MST (Minimum Spanning Tree)

Introduzione

INPUT

Grafo connesso non orientato pesato $G = (V, E)$ con $W : E \to R^+$ tale che $W(u, v)$ è il peso dell’arco (u, v)

OUTPUT

$T \subseteq E$ aciclico tale che:

1. $\forall\ v \in V,\ \exists\ (u, v) \in T$
2. $W(T) = \sum_{(u,v)\ \in\ T} W(u, v)$ è minimo

$G_T = (V, T) \to$ MST

🚨 In poche parole…
Devo avere ogni nodo collegato almeno da un arco, in modo tale che la somma di tutti questi archi sia la più piccola possibile

Qualche Definizione

- Taglio: Partizione di V in due insiemi V' e V-V'
- Arco attraversa taglio: arco $(u,v) \in E \ \ t.c.\ u \in V' \ \wedge \ v \in V' - V$
- Taglio che rispetta l'insieme: un taglio rispetta un insieme $A \subseteq E$ se nessun arco di A attraversa il taglio
- Arco leggero: arco di peso minimo che attraversa il taglio

Teorema dell'arco sicuro

Dati:

- un grafo connesso non orientato e pesato $G = (V, E)$
- un sottoinsieme A dell'insieme T di archi di un MST
- un qualsiasi taglio $(S, V - S)$ che rispetti A
- un arco leggero (u, v) del taglio

Allora l'arco leggero (u, v) è sicuro per A, ovvero $A \cup \{(u, v)\} \subseteq T$

NOTA! (u, v) è un arco sicuro per A se quell'arco appartiene al MST

Dimostrazione

IPOTESI: Esiste almeno un MST $T \subseteq E \ \ t.c \ \ A \subseteq T$

TESI: trovare MST $T' \subseteq E \ \ t.c \ \ A \cup \{(u, v)\} \subseteq T$

Visto che $(S, V - S)$ rispetta A e (u, v) attraversa il taglio, allora $(u, v) \notin A$.

Abbiamo quindi 2 casi:

Se $(u, v) \in T$

Allora $A \cup \{(u, v)\} \subseteq T$ il quale è una MST

Se $(u, v) \notin T$

Dal momento che T è connesso, allora esisterà un cammino p che va da u a v. Visto che (u, v) attraversa il taglio, allora significa che si trovano da due parti opposte rispetto a quest'ultimo. Esiste allora almeno un arco (x, y) di p che attraversa il taglio.

Sia $T' = (T \setminus \{(x, y\} \cup \{(u, v)\})$:

Sappiamo che $A \subseteq T$ e che $(x, y) \notin A$ visto che attraversa il taglio, allora $A \subseteq T \setminus \{(x, y\}$

A maggior ragione $A \subseteq (T \setminus \{(x, y)\}) \cup \{(u, v)\} = T' \ \Rightarrow A \cup \{(u, v)\} \subseteq T'$.

Verificando il peso di T':

$$w(T') = w(T) - w(x,y) + w(u,v)$$

Dal momento che (u,v) è un arco leggero del taglio attraversato anche da (x,y) allora $w(x,y) \geq w(u,v) \Rightarrow \ w(T') \leq w(T)$

Essendo T un MST, allora lo è anche T', il quale contiene $A \cup \{(u,v)\}$.

Corollario

$A \subseteq T$ è tale che $G_A = (V, A)$ è una foresta con $|V| - |A|$ alberi.

Sia $C = (V_C, A_C)$, con $V_C \subseteq V \ e \ A_C \subseteq A$, una componente connessa di G_A.

$\Rightarrow (V_C, V - V_C)$ è sicuramente un taglio che rispetta A

$\Rightarrow$ un arco leggero di $(V_C, V - V_C)$ è un arco sicuro per A

🚨 In poche parole…
Per trovare un nuovo arco sicuro da aggiungere ad A:
- Considero una delle componenti C della foresta
- Trovo arco leggero che collega un vertice in C con uno non in C

Algo Generico

1. Inizializza un insieme A vuoto
2. Aggiunge ad ogni passo un arco (u,v) tale che $A \cup \{(u,v)\}$ è un sottoinsieme dell'insieme T degli archi di MST
3. Algoritmo termina quando $A = T$, ovvero $G_A = (V, T) \Rightarrow MST$

Algoritmo GENERIC-MST

GENERIC-MST (G,W)

$A \leftarrow \emptyset$

WHILE $|V| - |A| > 1$

trova arco (u,v) sicuro per A

$A = A \cup \{(u,v)\}$

RETURN A

Cosa fa?

1. A rimane aciclico durante le iterazioni
2. $G_A = (V, A)$ ad ogni iterazione è una foresta di $|V| - |A|$ alberi

3. All’inizio, G_A contiene $|V|$ alberi (singoli vertici)
4. Ogni iterazione riduce di 1 il numero di alberi e l’arco sicuro collega sempre componenti distinte di G_A
5. Quando arriva ad un solo albero l’algoritmo termina (ovvero tutti i vertici sono collegati)
6. Il numero di iterazioni è parti a $|V| - 1$

Algoritmo di Kruskal

Algoritmo per trovare MST, tramite l’ordinamento degli archi in ordine crescente di costo e successivamente analizzandoli singolarmente, inserendo l'arco nella soluzione se non forma cicli con gli archi precedentemente selezionati (ovvero connette due componenti diverse di G_A.

Algoritmo definitivo

KRUSKAL-MST (G=(V,E), W)

$A \leftarrow \emptyset$

$E \leftarrow \langle e_1, ..., e_n \rangle$ ordinati per peso non decrescente

FOREACH $v \in V$

MAKE_SET (v)

FOR i from 1 to n

$(u, v) \leftarrow e_i$

IF FIND_SET (u) $\neq$ FIND_SET(v)

$A = \{(u, v)\} \cup A$

UNION (u,v)

RETURN A

Tempo di calcolo

Sapendo che:

- $|E| \geq |V| - 1$
- $\alpha \leq log|V| \rightarrow \alpha \leq log|E|$

L’ordinamento ha tempo $O(|E|log|E|)$, FOREACH invece $O(|V|)$ e infine il FOR complessivamente è $O(|E|\alpha)$. Sommando troviamo:

$$O(|E|log|E| + (|V| + |E|)\alpha) \rightarrow O(|E|log|E|)$$

Algoritmo di Prim

Funzionamento di base

1. Sceglie vertice arbitrario r (componente C all’inizio composta quindi solo da vertice r)

2. Trova l’arco di peso minimo che connette r ad un altro vertice v (entra così anche v in C)
3. Trovo arco di peso minimo che connette un vertice in C ad un vertice v non in C (anche questo entra in C)
4. Ripeto il passo 3
5. Termina quando C comprende tutti i vertici del grafo e quindi coincide con il MST

Proprietà dell’algoritmo

Ad ogni passo:

1. Il sottoinsieme A degli archi di MST aggiunti fanno parte della componente C. La foresta è composta quindi da:
   - $C = (V_C, A)$
   - $|V - V_C|$ componenti di vertici singoli (non ancora inseriti)
2. Il taglio $(V_C, V - V_C)$ rispetta l’insieme A
3. L’arco sicuro è l’arco leggero (di peso minimo) che connette un vertice in C con uno non in C.

Elementi utili

Coda Q

Coda di min-priority che contiene tutti i vertici che non appartengono a C (quindi all’inizio tutti), permettendo di estrarre un vertice v tale che (u, v) è l’arco leggero (peso minimo) che collega un vertice $u \in C$ con un vertice $v \notin C$.

Campi dei vertici

Ad ogni vertice v sono associati due campi:

- **v.key** → minimo valore del peso degli archi (u, v) incidenti in v tale che $u \in C$.
- $v.\pi$ → indica un vertice u tale che (u, v) è l’arco di peso minimo di v.key

> 🚨 All’inizio $v.key = \infty$ e $v.\pi = NIL$ per tutti i vertici tranne per il primo, il quale scelto in modo arbitrario e ha $r.key = 0$.

Ad ogni passo…

1. Viene estratto da Q il vertice u con il minor valore del campo key:
   - l’arco $(u.\pi, u)$ è un nuovo arco di MST
   - u è un vertice che si aggiunge alla componente C
2. Per ogni vertice v adiacente a u, se v è in Q e $v.key > W(u, v)$, vengono aggiornati:
   - $v.key$ al valore $W(u, v)$
   - $v.\pi$ al valore u

3. Algo termina quando Q vuota

Algoritmo

PRIM-MST (G,W,r)

FOREACH $v \in V$

$v.key \leftarrow \infty$

$v.\pi \leftarrow NIL$

$r.key \leftarrow 0$

Aggiungi tutti i vertici di V alla coda Q

WHILE $Q \neq \emptyset$

$u \leftarrow$ *estrai vertice da Q*

FOREACH $v \in adj(u)$

IF $v \in Q$ **AND** $W(u,v) < v.key$

$v.key \leftarrow W(u,v)$

$v.\pi \leftarrow u$

Tempo di calcolo

L'inizializzazione dei valori e l'aggiunta dei vertici a Q hanno tempo lineare $O(|V|)$, anche WHILE $O(|V|)$, l'estrazione del vertice da Q $O(log|V|)$, l'ultimo FOREACH invece $O(|E|)$ e infine l'assegnazione del valore da $W(u,v)$ $O(log|V|)$. Possiamo quindi calcolare il tempo totale:

$$O(|V|) + O(|V|log|V|) + O(|E|log|V|) \rightarrow O(|E|log|E|)$$