

DEFINIZIONE DI GRAFO PESATO: UN GRAFO PESATO E' UN GRAFO $G = (V, E)$ DOVE A OGNI ARCO E' ASSOCIAZIONE UN PESO c

UN COSTO ATTIVAVERSO UNA FUNZIONE $c: E \rightarrow \mathbb{R}$. IL COSTO DI UN CAMMINO DA S A T E' LA SOMMA DEI PESI DI OGNI ARCO CHE COMPOSTO IL CAMMINO.

COSTO DI OGNI ARCO CHE COMPOSTO IL CAMMINO. UN CAMMINO DA S A T SI DICE MINIMO SE HA PESO MINIMO (SENTO DISTANZA) FRA TUTTI I CAMMINI DA S A T. NATURALMENTE PIU' ESISTONO PIU' DI UN CAMMINO MINIMO. SE NON ESISTANO ARCOLI CON PESO NEGATIVO, UN CAMMINO MINIMO FRA DUE NODI CONNESSI ESTATE SEMPRE, ALTRIMENTI PIU' NON ESSERE.

SPIEGAZIONE DA OMOSI: DATO UN GRAFO PESATO, DUNQUE UN GRAFO IN CUI GLI ARCOLI CHE CONNESSIONI UN NODO AD UN ALTRO HANNO UN BLOCO PESO (SENTO ANCHE COSTO), L'ALGORITMO DI DIJKSTRA SI OCCUPA DI TROVARE IL CAMMINO MINIMO DA UN NODO AI PARENTEZZI DI TUTTI GLI ALTRI NODI, ALL'INTERNO DEL GRAFO, PER "CAMMINO MINIMO" SI INTENDE IL PERCORSO DA NODO A NODO ALL'INTERNO DEL GRAFO CON PESO MINIMO. L'ALGORITMO DI DIJKSTRA FUNZIONA SE E SOLO SE I PESI DEGLI ARCOLI DEL GRAFO SONO POSITIVI ALTRIMENTI CIÒ NON SI PUÒ FAR.

DESCRIZIONE INFORMATICA DI CAMMINO: L'IDEA E' CHE POSSIAMO VEDERE L'ALGORITMO COME UNA VISITA IN AMMAGGIO IN CUI A Ogni PESO

• PER I NODI, GIA' VISITATI SAPPIAMO LA DISTANZA ESISTE E ABBIAMO UN ALBERO T
SEI CAMMINI MINIMI

- PER I NODI NEI QUALI VISITARE ABBIAMO UNA DISTANZA PROVVISORIA, CHE' LA LUNGHEZZA DEL CAMMINO PIU' BRUDELLA TROVATO FINORA PASSANDO PER NODI GIA' VISITATI PIU' VICOLO.
- ESTRAMMO DALLA CORSA IL NODO CON DISTANZA PROVVISORIA MINIMA (QUELLO VICOLO DIVENTA DEFINITO)
- SI AGGIORNANO LE DISTANZE PROVVISORIE DEI NODI ADJACENTI AL NUOVO ESTROMMO REFERENDO CONTO DEL NUOVO ARCO IN T.

PSEUDO CODICE:

Dijkstra(G, s)^{Nodo di Partenza}
for each (u nodo in G) $dist[u] = \infty$ // tutti i nodi sono bianchi
 $parent[s] = null$; $dist[s] = 0$ // s diventa grigio ~~potrebbe cambiare~~
 $\times Q = \text{heap vuoto}$
for each (u nodo in G) $Q.add(u, dist[u])$ ~~INSERIMENTO TUTTI I NODI, IN CASA~~
while (Q non vuota)
 $u = Q.getMin()$ // estraggo nodo a distanza provvisoria minima, u diventa nero
 for each ((u, v) arco in G) // v diventa o resta grigio
 if ($dist[u] + c_{u,v} < dist[v]$) // se v nero falso
 $parent[v] = u$; $dist[v] = dist[u] + c_{u,v}$
 $Q.changePriority(v, dist[v])$ // moveUp

COMPLICATITÀ: LA COMPLICATITÀ DELL'ALGORITMO SPERDENDO DATI CON PRIORITY CHE SI SCEGLIE DI USARLI. SE UN DATO A PRIORITY E' REAIZZATO COME:

- SEQUENZA NON ORDINATA
 - INSERIMENTO IN CASA O MODIFICA DELLA PRIORITY: $O(1)$
 - ELIMINAZIONE DEL MINIMO: $O(m)$
 - COMPLICATITÀ TOTALE DELL'ALGORITMO: $O(m^2)$

SEQUENZA DINAMICA

STessa connessione di prima



COMPLEMENTI

SIA $d(M)$ LA DISTANZA (LUNGHEZZA DI UN CAMMINO MINIMO) DA S A M. L'INVERANTE È COMPOSTA DI DUE PARTI:

- 1) PER OGNI NODO M NON IN Q, OSSIA "NERO"

$$\text{dist}[M] = d(M)$$

LA DISTANZA SALVATA È QUILA VERA

- 2) PER OGNI NODO M IN Q

$$\text{dist}[M] = d_{\setminus Q}(M) \rightarrow \text{LUNGHEZZA DI UN CAMMINO MINIMO DA S A M CHE PASSA SOLO PER NODI NERI}$$

TRA NODI M

NODI CHE NON SONO IN Q

CONVENZIONE: SE QUESTO CAMMINO NON ESISTE, OSSIA M NON È RAGGIUNGIBILE DA S SOLO attraverso nodi NERI $\text{d}_{\setminus Q}(M) = \infty$

L'inverante all'inizio:

- 1) VALE BANALMENTE PERCHE' NON CI SONO NODI NERI

- 2) VALE BANALMENTE PERCHE' LA DISTANZA È PER TUTTI INFINTO, TRAMONTE PER S PER CHI VUOLE

COME SI MANTIENE L'INVERANTE:

1) **ESTRAZIONE DEL NODO CON DISTANZA MINIMA:** SUPPONIAMO CHE DALLA COSA VERA ESTENDO UN NODO M CON DISTANZA $\text{dist}[M]$ MINIMA.

PER L'INVERANTE 2, QUESTO SIGNIFICA CHE ESISTE UN CAMMINO M DA S A M CHE PASSA SOLO DA NODI NERI (TRAMONTE M)

E CHE HA LUNGHEZZA PROPRIO $\text{dist}[M]$. Ora DOBBIANO DIMOSTRARE CHE QUESTO CAMMINO È ANCHE IL MINIMO ASSOLUTO TRA TUTTI I CAMMINI DA S A M. SUPPONIAMO PER ASSURDO CHE ESISTA UN ALTO CAMMINO M' DA S A M CON COSTO MINORE.

QUESTO CAMMINO DOVREBBE ALLORA CONTENERE ALMENO UN NODO CHE NON È NERO (NON APPARTIENE A Q), ALTREMENTI AVREBBE GIÀ DOVUTO ESSERE CAPTURATO DA dist . SPA W IL PRIMO NODO NON NERO CHE APPARE IN M' ALLORA IL CAMMINO M' SI PUÒ SCRIVERE COME:

$$M' = M_1 M_2$$

DOVE M_1 È IL CAMMINO DA S A W, CHE PASSA SOLO DA NODI NERI TRAMONTE W. M_2 È IL CAMMINO DA

W A M. PER L'INVERANTE 2, LA DISTANZA A W È GIÀ MESSA COME MINIMA TRA I CAMMINI CHE PASSANO SOLO DA NODI NERI TRAMONTE W. QUINDI IL COSTO DI M_2 È MAGGIORALE O UGUALE AL COSTO DI QUELLO CHE L'ALLENATORE CONOSCE ORA'. QUITO LA SUPPOSIZIONE È FALSA: NON ESISTE CAMMINO PIÙ COSTO.

2) AGGIORNAMENTO DELLA DISTANZA

POICHE' L'INSIEME DEI NODI NERI MOGLIE MASSIMATO PER LEGGERE DI M, OCCORRE PERCIA' AGGIORNARE L'INVERANTE 2: PER OGNI NODO V IN

Q, dist[u] deve essere la lunghezza del minimo per i cammini da s a v

* Dato che devi volta sì trovare un minimo, conviene rappresentare l'insieme dei nodi ancora da visitare come coda a priorità. L'uso di una coda con priorità come heap al posto di un array fu introdotto da Johnson.