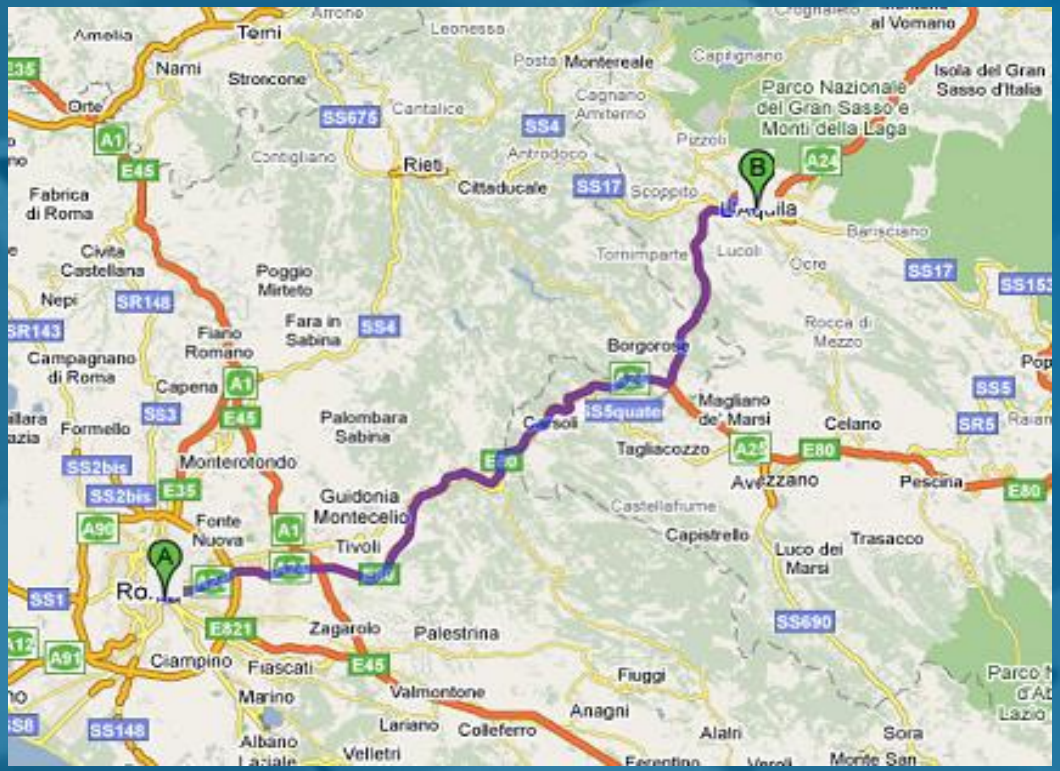
DOMANDA: Qual è il massimo numero di cammini minimi tra 2 nodi in un grafo, espresso in funzione del numero n di nodi?

Algoritmica concreta: il navigatore satellitare

- Sono in auto a Roma in Piazza Dante 12, e devo recarmi ad un appuntamento a L'Aquila, in Via Vetoio 1. Non conosco la strada, ma dispongo di un navigatore satellitare, il quale mi aiuterà ad arrivare a destinazione. Posso scegliere tra due opzioni:
  1. Scegli la strada più breve possibile (funzione obiettivo 1), oppure
  2. Scegli la strada che richiede il minor tempo possibile (funzione obiettivo 2).
- Come calcola la soluzione? Semplice: rappresenta l'intera rete stradale italiana come un grafo diretto $G=(V,A)$, in cui gli archi sono le strade (con i loro sensi di marcia), e i nodi sono le intersezioni fisiche tra le varie strade. Il grafo viene quindi pesato rispetto alla funzione obiettivo selezionata, ovvero rispettivamente:
  1. Lunghezza della strada $\Rightarrow$ funzione peso w_1, sempre positiva;
  2. Tempo di percorrenza $\Rightarrow$ funzione peso w_2, sempre positiva.

  Infine, calcola (**rapidamente!**) il **cammino minimo** in $G=(V,A,w_1)$ o in $G=(V,A,w_2)$ tra il punto di partenza e quello di arrivo.

Il percorso più breve



Quanto ci ho messo a trovarlo? La rete stradale italiana consta di milioni di nodi e di archi, quindi se utilizzassi un algoritmo quadratico impiegherei ordine di migliaia di miliardi ($\sim 10^{12}$) di operazioni, cioè ore! Devo fare meglio, trovare quindi un algoritmo lineare (o quasi) nella dimensione del grafo.

Lower bound temporale per il problema di trovare un cammino minimo tra due nodi x e y? Ovviamente $\Omega(|V|+|A|)$ cioè $\Omega(n+m)$, perché non posso trascurare alcun arco del grafo nella ricerca del cammino minimo!

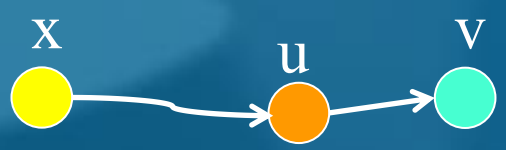
Proprietà dei cammini minimi

- Sottostruttura ottima: ogni sottocammino di un cammino minimo è anch'esso un cammino minimo (ma la concatenazione di cammini minimi non è necessariamente un cammino minimo, in generale!)
- Grafi con cicli negativi: se due vertici x e y appartengono a un ciclo di costo negativo, non esiste nessun cammino minimo finito tra di essi (né tra tutte le coppie di nodi che ammettono un cammino passante per tale ciclo): posso infatti ciclare indefinitamente su tale ciclo riducendo arbitrariamente il costo del cammino!
- Se G non contiene cicli negativi, tra ogni coppia di vertici connessi in G (cioè uniti da almeno un cammino) esiste sempre un cammino minimo **semplice**, in cui cioè tutti i vertici sono distinti (infatti, ogni eventuale ripetizione di vertici indurrebbe un ciclo di costo ≥ 0, che può quindi essere rimosso allorquando cerchiamo un cammino di costo minimo)
- Grafi non diretti: se esiste un arco di costo negativo, allora non esiste nessun cammino minimo finito tra tutte le coppie di nodi che ammettono un cammino passante per tale arco (e quindi, se il grafo è connesso, non esiste alcun cammino minimo nel grafo!): posso infatti passare avanti indietro indefinitamente su tale arco riducendo arbitrariamente il costo del cammino!

Proprietà della distanza fra vertici

- Disuguaglianza triangolare: la distanza tra nodi di un grafo G soddisfa sempre la disuguaglianza triangolare: per ogni tripla di nodi $x,y,z \in V$, $d_{xy} \leq d_{xz}+d_{zy}$ (l'uguaglianza sussiste quando esiste un cammino minimo da x a y che passa per z)
- Condizione di Bellman: per ogni arco (u,v) e per ogni vertice x, essendo $d_{uv} \leq w(u,v)$, dalla disuguaglianza triangolare segue che:

$$d_{xv} \leq d_{xu} + d_{uv} \leq d_{xu} + w(u,v)$$



Ricostruire cammini minimi dalle distanze

Dati due nodi x e y, se conosciamo le distanze da x verso **tutti** i nodi del grafo, è facile risalire in tempo O(n+m) (usando liste di adiacenza) al cammino minimo che congiunge x e y; infatti, dalla condizione di Bellman, un arco (u,v) appartiene ad un cammino minimo da x a v se e solo se $d_{xu}+w(u,v)=d_{xv}$, e quindi posso utilizzare il seguente algoritmo:

```
algoritmo cammino(grafo G, distanze d, vertice x, vertice y)
1.  π ← ⟨y⟩; v ← y
2.  while (v ≠ x) do
3.      for each ( arco (u, v) in G ) do
4.          if (d_xu + w(u, v) = d_xv) then
5.              aggiungi u come primo vertice in π
6.              v ← u
7.              break
8.  return π
```

Nel seguito quindi per risolvere i problemi fondamentali di cammino minimo, ci limiteremo a trovare le distanze tra i nodi.

Tecnica del rilassamento dei cammini

- Partendo da stime per eccesso delle distanze $D_{xy} \geq d_{xy}$ si aggiornano le stime, decrementandole progressivamente fino a renderle esatte
- L'aggiornamento delle stime è basato sul seguente passo di rilassamento (π_{vy} denota un qualche cammino in G tra un generico nodo v e il nodo destinazione y; tale nodo v sarà selezionato secondo un qualche criterio indotto dall'algoritmo soggiacente):

(RILASSAMENTO) **if** $(D_{xv} + w(\pi_{vy}) < D_{xy})$
then $D_{xy} \leftarrow D_{xv} + w(\pi_{vy})$

I tre problemi fondamentali

Dato un grafo G, i tre problemi classici legati ai cammini minimi sono i seguenti (in ordine di difficoltà):

1. Cammino minimo tra due nodi: dati due nodi x e y in G, trovare un cammino minimo in G che congiunge x ed y.
2. Cammini minimi a sorgente singola: dato un vertice s in G, detto sorgente, trovare i cammini minimi da s verso tutti i vertici da esso raggiungibili nel grafo G.
3. Cammino minimo tra tutte le coppie di nodi: trovare un cammino minimo in G che congiunge ogni coppia di vertici x ed y di G.

Nel seguito ci concentreremo sui problemi 2 e 3, e mostreremo diversi algoritmi risolutivi, ciascuno valido per una particolare classe di grafi

Cammini minimi a sorgente singola: l'albero dei cammini minimi

- Come detto, il cammino minimo tra due nodi non è necessariamente unico, e quindi a maggior ragione l'unione di tutti i cammini minimi da una sorgente verso tutti i nodi del grafo (problema #2) può contenere moltissimi cammini
- Vale però una proprietà interessante, ovvero la seguente: Dato un grafo diretto G che non contiene cicli negativi, e dato un vertice sorgente s, esiste sempre un insieme di cammini minimi da s verso tutti i vertici da esso raggiungibili nel grafo G la cui struttura topologica è quella di un albero orientato dalla radice s verso le foglie (che più propriamente prende il nome di arborescenza), detto appunto **albero dei cammini minimi (ACM)** radicato in s
- Si noti che per la proprietà di sottostruttura ottima, l'ACM contiene anche tutti i cammini minimi tra antenati e discendenti nell'albero
- La stessa proprietà di esistenza dell'ACM vale anche per grafi non diretti che non contengono archi di peso negativo

Esistenza dell'ACM

Teorema: Dato un grafo diretto G che non contiene cicli negativi, e dato un vertice sorgente s, l'ACM radicato in s esiste sempre.

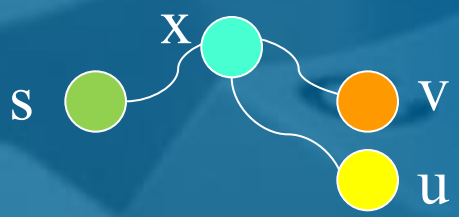
Dim: Forniamo una prova costruttiva. Partiamo dall'albero T che contiene solo s, ed estendiamolo. All'inizio prendiamo un qualsiasi cammino minimo da s ad un nodo v in G. Ovviamente se aggiungiamo tale cammino a T, esso rimane un albero (in particolare, in questo momento T è un cammino):

Ora prendiamo un cammino minimo da s ad un nodo u non contenuto nel primo cammino che abbiamo aggiunto. Se procediamo a ritroso da u verso s, ci sono due possibilità:

Esistenza dell'ACM (2)

1. Non incontriamo alcun nodo del primo cammino aggiunto, a parte s; in tal caso posso chiaramente aggiungere questo secondo cammino a T, il quale continua ad essere un albero;

2. Incontriamo un nodo x che apparteneva al primo cammino; in tal caso, posso aggiungere a T il cammino da x ad u, in modo tale che T rimanga un albero, e chiaramente per la proprietà di sottostruttura ottima dei cammini minimi, il cammino da s a u in T così ottenuto è minimo.

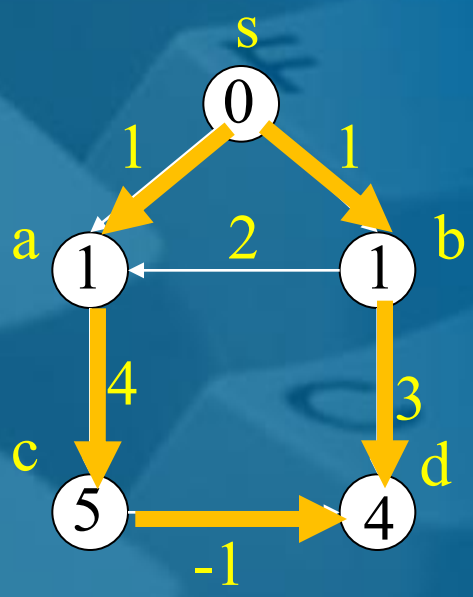


Andando avanti in questo modo e considerando tutti i vertici raggiungibili da s, otterremo l'ACM radicato in s. QED

Osservazione: lo stesso risultato può essere esteso a grafi non diretti che non contengono archi di costo negativo.

Esempio di alberi dei cammini minimi

Si noti che l'ACM non è necessariamente unico, come mostra il seguente esempio:



oppure

Sommario dei risultati a venire

- **Algoritmo basato su ordinamento topologico** (ACM in **grafi diretti aciclici**)
- **Algoritmo di Bellman e Ford** (ACM in **grafi diretti che non contengono cicli negativi** o **grafi non diretti che non contengono archi di costo negatico**)
- **Algoritmo di Dijkstra** (ACM in **grafi (diretti e non diretti) con pesi non negativi**)
- **Algoritmo di Floyd e Warshall** (cammini minimi tra **tutte le coppie di nodi** in **grafi diretti che non contengono cicli negativi** o **grafi non diretti che non contengono archi di costo negatico**)

Algoritmo basato su ordinamento topologico

(ACM in grafi diretti aciclici)

Ordinamento topologico

Definizione 1: Un grafo diretto G si dice aciclico se non contiene cicli orientati (in cui cioè tutti gli archi hanno lo stesso orientamento).

Definizione 2: Dato un grafo diretto $G=(V,A)$, un ordinamento topologico di G è una funzione biettiva $\sigma: V \rightarrow \{1, \ldots, n\}$ tale che se esiste un cammino da u a v in G, con $u \neq v$, allora $\sigma(u)<\sigma(v)$.
(Attenzione: se $\sigma(u)<\sigma(v)$, non è detto che ci sia un cammino da u a v in G).

Teorema: L'ordinamento topologico di un grafo G esiste se e solo se G è **aciclico**.

Dim.: (solo se $\Rightarrow$) Infatti, se per assurdo G ammettesse un ordinamento topologico σ e avesse un ciclo, allora su tale ciclo esisterebbe un cammino da un nodo u a un nodo v ed un cammino da v ad u, cioè $\sigma(u)<\sigma(v)$ e $\sigma(v)<\sigma(u)$ (assurdo).

Per il (se $\Leftarrow$) si veda il prossimo algoritmo costruttivo:

Calcolo di un ordinamento topologico

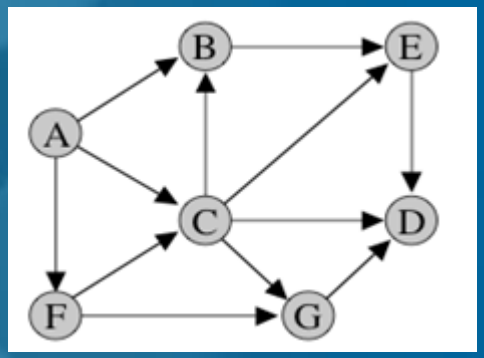
algoritmo `ordinamentoTopologico`(*grafo diretto* G) $\rightarrow$ *lista*
  $\hat{G} \leftarrow G$
  $ord \leftarrow$ lista vuota di vertici
  while (esiste un vertice u senza archi entranti in $\hat{G}$) **do**
    appendi u come ultimo elemento di ord
    rimuovi da $\hat{G}$ il vertice u e tutti i suoi archi uscenti
(*) **if** ($\hat{G}$ non è diventato vuoto) **then errore** il grafo G non è aciclico
  return ord

(*) perché altrimenti in $\hat{G}$ ogni vertice deve avere almeno un arco entrante, e quindi posso trovare un ciclo percorrendo archi entranti a ritroso, e quindi G non può essere aciclico)

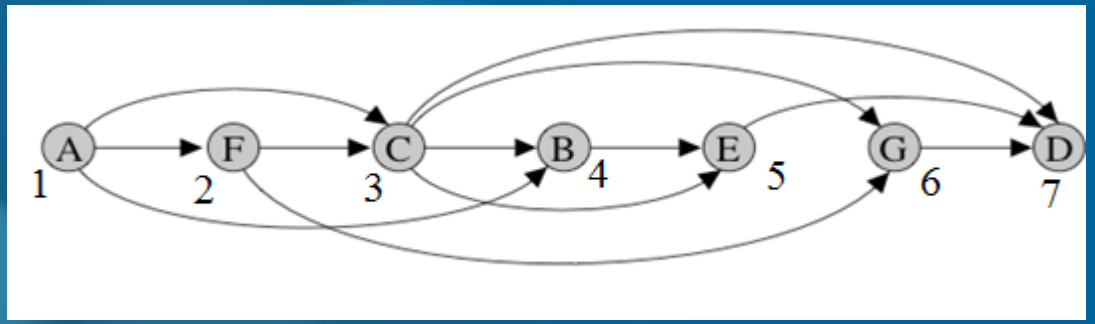
Tempo di esecuzione (con liste di adiacenza):
$\Theta(n+m)$ (dimostrare!)

Esempio

Applicare l'algoritmo di ordinamento topologico sul seguente grafo:



Ad esempio, partendo dall'eliminazione del nodo A, si può ottenere il seguente ordinamento topologico: (si noti che $\sigma(E)<\sigma(G)$, ma non esiste un cammino da E a G). Si noti anche che l'ordinamento topologico non è unico, ad esempio al momento della scelta di B avrei indifferentemente potuto scegliere G.



Cammini minimi in grafi diretti aciclici

Supponiamo di dover trovare l'ACM radicato in un nodo s in un grafo diretto aciclico G. Innanzitutto calcolo un ordinamento topologico *ord*=<u_1,u_2,…,u_i=s,… ,u_n> arbitrario di G. Quindi, inizializzo tutte le stime di distanza da s a $+\infty$, escluso $D_{ss}=0$. Infine, eseguo i rilassamenti in ordine topologico, da sinistra verso destra: infatti, poiché tutti gli archi sono orientati verso destra, quando mi trovo sul generico nodo u_i, la sua stima di distanza e quella di tutti i nodi precedenti è esatta (infatti, tali stime non possono essere ulteriormente rilassate perché non ci sono archi verso sinistra!)

algoritmo `distanzeAciclico`$(grafo\ diretto\ G, vertice\ s) \rightarrow distanze$
  inizializza D tale che $D_{sv} = +\infty$ per $v \neq s$, e $D_{ss} = 0$
  $ord \leftarrow$ `ordinamentoTopologico`(G)
  for $i = 1$ **to** n **do**
    sia u l'i-esimo vertice nell'ordinamento topologico ord
    for each $((u, v) \in A)$ **do**
      if $(D_{su} + w(u, v) < D_{sv})$ **then** $D_{sv} \leftarrow D_{su} + w(u, v)$
  return D

Rilassamento. Si noti che i primi rilassamenti avvengono solo quando viene raggiunto il nodo sorgente s nel ciclo **for**

Tempo di esecuzione (liste di adiacenza): $\Theta(n+m)$

Esercizi

1. Assegnare pesi arbitrari al grafo orientato appena esaminato, e utilizzare l'ordinamento topologico trovato per costruire l'albero dei cammini minimi radicato in C.
2. Quanto costa calcolare tutte le distanze da un nodo sorgente arbitrario in un albero non orientato e con pesi positivi di n nodi?