LEZIONE 11: DIMOSTRAZIONE DI CORRETTEZZA DELL'ALGORITMO DI DIJKSTRA

L'algoritmo di Dijkstra risolve SINGLE-SOURCE SHORTEST-PATH per grafi orientati nel caso in cui la funzione peso ha valori nonnegativi sugli archi, $w: E \to \mathbb{R}_{\geq 0}$.

Come vedremo, con una buona implementazione, il tempo computazionale dell'algoritmo di Dijkstra è inferiore a quello di Bellman-Ford.

Possiamo pensare all'algoritmo di Dijkstra come a una generalizzazione della ricerca breadth-first ai grafi pesati.

L'algoritmo di Dijkstra mantiene un insieme S di vertici i cui pesi dei rispettivi cammini minimi dalla sorgente sono già stati determinati.

L'algoritmo seleziona ripetutamente il vertice $u \in V - S$ con la più piccola stima del cammino minimo, aggiunge u ad

S, e rilassa tutti gli archi (uscenti) che dipartono da u.

```
DIJKSTRA(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
        INSERT(Q, u)
while Q ≠ ∅
        u = EXTRACT-MIN(Q)
        S = S ∪ {u}
        for each v ∈ G.Adj[u]
            RELAX(u, v, w)
            if the call of RELAX decreased v.d
                DECREASE-KEY(Q, v, v.d)
```

OSS.: L'algoritmo mantiene l'invariante $Q = V \setminus S$ dalle linea 6, fin poi.

<u>THM (CORRETTEZZA DELL'ALGORITMO DI DIJKSTRA)</u>: L'algoritmo di Dijkstra eseguito su un grafo orientato $G=(V,E)$, su una funzione di peso $w: E \to \mathbb{R}_{\geq 0}$, e su un vertice sorgente $s \in V$, termina con $u.d = \underbrace{\delta(s,u)}_{\text{peso di un cammino minimo da } s \text{ a } u}$, $\forall u \in V$.

<u>PROOF</u>: Dimostreremo che all'inizio di ogni iterazione del ciclo while (linee 6-12) si ha che

$$v.d = \delta(s, v) \quad \forall v \in S. \quad (*)$$

L'algoritmo termina quando $S = V$, così che $v.d = \delta(s,v)$ $\forall\, v \in V$.

La dimostrazione di ⊛ avverrà per induzione sul numero di iterazioni (più precisamente, per induzione sulla cardinalità $|S|$ di S) già avvenute.

Ci sono due casi base:

- $|S| = 0 \iff S = \emptyset \iff$ ⊛ vero banalmente;
- $|S| = 1 \implies S = \{s\} \iff s.d = 0 = \delta(s,s)$.

<u>Ipotesi induttiva</u>: $|S| \geq 1$, $v.d = \delta(s,v)$ $\forall\, v \in S$

<u>Passo induttivo</u>: L'algoritmo adesso estrae il vertice $u \in V \setminus S$ e dobbiamo mostrare che $u.d = \delta(s,u)$.

- Se non ci sono cammini da s a u, allora $\delta(s,u) = \infty = u.d$.

p cammino min da s a u

$$
\begin{array}{l}
p = \langle v_0, v_1, v_2, \dots, v_k \rangle \\
\quad\; v_0 = s,\ v_1 = x,\ v_2 = y,\ \dots,\ v_k = u \\
\quad\; \underbrace{s \in S,\ x \in S}_{\subseteq S},\ y \notin S,\ u \notin S
\end{array}
$$

- Se esiste un cammino da s a u, allora sia y il primo vertice di un cammino minimo da s a u tale che $y \notin S$, e sia x il predecessore di y su tale cammino (potrebbe accadere $y = u$ oppure $x = s$).

Siccome y appare non successivamente a u su questo cammino minimo e siccome i pesi sugli archi sono nonnegativi,

$$\boxed{\delta(s,y) \le \delta(s,u)} \leftarrow 4$$

Poiché EXTRACT-MIN (linea 7) restituisce u in virtù del fatto che $u.d$ è minimo tra tutti i $v.d$ per $v \in Q = V \setminus S$, quindi

$1 \rightarrow \boxed{u.d \le y.d}$ inoltre dalla upper-bound property abbiamo $\boxed{\delta(s,u) \le u.d}$.

$\uparrow$
2

Per x vale l'ipotesi induttiva, $x.d = \delta(s,x)$.
Durante l'iterazione del ciclo while in cui x è stato aggiunto a S, (x,y) è stato rilassato e per la proprietà di convergenza

$$\boxed{y.d = \delta(s,y)}. \leftarrow 3$$

Così abbiamo che

$$\delta(s,y) \underset{4}{\le} \delta(s,u) \underset{2}{\le} u.d \underset{1}{\le} y.d \underset{3}{=} \delta(s,y)$$

$$\Rightarrow \quad u.d = \delta(s,u).$$

(in realtà, $\delta(s,y) = \delta(s,u) = u.d = y.d$)

e $u.d$ non cambierà più. $\square$

COR: Dopo aver eseguito l'algoritmo di Dijkstra su un grafo orientato G, con funzione di peso sugli archi non negative, e vertice sorgente s. Il sottografo dei predecessori G_π è un albero dei cammini minimi radicato ad s.

Proof: Immediata, dal thm di correttezza + predecessor subgraph property..

ANALISI

L'algoritmo mantiene la coda di priorità (minimo) Q.

L'algoritmo invoca tre operazioni
- INSERT (linea 5)
- EXTRACT-MIN (linea 7)
- DECREASE-KEY (linea 12)

INSERT $\rightsquigarrow O(1) \times |V|$ volte. $\rightsquigarrow O(|V|)$

EXTRACT MIN viene invocato una volta per ogni vertice.

La lista di adiacenze di un vertice viene analizzata una sola volta durante il corso dell'algoritmo, e si fa questa cosa per ogni vertice.

$$\sum_{v \in V} |Adj(v)| = |E|.$$

Il tempo di esecuzione dell'algoritmo dipende dalla particolare implementazione della coda con priorità Q.

Una semplice implementazione sfrutta l'enumerazione dei vertici da 1 a $|V|$: memorizza v.d nella v-esima posizione di un array.

EXTRACT-MIN $\rightsquigarrow O(|V|)$
+
viene ripetuto $|V|$ volte

$\rightarrow$ Tempo computazionale globale:

$$O(|V|^2 + |E|) = O(|V|^2)$$

Implementando la coda Q con un heap o un Fibonacci-heap si può migliorare il tempo di esecuzione a

$$O(|V| \log(|V|) + |E|). \quad \square$$