12/03/2024

LEZ 3 : MST = MINIMO SPANNING TREE

→ ANCORA ALGO. GREEDY

MST = PROBLEMA DI OTTIMIZZAZIONE

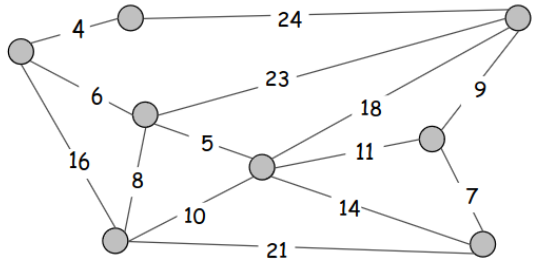
↳ MINIMO ALBERO RICOPRENTE.

DESCRIZIONE PROBLEMA

- INPUT = GRAFO G V NOT ORIENTED, CONNESSO E PESATO.
- OUTPUT = SUBSET DEGLI ARCHI $T \subseteq E$ (E ARCHI DI G) t.c. G SIA ANCORA CONNESSO E LA SOMMA DEI PESI OF ARCHI SIA MINIMO.

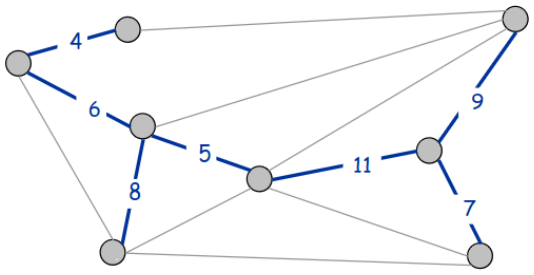
ESEMPIO:

GRAFO G



G = (V, E)

MST



T, $\Sigma_{e \in T} c_e = 50$

FORMALIZZAZIONE PROBLEMA:

- INPUT = GRAFO $G = (V, E)$ ✗ ORIENTED, PESATO E CONNESSO
- SOL. FLESSIBILE = ALBERO RICOPRENTE T (CHE CONTIENE I NODI DI G, TUTTI CONNESSI), t.c. EDGE DI $T \subseteq E$
- MISURA = (DA MINIMIZZ) PESO DI TUTTI GLI ARCHI DI T

APPLICAZIONI DEL PROBLEMA

- COSTRUZIONI URBANE (STRADE, COLL. TUBATURE, ETC.)
- NETWORK DESIGN (COLL. CORRENTE ELETTRICA)
- ALGO APPROSSIMATIVI PER PROBLEMI NP-DIFFICILI.

CREAZIONE ALGORITMO

prima alcune propietà:

- IL MST NON È UNICO IN GENERALE

  MA SE ASSUMIAMO PESI DISTINTI LA SOLUZIONE È UNICA.

  DA DIMOSTRARE → X CASA

ALGORITMI

1. Kruskal's algorithm. Start with $T = \phi$. Consider edges in ascending order of cost. Insert edge e in T unless doing so would create a cycle.
2. Reverse-Delete algorithm. Start with T = E. Consider edges in descending order of cost. Delete edge e from T unless doing so would disconnect T.
3. Prim's algorithm. Start with some root node s and greedily grow a tree T from s outward. At each step, add the cheapest edge e to T that has exactly one endpoint in T.

Remark. All three algorithms produce an MST.

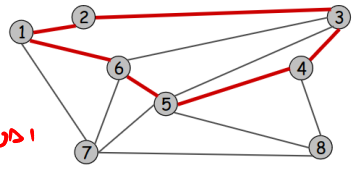
OGGI VEDREMO IN DETTAGLIO KRUSKAL.

ma prima di vedre nel dettaglio l'algoritmo, dobbiamo studiare delle propietà forti per la dimostrazione di correttezza

PROPRIETÀ GRAFO: CICLI E TAGLI

CONSIDERIAMO PRIMA QUESTI CONCETTI:

Cycle. Set of edges the form a-b, b-c, c-d, …, y-z, z-a.



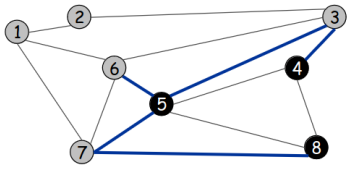
Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1

SET DI NODI

A Cut. A cut is a subset of nodes S. (Sometime defined as a partition of V into S and V\S.)

→ SET DI ARCHI

Cutset. The corresponding cutset D of a cut S is the subset of edges with exactly one endpoint in S.



Cut S = { 4, 5, 8 }
Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8

NODI DI G.

↑

→ "DIVISO" IN 2 SET, $S \subseteq V \setminus S$

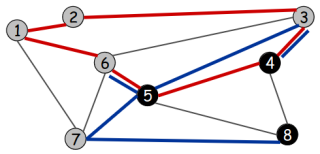
→ $\subseteq E$, t.c. IN OGNI ARCO (ARCHI)

1 NODO ∈ INSIEME CUT

L'ALTRO NODO ∉ //

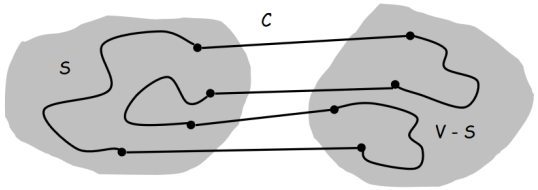
INTERSEZIONE CICLO - TAGLIO → QUASI SEMPRE

Claim. A cycle and a cutset intersect in an even number of edges.

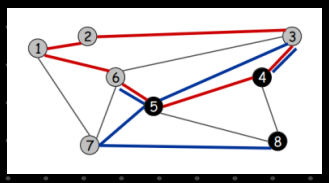


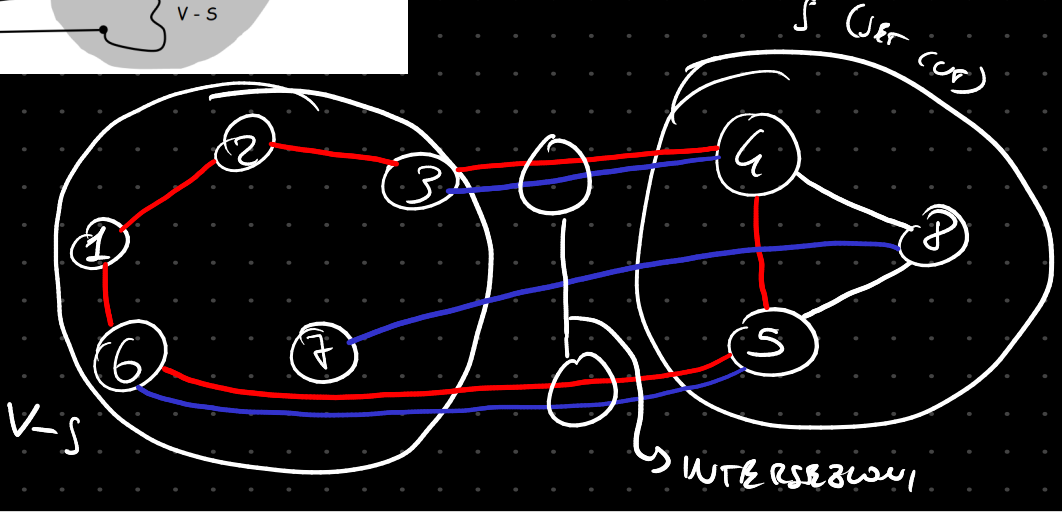
Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
Cutset D = 3-4, 3-5, 5-6, 5-7, 7-8
Intersection = 3-4, 5-6

Pf. (by picture)



$|\{\text{ARCHI DEL CICLO}\}| \cap |\{\text{ARCHI DEL CUT}\}| = \text{n. PARI}$





se il ciclo interseca con il set taglio, vuol dire che i nodi del ciclo in parte saranno in V-S, altri sono in S quindi il ciclo deve passare tra un insieme e l'altro almeno 2 volte o un numero pari di volte, poiche deve necessariamente chiudersi (essendo un ciclo).

A COSA SERVONO QUINDI? × LE NEXT PROPRIETÀ:

SIA G GRAFO

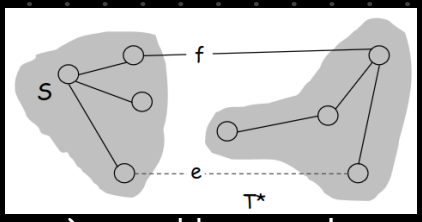
- **CUT PROPERTY**: UN ARCO e $\in$ AD UN CUT E AD UN CUTSET 'PROPRIETÀ DEL TAGLIO' E HA PESO MINIMO TRAI CUTSET, SICURAMENTE $\exists$ UN MST CHE CONTIENE e.

- **CYCLE PROPERTY**: L'ARCO DI PESO MAX CONTENUTO IN UN CICLO PROBABILMENTE $\notin$ AD UN MST DEL GRAFO 'PROPRIETÀ DEL CICLO'

DIM CUT PROPERTY

- SIA S CUT QUALSIASI, E e ARCO MINIMO CONTENUTO NEL CUTSET.

- SUPPONIAMO CHE $e \notin T^*$ (M.S.T.) :



  - SE ADD e, CREO UN CICLO IN T.
    supponendo che (e) sia stato 'scartato' perchè sennò avrebbe creato un ciclo (cosa che non deve avere un MST).
  - X PROPRIETÀ CUT-CYCLE INTERSECT, $\exists$ UN ALTRO ARCO F CHE $\in$ AL CUTSET
  - SE CONSIDERIAMO $T' = T^* \cup \{e\} - \{F\}$ NOTIAMO CHE ANCHE LUI È UN M.S.T.
  - E SAPENDO CHE $C_e \leq C_f \Rightarrow COST(T') \leq COST(T^*)$
    ↳ COSTO

T' QUINDI È UN MST CONTENENTE e

□

DIM CYCLE PROPERTY

- SIA C CICLO IN G e f ARCO CON PESO MAX IN C ALLORA $\exists$ MST CHE $\not\ni$ CONTAIN f.

- SUPPONIAMO CHE MST T^* CONTIENE f.

  - SE TOGLIAMO f, CREO 1 CUT S e INSIEME $V = G - S$
  - X CYCLE PROPERTY, $\exists$ UN ARCO e CHE $\in A$ S e A V.
  - SE ADD e, T' RISULTANTE È SEMPRE M.S.T.
  - SICCOME $c_f \geq c_e$, ALLORA $COST(T^*) \geq COST(T')$

  QUINDI, $\exists$ MST (T') CHE NON CONTIENE F.

ORA VEDIAMO L'ALGORITMO

DESCRIZIONE: START DA G e T (M.S.T.) VUOTO. ORDINO GLI ARCHI DI G IN ORDINE CRESCENTE E VISITO UN ARCO X VOLTA E LI ADD TO T, EVITANDO GLI ARCHI CHE FORMINO CICLI.

IMPLEMENTAZIONE: X ESSERE FAST OCCORRE USARE LA S.D. UNION-FIND. PER:

- MANTENERE LE COMPONENTI CONNESSE DELLA SOLUZIONE
- X VERIFY USIAMO L'OP FIND() X TROVARE E CHE E IN STESSI COMP. CONNESSE, (ARCO DA SCARTARE QUINDI).

PSEUDOCODICE

```
algorithm Kruskal (graph G=(V,E,c))
    UnionFind UF
    T=∅
    sort the edges in ascending order of costs
    for each vertex v do UF.makeset(v)
    for each edge (x,y) in order do
        T_x=UF.find(x)
        T_y=UF.find(y)
        if T_x≠ T_y then
            UF.union(T_x,T_y)
            add edge (x,y) to T
    return T
```

∈ STESSO GRUPPO, ADIACENTI QUINDI.

] -> SE UGUALI FORMANO UN CICLO QUINDI

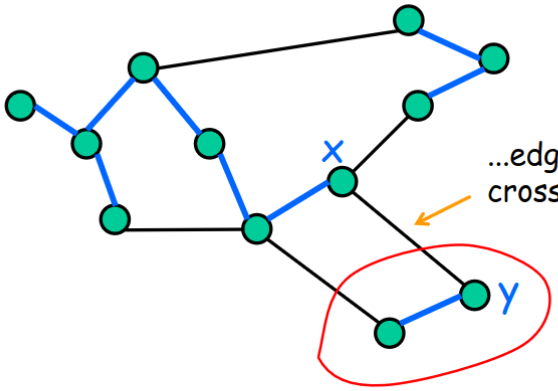
-> ADD ARCO.

CORRETTEZZA ^ COMPLEXITY

CORRETTO? SI'

CONSIDERANDO L'IMMAGINE SOTTO, DOVE L'ALGO DECIDE DI AGGIUNGERE (x,y) NELL'ALBERO.

when the algorithm decides to add edge (x,y) to the solution



since the algorithm look at the edges in increasing order of costs...

...edge of minimum cost crossing the cut S,V\S

consider the set S of vertices belonging to the same connected component of y

COMPLESSITÀ:

```
algorithm Kruskal (graph G=(V,E,c))
    UnionFind UF
    T=∅
    sort the edges in ascending order of costs
    for each vertex v do UF.makeset(v)
    for each edge (x,y) in order do
            T_x=UF.find(x)
            T_y=UF.find(y)
            if T_x≠ T_y then
              UF.union(T_x,T_y)
              add edge (x,y) to T
    return T
```

① ② ③ ④

m al massimo è n^2

ARCHI

① ORDINO GLI ARCHI → $O(m \log m)$ ma $m = O(n^2)$ =

$= O(m \log n)$

② n MAKESET = $O(n)$ → APPROSSIMABILE

③ m FIND

④ n-1 UNION

→ COSTO DIPENDE DA CHE S.D. UTILIZZIAMO.

IN DEFINITIVA

$O(m \log n)$

-using QuickFind with *union by size*
$O(m \log n + m + n \log n) = O(m \log n)$
-using QuickUnion with *union by size*
$O(m \log n + m \log n + n) = O(m \log n)$