Algoritmo di Prim / 22-03

Risolve il problema del minimo albero ricoprente

Dato un grafo connesso, non orientato e pesato
⇓
x albero

→ minimo albero ricoprente (minimum spanning tree) di G è un albero ricoprente di G in cui la somma dei pesi degli archi è minima

Quindi:

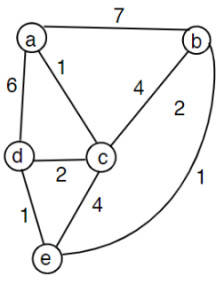
- albero libero (senza radice) → connesso e aciclico
- ricoprente (contiene tutti i nodi)
- somma pesi archi minima

L'algoritmo di Prim somiglia a Dijkstra

→ si prende ogni volta, fra tutti i nodi adiacenti a quelli x cui si è già trovato il minimo (neri), quello connesso a un nodo nero dall'arco di costo minimo.
⇓
nodo più vicino all'albero già costruito

→ si aggiornano in seguito (come in Dijkstra) gli altri nodi

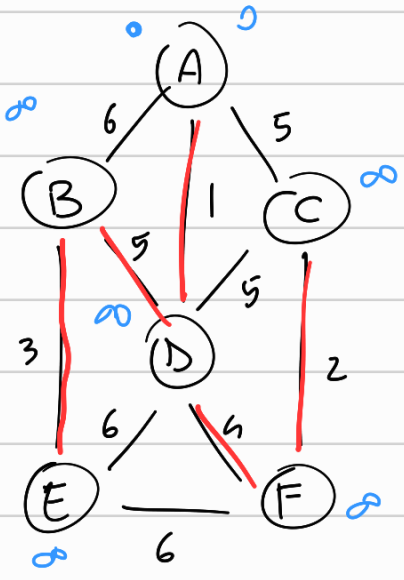
Esempio



estratto	*a*	*b*	*c*	*d*	*e*	*albero*
	0	∞	∞	∞	∞	
a	0	7	1	6		$(a,b),(a,c),(a,d)$
c		4		2	4	**(a,c)**, $(b,c),(c,d),(c,e)$
d					1	**(a,c)**, **(c,d)**, $(b,c),(d,e)$
e		1				**(a,c)**, **(c,d)**, **(d,e)**, (b,e)
b						**(a,c)**, **(c,d)**, **(d,e)**, **(b,e)**

prima colonna: nodo estratto; successive: nodi per i quali viene modificata `dist` e come; ultima: archi dell'albero ricoprente (in grassetto quelli definitivi)

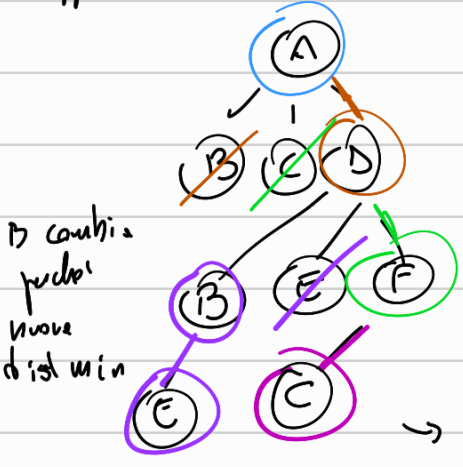
// 22-03



Prim

A	∞	0	0	–	–	–	distanze
B	∞	6	5	5	5	–	provvisorie
C	∞	5	5	2	–	–	fra nodi nei
D	∞	1	0	0	–	–	e albero già
E	∞	∞	6	6	6	3	costruito
F	∞	∞	4	0	–	–	

- doppio cerchio = nero



B cambia perché nuova dist min

escludo quelli già visitati

→ non ho altri nodi, torno indietro

Non è rilevante il nodo sorgente (qui A) infatti ho albero libero: posso avere un altro albero ricoprente di medesimo costo

Codice

Prim (G, s) G non orientato pesato
(se non raggiungo alcuni nodi rimangono a distanza infinito, non è un problema)

for each (u nodo in G) marco u come bianco; (devo marcare i nodi x evitare alberi già fatti)
dist [u] = ∞

dist [s] = 0; parent [s] = null
H = heap vuoto
for each (u nodo in G) H.add (u, dist [u])
while (H non vuoto)
u = H.getMin(); marco u nero
for each (u,v) arco in G // guardo gli adiacenti
if (v bianco) && $c_{u,v}$ (costo) < dist [v] → pesi anche < 0 (no vincolo)
dist [v] = $c_{u,v}$;
parent [v] = u;
H.changePriority (v, dist [v])

" in questo caso cerco un sottografo, non un cammino

Complessità

Identica a Dijkstra

Correttezza Sia T = albero costruito fino a un certo punto (di nodi neri)

Post: T sarà minimo albero ricoprente di G (m.a.r)

Inv:

Inv1) $T \subseteq$ qualche m.a.r che può essere completato a ottenere m.a.r.

Inv2) (ovviamente) $\forall u$ in H, $u \neq s$ $dist[u]$ = "distanza tra u e T" = costo minimo arco tra u e un nodo nero (in T) (se non ho archi, sarà ∞)

Pre: T vuoto perché all'inizio non ho nodi neri (tutti i nodi sono in H)

Pre $\overset{?}{\Rightarrow}$ Inv

- l'albero vuoto è contenuto in m.a.r. ✓ ($\emptyset \subseteq$ m.a.r)
- $dist[u] = \infty$ e non ho un arco nodo nero - u perché no nodi neri ✓

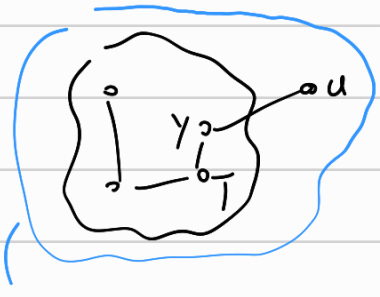
Alla fine: Inv $\wedge$ H vuoto $\Rightarrow$ Post

- $T \subseteq$ m.a.r tutti neri $\Rightarrow$ T è m.a.r ✓
- $dist[u]$ sono minime (tutti in T)

Inv si preserva:

Prima di un'iterazione vale Inv1 $\wedge$ Inv2 con un certo T

Estraggo da H un nodo u t.c. $dist[u] \leq dist[w]$ $\forall w$ in H



u diventa nero, lo aggiungo a T

T diventa $T + (y, u)$ per qualche y

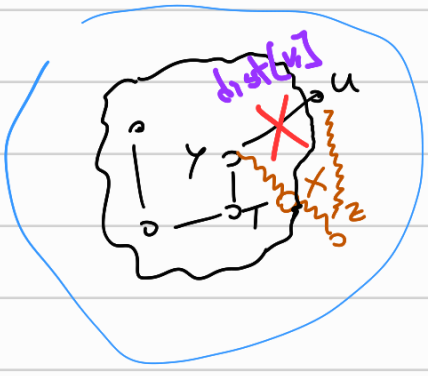
6

a sua volta deve essere contenuto in un m.a.r

Supponiamo per assurdo non sia vero che $(T + \langle y,u \rangle) \subseteq$ m.a.r.

Ma $T \subseteq$ m.a.r. per (Inv 1) quindi in T non c'è l'arco $\langle y,u \rangle$

Poiché ho un albero (tutti i nodi del suo interno sono collegati)
allora deve esserci un cammino che collega y a u con un arco $\langle x,z \rangle$
con x nero, z in H
(da inv 2)



$$C_{y,u} \leq C_{x,z}$$

m.a.r $\setminus$ (x,z) $\cup$ (y,u) = ottengo un singolo albero
togliendo arco

ottengo due alberi (foresta) ↓

quindi ho ottenuto un m.a.r in contrasto con ipotesi assurda
(ho sostituito un arco alla fine, di costo $\leq$ a quello precedente)

- Terminazione banale: funzione di terminazione è il n° di nodi non neri rimasti (heap decrease)