Algoritmi e Strutture Dati

Capitolo 13

Cammini minimi:

Algoritmo di Floyd e Warshall

Punto della situazione

- Algoritmo basato sull'ordinamento topologico: albero dei cammini minimi in grafi diretti aciclici. Complessità $\Theta(n+m)$ (con liste di adiacenza).
- Algoritmo di Bellman&Ford: albero dei cammini minimi in grafi diretti che non contengono cicli negativi. Complessità:
  - $\Theta(n \cdot m)$ (con liste di adiacenza)
  - $\Theta(n^3)$ (con matrice di adiacenza)
- Domanda: Quanto costa l'algoritmo basato sull'ordinamento topologico se il grafo è rappresentato mediante una matrice di adiacenza? $\Theta(n^2)$ (pensateci, lo chiederò all'orale!)

Algoritmo di Floyd e Warshall

(cammini minimi tra tutte le coppie di nodi in grafi diretti che non contengono cicli negativi)

Premessa

- Una **tecnica algoritmica** può essere vista come un meta-algoritmo che permette di risolvere un'ampia classe di problemi, i quali, per loro natura, si prestano ad essere risolti appunto in modo analogo
- In particolare, abbiamo già visto dettagliatamente la tecnica *divide et impera*
- Oggi vedremo la tecnica della Programmazione Dinamica, che è esattamente quella adottata dall'algoritmo di Floyd&Warshall

Richiamo: la tecnica *divide et impera*

Tecnica top-down:

1. Dividi l'istanza del problema in due o più sottoistanze (sottoproblemi)
2. Risolvi ricorsivamente il problema sulle sottoistanze
3. Ricombina opportunamente le soluzioni dei sottoproblemi allo scopo di ottenere la soluzione globale del problema

Esempi: mergesort, quicksort

Programmazione dinamica

Tecnica bottom-up:

1. Identifica dei sottoproblemi del problema originario, procedendo logicamente dai problemi più piccoli verso quelli più grandi
2. Utilizza una tabella per memorizzare le soluzioni dei sottoproblemi incontrati: quando si incontra lo stesso sottoproblema, sarà sufficiente esaminare la tabella
3. Si usa quando i sottoproblemi non sono indipendenti, ovvero quando vale il principio di sottostruttura ottima della soluzione: una sottosoluzione può essere usata direttamente per costruire la soluzione del problema!

Esempi?

Un esempio banale: **`Fibonacci3`**

algoritmo `fibonacci3`*(intero n)* $\rightarrow$ *intero*

sia Fib un array di n interi

Fib[1] $\leftarrow$ *Fib[2]* $\leftarrow$ *1*

for i = 3 **to** n **do**

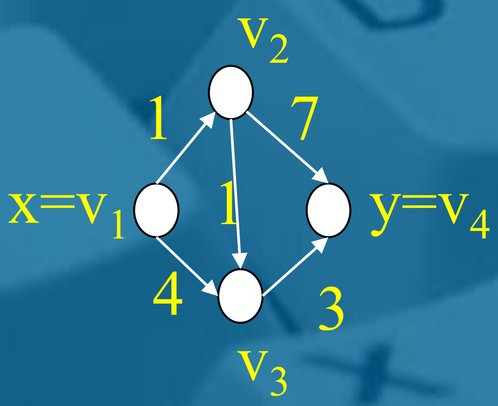
Fib[i] $\leftarrow$ *Fib[i-1] + Fib[i-2]*

return *Fib[n]*

tabella

Approccio di Floyd e Warshall

- Elegante applicazione della tecnica della programmazione dinamica
- Supponiamo di enumerare i vertici di $G=(V,A,w)$ da 1 a n, cioè $V=\{v_1, v_2, \dots v_n\}$. Un cammino minimo k-vincolato da x a y è un cammino di costo minimo tra tutti i cammini da x a y che possono usare come vertici intermedi solo un sottoinsieme (anche vuoto) dei vertici $I_k=\{v_1, v_2, \dots v_k\}$ (in particolare, un cammino minimo 0-vincolato tra due vertici x e y non può usare vertici intermedi, e quindi esiste se e solo se esiste l'arco (x,y) in G)



Tra x e y, il cammino minimo:
- 0-vincolato è lungo $+\infty$
- 1-vincolato è lungo $+\infty$
- 2-vincolato è lungo 8: $\langle x,v_2,y \rangle$;
- 3-vincolato è lungo 5: $\langle x,v_2,v_3,y \rangle$;
- 4-vincolato (ovvero senza vincoli) è lungo 5.

- Idea di Floyd e Warshall: calcolare cammini minimi k-vincolati per k=0,1,…, n

Relazioni tra distanze vincolate

- Sia d^k_{xy} il costo di un cammino minimo k-vincolato da x a y. Chiaramente, valgono le seguenti proprietà:
  - $d^0_{xy} = w(x,y)$ se $(x,y) \in A$, $+\infty$ altrimenti
  - $d^{k-1}_{xv_k} = d^k_{xv_k}$ e $d^{k-1}_{v_kx} = d^k_{v_kx}$
  - $d^n_{xy} = d_{xy}$
- Per le proprietà di cui sopra e per la proprietà di minimalità dei sottocammini di cammini minimi, si ha:

$$d^k_{xy} = \min\{\, d^{k-1}_{xy}, d^k_{xv_k} + d^k_{v_ky} \,\} = \min\{\, d^{k-1}_{xy}, d^{k-1}_{xv_k} + d^{k-1}_{v_ky} \,\}$$

⇒L'algoritmo calcola d_{xy} dal basso verso l'alto, incrementando k da 0 a n

Pseudocodice

algoritmo `FloydWarshall`$(grafo\, G) \rightarrow distanze$
  inizializza D tale che $D_{xy} = w(x,y)$ se $(x,y) \in A$, e $D_{xy} = +\infty$ altrimenti
  for each $v \in V$ **do**
    for each $((x,y) \in V \times V)$ **do**
      if $(D_{xv} + D_{vy} < D_{xy})$ **then** $D_{xy} \leftarrow D_{xv} + D_{vy}$
  return D

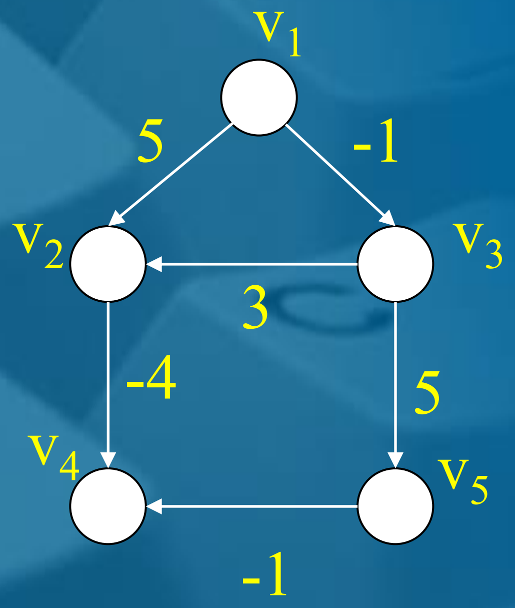
Tempo di esecuzione: $\Theta(n^3)$

(sia con liste di adiacenza che con matrice di adiacenza)

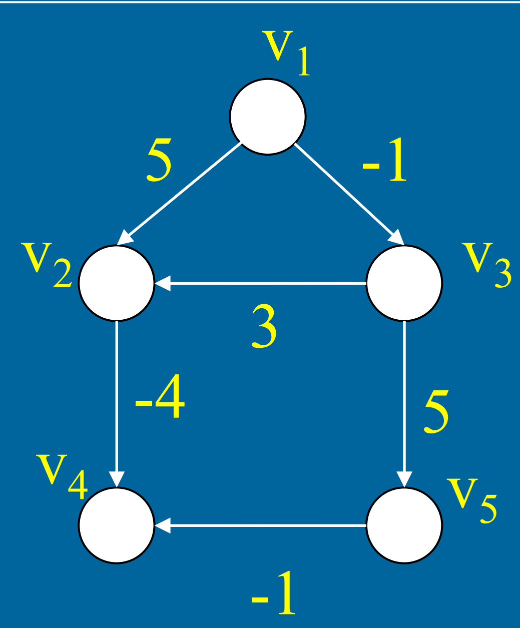
D: Come si confronta con l'applicazione ripetuta di Dijkstra?
R: Utilizzando gli Heap di Fibonacci, n applicazioni dell'algoritmo di Dijkstra richiederanno tempo $O(n\,(m+n \log n)) = O(n\,m+n^2 \log n) = O(n^3)$. Quindi, Dijkstra è più efficiente. Tuttavia, si applica solo su un sottoinsieme delle istanze ammissibili per F&W.

Approfondimento

Applicare l'algoritmo di Floyd e Warshall al seguente grafo:



Soluzione



Posso applicare F&W?

Sì, non ci sono cicli (negativi)!

Inizializziamo la matrice delle distanze:

$$D_0 = \begin{bmatrix} 0 & 5 & -1 & +\infty & +\infty \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & +\infty & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

$$D_1 = \begin{bmatrix} 0 & 5 & -1 & +\infty & +\infty \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & +\infty & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0 & 5 & -1 & \boldsymbol{1} & +\infty \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & -\boldsymbol{1} & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} 0 & \boldsymbol{2} & -1 & -\boldsymbol{2} & \boldsymbol{4} \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & -1 & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & 2 & -1 & -2 & 4 \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & -1 & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

$$D_5 = \begin{bmatrix} 0 & 2 & -1 & -2 & 4 \\ +\infty & 0 & +\infty & -4 & +\infty \\ +\infty & 3 & 0 & -1 & 5 \\ +\infty & +\infty & +\infty & 0 & +\infty \\ +\infty & +\infty & +\infty & -1 & 0 \end{bmatrix}$$

Buon Natale!