Olimpiadi di Informatica 2011
Giornate preparatorie

Dipartimento di Informatica
Università di Torino

marzo 2011

16 – Cammini minimi: l'algoritmo di Dijkstra
(versione 04/04/11)

Visita in ampiezza di un grafo non pesato e cammini minimi.

Ricorda la visita in ampiezza:

```
visitAllInAmpiezza(G) {
   marca tutti i nodi di G come non visitati (o bianchi).
   Q = CodaVuota();
   for each (nodo u del grafo G)
     if (u è non visitato) visitaInAmpiezza(G, u);
}

visitaInAmpiezza (G, s) {
  marca s come visitato; padre[s] = null; accoda s in Q;
  while(Q non è vuota) {
     u = estrai il primo da Q;
     for each (nodo v adiacente a u)
        if (not v visitato) {
          marca v come visitato; padre[v] = u; accoda v in Q;
        }
   }
}
```


Visita in ampiezza con memorizzazione dei livelli

Memorizziamo il livello di ogni nodo nell'albero di visita:

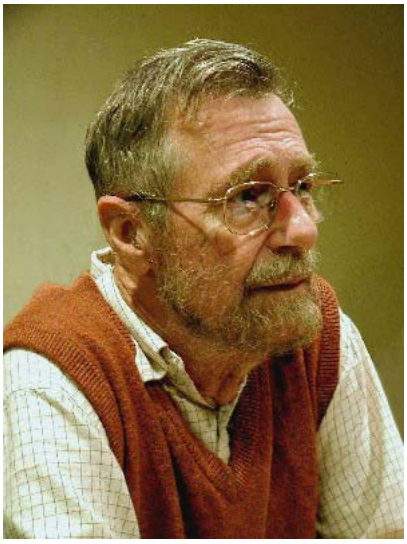
```
visitaInAmpiezza (il grafo G a partire dal nodo s) {
  marca s come visitato; padre[s] = null;
  liv[s] = 0;
  accoda  s in Q;
  while(Q non è vuota) {
     u = estrai il primo da Q;
     for each (nodo v adiacente a u) {
        if (not v visitato) {
          marca v come visitato; padre[v] = u;
          liv[v] = liv[u]+1;
          accoda  v in Q;
        }
     }
  }
}
```

Visita in ampiezza di un grafo non pesato e cammini minimi.

- Proprietà importante. Il livello di un nodo v nell'albero di visita è uguale alla lunghezza del cammino minimo da s a v.

Cammini minimi in un grafo pesato: algoritmo di Dijkstra.

E. W. Dijkstra (pronuncia "daikstra")



E. W. Dijkstra 1930-2002

Edsger Wybe Dijkstra was one of the most influential members of computing science's founding generation. Among the domains in which his scientific contributions are fundamental are

- algorithm design
- programming languages
- program design
- operating systems
- distributed processing
- formal specification and verification
- design of mathematical arguments

In addition, Dijkstra was intensely interested in teaching, and in the relationships between academic computing science and the software industry.

During his forty-plus years as a computing scientist, which included positions in both academia and industry, Dijkstra's contributions brought him many prizes and awards, including computing science's highest honor, the ACM Turing Award. (fonte wikipedia)

Definizioni (ovvie !)

In un grafo orientato pesato:

- peso o costo di un cammino da un nodo s a un nodo t: è la somma dei pesi o costi degli archi che compongono il cammino;
- cammino minimo da un nodo s a un nodo t: un cammino da s a t si dice minimo se è un cammino di peso minimo fra tutti i cammini da s a t; naturalmente possono esistere cammini minimi distinti da s a t, aventi lo stesso peso minimo;

Nota: se non esistono archi con peso negativo, un cammino minimo fra due nodi connessi esiste sempre; se invece vi sono archi con peso negativo, il cammino minimo può non esistere: in particolare, se due nodi appartengono ad un ciclo di costo negativo, non esiste alcun cammino minimo finito fra di essi (perché continuando a "girare" nel ciclo il costo continua a diminuire).

Il problema

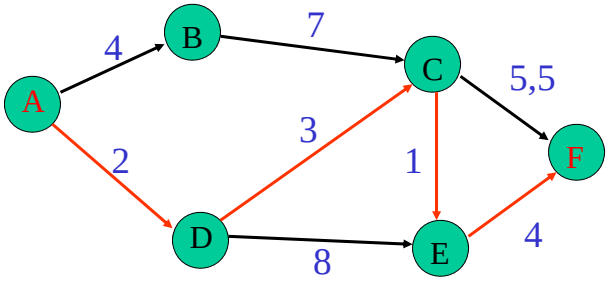
Dato un grafo orientato pesato G, con pesi non negativi (che a seconda dell'applicazione possono rappresentare distanze, tempi di percorrenza, costi di attività, ecc.), e dati un nodo di partenza s (start) e un nodo di arrivo t (target), trovare il cammino di peso minimo da s a t (se ne esistono più d'uno, trovarne uno).

Esempi: trovare su una mappa stradale o ferroviaria o della metro il percorso più breve fra due punti, oppure il percorso più veloce, ecc.

Il problema è generalizzabile nel problema seguente:
dati
- un grafo orientato pesato G con pesi non negativi,
- un nodo di partenza s in G,

trovare per ogni nodo u del grafo il cammino minimo da s a u.

Esempio. Definizione di distanza.



A, D, C, E, F è un (il) cammino minimo da A ad F.

La somma dei pesi degli archi di un cammino minimo da s a t viene chiamata distanza di t da s.

Nell'esempio la distanza di F da A è 10.

Esempi di applicazioni

- Trovare l'itinerario più corto, o più veloce, da un luogo ad un altro su una mappa stradale.
- Trovare il percorso di durata minima fra due stazioni di una rete ferroviaria o di una rete di trasporto pubblico urbano.
- Protocollo di routing OSPF (Open Shortest Path First) usato in internet per trovare la migliore connessione da ogni nodo della rete a tutte le possibili destinazioni.
- ...

Una proprietà

Un sottocammino di un cammino minimo è un cammino minimo.

Dimostrazione

Sia u, ...→ v un sottocammino di un cammino minimo da s a t:

s, ...→, u, ...→, v, ...→ t

Se non fosse minimo, ci sarebbe un altro cammino da u a v di costo inferiore. Ma allora sostituendo tale cammino nel cammino da s a t, si otterrebbe un cammino da s a t di costo inferiore rispetto al cammino minimo.

Esempio: se il percorso più breve fra Torino e Firenze comprende un tratto fra Alessandria e La Spezia, questo sarà evidentemente il percorso più breve fra Alessandria e La Spezia !

Algoritmo di Dijkstra: l'idea.

L'algoritmo di Dijkstra (1959) è una visita in ampiezza in cui ad ogni passo:

- si estrae dalla coda il nodo a distanza provvisoria minima dal nodo di partenza s; la coda Q è quindi una coda con priorità;
- si aggiornano le distanze provvisorie dei nodi adiacenti al nodo estratto dalla coda.

L'uso di una coda con priorità realizzata come *heap* invece che come semplice array o lista fu introdotto da Johnson nel 1977; tale versione dell'algoritmo è quindi talvolta chiamata *algoritmo di Johnson*.

Algoritmo di Dijkstra: un'osservazione.

Per non dover distinguere il caso di nodo non ancora in coda dal caso di nodo in coda con distanza da aggiornare, conviene inserire all'inizio in coda tutti i nodi assegnando loro come distanza provvisoria l'infinito (ossia, ad es., il massimo intero positivo).

Algoritmo di Dijkstra

```
camminiMinimi (il grafo G a partire dal nodo s) {
  marca tutti i nodi di G come non-visitati ;
  for each (nodo x di G)  dist[x] = ∞;
  padre[s] = null; dist[s] = 0;
  Q = PriorityQueue_Vuota;
 for each (nodo x di G)  inserisci x con priorità dist[x] in Q;
 while(Q non è vuota) {
    u = estrai da Q il minimo (cioè il nodo con minima dist);
    marca u come visitato (cioè nero);
    for each (nodo v adiacente a u) {
       if ( v non-visitato   &&   dist[u] + c_uv < dist[v] ) {
         padre[v] = u; dist[v] = dist[u] + c_uv;
         decreasePriority(v,  dist[v], Q);   cioè fa una moveUp
       }
     }
  }
}
```

Nota

Nota: il controllo che il nodo v sia non visitato non è necessario, perché se v è nero allora dist[v] è già minima, e quindi il test dist[u] + c_{uv} < dist[v] darà comunque risultato *false*.

Possiamo quindi ulteriormente semplificare l'algoritmo.

Algoritmo di Dijkstra: versione semplificata.

```
camminiMinimi (il grafo G a partire dal nodo s) {
  for each (nodo x di G)  dist[x] = ∞;
  padre[s] = null; dist[s] = 0;
  Q = PriorityQueue_Vuota;
 for each (nodo x di G)  inserisci x con priorità dist[x] in Q;
 while(Q non è vuota) {
    u = estrai da Q il minimo (cioè il nodo con minima dist);
    for each (nodo v adiacente a u) {
       if (dist[u] + c_uv < dist[v]) {
         padre[v] = u; dist[v] = dist[u] + c_uv;
         decreasePriority(v,  dist[v], Q);  cioè fa una moveUp
       }
     }
  }
}
```

Complessità della versione originale.

- Se la coda con priorità è realizzata come sequenza non ordinata, ciascun inserimento in coda o la modifica della priorità ha complessità $O(1)$, ma l'estrazione del minimo ha complessità $O(n)$, quindi la complessità dell'algoritmo è $O(n^2)$.
- Se la coda con priorità è realizzata come sequenza ordinata, l'estrazione del minimo ha complessità $O(1)$,ma l'inserimento in coda o la modifica della priorità ha complessità $O(n)$, quindi la complessità dell'algoritmo è $O(n^2)$.

Complessità della versione di Johnson.

Siano n = numero dei nodi, m = numero degli archi.

- inizializzazione (di tutti i nodi come bianchi): $O(n)$;
- n estrazioni dalla coda con priorità (heap): $O(n \log n)$;
- ciclo interno:
  - percorre gli archi uscenti da ogni nodo per trovare i nodi adiacenti;
  - per ogni nodo adiacente eventuale *moveUp* nello heap;
  - quindi $O(m \log n)$;

Complessivamente: $O((m+n) \log n)$

Se il grafo è "denso", cioè ha un numero di archi $\Theta(n^2)$, la complessità diventa $\Theta(n^2 \log n)$, quindi peggiore della versione originale quadratica.