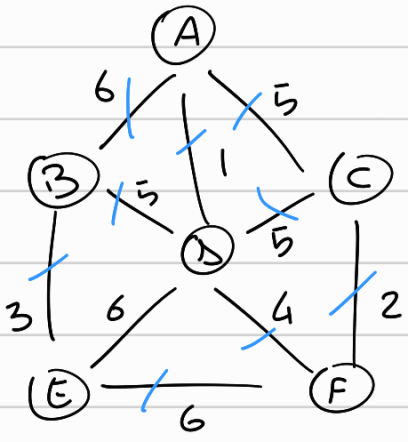
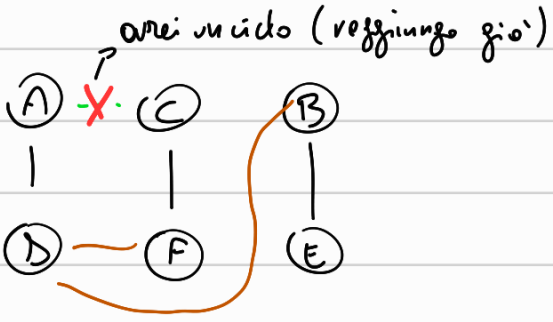
Algoritmo di Kruskal



Si occupa di costruire una foresta che poi si riduce a un albero.
Non ho nodo sorgente. Ho grafi connessi ciclici

Prendo l'arco di costo minimo di volta in volta



procedo in ordine crescente di pesi, quelli di stesso peso è arbitraria la scelta
Una volta che ho tutti i nodi connessi, posso fermarmi

Algoritmo

Kruskal (G)

S = sequenza archi ordinata x costo crescente
T = nodi isolati (no archi) // no parent in questo caso, nullo, archi a caso
while (S non vuoto) *2
*1
(u,v) = S.getFirst() estraggo un arco
if (u e v non connessi in T) // controllo che u e v non siano connessi in T, altrimenti nullo
T = T + (u,v)
*3

return T

Possibile ottimizzazione (ottimizzazione minore) (non cambia la complessità asintotica)

→ mi fermo quando ho messo $n-1$ archi (max archi in un albero di n nodi)

↳ gli altri introdurrebbero cicli

*1 count = 0

*2 count < n-1

*3 count ++

Complessità:

Detti n numero nodi e m archi

Ordinamento con alg ottimo → $\Theta(m \log m)$ (fatti)

Assunte l'oggiunta in T costante, ripeto al max $n-1$ oggiunte

get first al max $\Theta(m)$ se l'arco è in fondo

Verificare se u e v connessi in T:

l'idea è una visita da u dentro T e/o da v per trovare u

→ caso peggiore è un albero vicino a quello finale, dovendo passare per tutti gli archi (costo n^2)

↳ per risolvere il problema si usano strutture dati particolari:

Correttezza

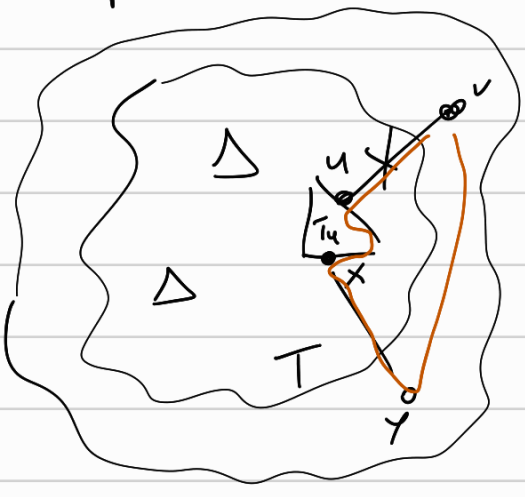
Analoga a Prim

Inv: $T \subseteq$ m.a.r

Vale all'inizio perché $T = \emptyset$

Vale alla fine perché alla fine T ha $n-1$ archi $\rightarrow$ T è un m.a.r.

Si presenta:



- estraggo (u,v) di costo minimo
- se ho ciclo non metto il nodo
  quindi $T \subsetneq$ m.a.r
- se lo metto posso assumere x assurdo
  che non posso più completare m.a.r.

$\rightarrow$ ma T è completabile come m.a.r pur avere un albero quindi ha un altro arco (x,y)

$C_{u,v} \subsetneq C_{x,y}$ (ho sostituito l'arco come per Prim)

$y \notin T_u$, x in T, $y \notin T$ perché altrove starebbe in T_u

Problema - Rendere efficiente TEST connessione
(u e v connessi in T)

Si usano particolari strutture dati delle

Strutture UNION-FIND (cenni)

Modellano strutture dati "collezione di insiemi",
(Per ricordarsi che collezione è la foresta)

con operazioni
- makeSet(e) restituisce singleton con solo e
- find(e) restituisce l'insieme che contiene e
- union(e_1, e_2) fonde l'insieme che contiene e_1 con quello che contiene e_2 (e sceglie unico rappresentante)

Rappresentando insiemi con un elemento "rappresentante":
→ find restituisce il rappresentante dell'insieme che contiene e

Posso rappresentare tali insiemi con alberi n-ari in cui la radice è il rappresentante.

Esempio Universo = $\mathbb{N}$

makeSet(1)
:
makeSet(12)

12 insiemi singleton

union (1,2)

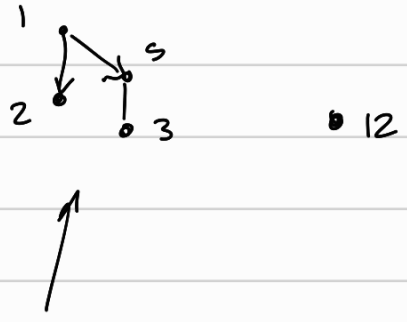
scelgo come rappresentante il 1° argomento

union (5,3)

union (2,3) → fusione

scelgo di fondere su quello a sx (lego le radici)
↳ mantengo il suo rappr.

find (3) fino al riferimento a



Lo scopo di tali strutture è rendere efficiente l'union e la find.

→ per controllare il test di connessione basta effettuare la find su entrambi i nodi e vedere se hanno stessa radice.