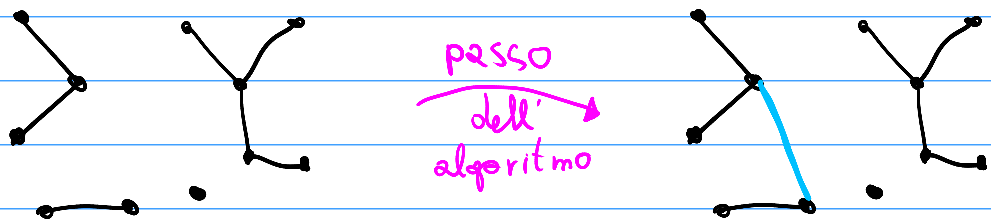
Algoritmo di Kruskal

- Algoritmo che dato un grafo { • semplice • pesato • connesso } $G=(V,E,w)$ produce un albero di copertura minimo $T \subseteq E$.

- Algoritmo greedy: si parte con $T := \emptyset$ e ad ogni passo si aggiunge un arco secondo un criterio di MINIMALITÁ "LOCALE".

Idea: ad ogni passo gli archi scelti compongono varie componenti connesse (senza cicli)



- Si aggiunge a T l'arco di costo minimo
  - che non introduce cicli
  - che unisce due componenti connesse.

- Dopo n-1 archi inseriti, l'albero é completo.

LISTA DI ARCHI

```python
def Kruskal(V,E,W):
    # Ordina rispetto ai pesi in
    # modo non decrescente
    E = sorted(E,key=lambda e: W[e])
    T = []
    for u,v in E:
        if componente(u,T) != componente(v,T):
            T.append((u,v))
    return T
```

scorriamo gli archi dal peso minore al peso maggiore

inseriamo nella soluzione un arco solo se non crea cicli, ovvero se unisce due componenti connesse

Vediamo esempio Precalcolato

- L'algoritmo è semplice, ma non è chiaro come verificare in modo **EFFICIENTE** se due vertici sono in due componenti connesse diverse.

Questo lo vedremo dopo: per ora discutiamo la complessità dell'algoritmo e la sua correttezza

COMPLESSITÁ: per gestire le componenti connesse dovremo costruire delle STRUTTURE DATI apposite, che dovremo mantenere durante l'algoritmo. In tutto avremo un costo, in termini di operazioni, che per ora chiamiamo GESTIONE DELLE COMPONENTI

Per il resto:

- ordinamento della lista di archi $O(|E| \log |E|) = O(|E| \log |V|)$
- ciclo sugli archi della lista ordinata, con $O(1)$ operazioni per ciclo (a parte la gestione delle componenti)
- la gestione delle componenti ci costerà $O(|E| \cdot g(|V|))$ in totale con $g(|V|) \leq O(\log |V|)$

COMPLESSITÁ: $O(|E| \log |V|)$

Thm L'algoritmo di Kruskal trova un albero di copertura di costo minimo

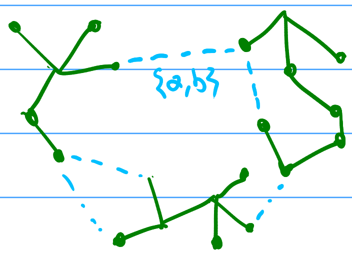
dim

Ⓐ DIMOSTRIAMO CHE PRODUCE UN ALBERO DI COPERTURA

① poiché un arco viene inserito in T solo se connette due componenti, allora in T non ci saranno mai cicli.

② L'algoritmo può terminare senza aver connesso tutti i vertici del grafo?

T =



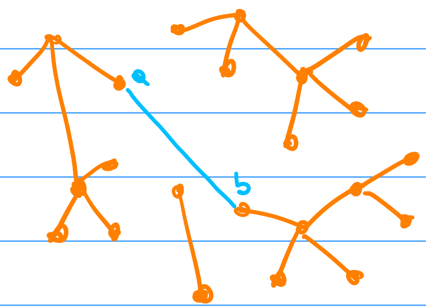
Se alla fine dell'algoritmo T ha più di una comp. connessa allora (visto che G è connesso) ci sono degli archi •- - - - -• che le connettono

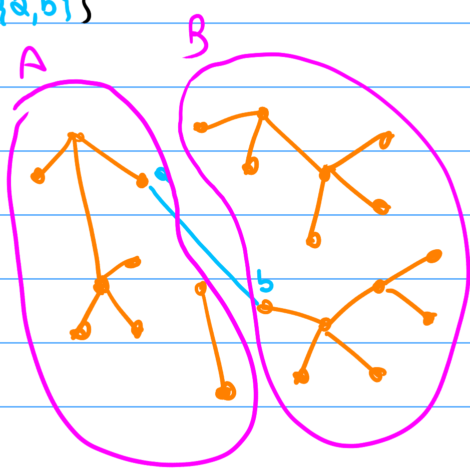
- Sia {a,b} l'arco di costo minimo tra questi.
- Al momento in cui {a,b} è stato analizzato, se non è stato scelto allora a e b erano già nella stessa componente
- Ma allora dovrebbero esserlo anche alla fine del ciclo, perché aggiungere archi in T non separa componenti connesse

③ Dimostriamo che l'albero di copertura ha costo minimo

- In ogni momento l'algoritmo ha una soluzione parziale costituita da tante componenti connesse senza cicli (questa struttura ha il nome di FORESTA, ma non è importante qui). Chiamiamo T questa soluzione parziale

- Dimostriamo che ad ogni passo T è un sottoinsieme di una soluzione minima
  - Caso base: $T := \emptyset$
  - Passo induttivo: $T \longrightarrow T \cup \{\{a,b\}\}$





- A, B è un taglio che non è attraversato da nessun arco di T ma che è attraversato da $\{a,b\}$

- Se $\{a,b\}$ è l'arco di costo minimo che attraversa A, B allora il teorema del taglio ci garantisce il passo induttivo

- Se invece esiste $\{c,d\}$ con $w(\{c,d\}) < w(\{a,b\})$ che attraversa A, B
  - $\{c,d\}$ viene analizzato prima di $\{a,b\}$
  - $\{c,d\}$ non introduce un ciclo nel momento in cui viene analizzato
  - quindi $\{c,d\}$ non esiste

GESTIONE DELLE COMPONENTI CONNESSE

- Mentre costruiamo le soluzioni parziali $T_0, T_1, \dots, T_{n-1}$ abbiamo bisogno di sapere se due vertici u e v sono connessi nella soluzione parziale T_i

- <u>OPZIONE</u> Potremmo fare una DFS o BFS su T_i ogni volta che viene modificata la soluzione, e tenere traccia di quali vertici sono connessi.

- questo costerebbe n volte $O(|V| + |T_i|)$ quindi $O(n^2)$ e sarebbe troppo per grafi con $|E| << n^2$

<u>OSSERVAZIONE</u> $T_0, T_1, \dots$ non sono input arbitrari:
- li stiamo costruendo noi, e quindi potremmo mantenere traccia delle componenti connesse ogni volta che aggiungiamo un arco

  i.e. UNIAMO DUE COMPONENTI CONNESSE

ESERCIZIO 23.2-1

- L'algoritmo di Kruskal può restituire alberi di copertura differenti per lo stesso grafo pesato G, a seconda di come vengono ordinati gli archi di peso uguale.
- Mostrate che per ogni albero di copertura minimo esiste un ordinamento iniziale degli archi tale che T è l'albero prodotto dall'algoritmo di Kruskal

ESERCIZIO 23-1

- Sia G un grafo pesato, connesso, non orientato, con $|E(G)| > |V(G)|$ con TUTTI I PESI SUGLI ARCHI DISTINTI.

① Dimostrate che il suo albero di copertura minimo T è UNICO

Chiamiamo T' un secondo albero di copertura minimo di G, se ha costo minimo tra tutti gli alberi di copertura minimi eccetto T

② Dimostrate che esistono archi $\{x,y\} \in T$ e $\{x',y'\} \notin T$ tali che $T \setminus \{\{x,y\}\} \cup \{\{x',y'\}\}$ è un secondo albero di copertura minimo

③ Descrivete un algoritmo efficiente per calcolare il secondo migliore albero di connessione minimo di G