.UBAfiuba
FACULTAD DE INGENIERÍA

Árbol recubridor mínimo

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Problema

Sea

G=(V,E) un grafo conexo y ponderado (con costos “w” positivos)

Queremos

Seleccionar un subconjunto de ejes $T \subseteq E$

De forma tal

Que el nuevo grafo G’=(V,T) sea conexo

Y el costo total $W = \sum_{e \in T} w_e$

sea mínimo entre todos los posibles grafos conexos

El resultado es un árbol

Un árbol

es un grafo simple no dirigido conexo y sin ciclos

El grafo G'=(V,T)

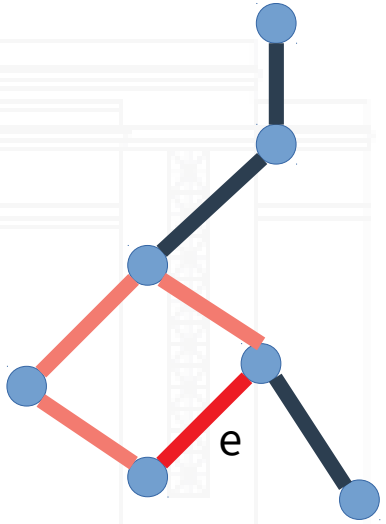
Por definición es conexo

Si G' tiene un ciclo C

Sea un eje e ∈ C,

Si extraemos e del grafo G', el grafo resultante aun sera conexo

Y tendrá un costo menor.



Por lo tango G' no tiene ciclos

Árbol recubridor mínimo

Se conoce el problema con el nombre

Árbol recubridor mínimo

Para un grafo G=(V,E)

existen varios posibles arboles rucubridores

Solo

un subconjunto de ellos (o 1) es mínimo

Propiedad de corte (cut property)

Supongamos por un instante

Que todos los costos de los ejes son diferentes

Para cualquier

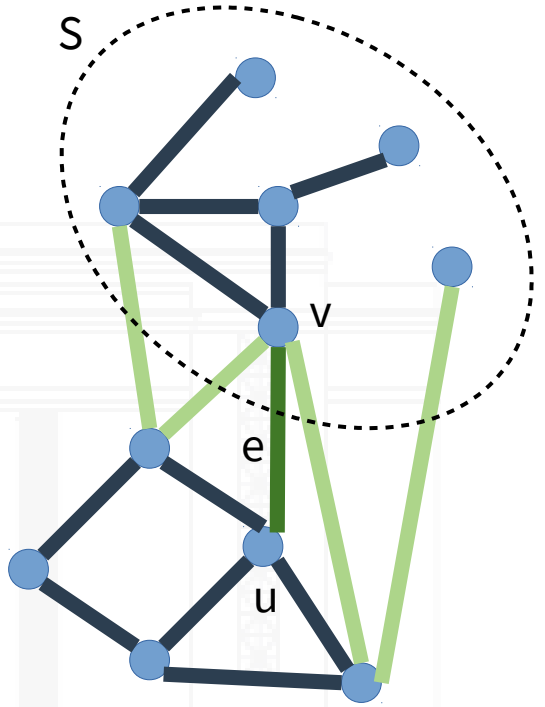
subset de nodos $S \subset V$ (con $S \neq V$ y $S \neq \emptyset$)

Existe un subconjunto de ejes F

Tal que para todo $f=(u,v) \in F$, $u \in S$ y $v \in V-S$

Existe un eje e = (u,v) ∈ F

Que tiene el costo menor que el resto de los demás en F



Propiedad de corte (cont.)

Sea un árbol recubridor T que no contiene a e ∈ F entre sus ejes

Pero que contiene e'=(u',v') ∈ F siendo "puente" entre S y V-S

Si intercambiamos e' por e

conformando un nuevo grafo sin ciclos y conexo (un árbol) *

Donde ahora e será "puente" entre S y V-S

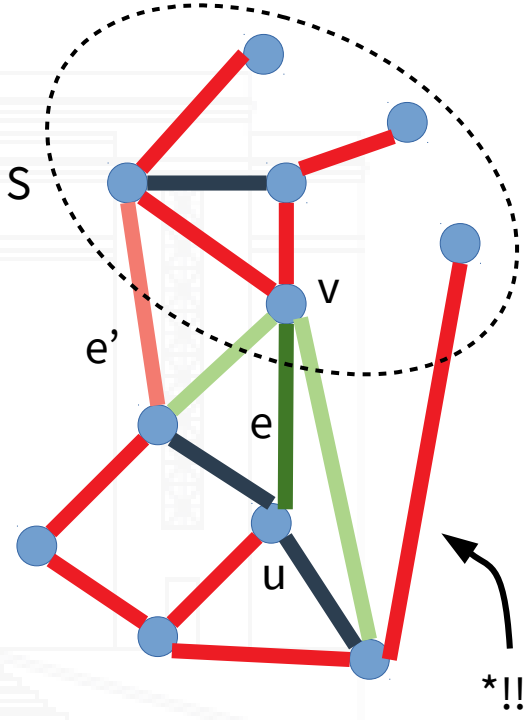
Sabemos que $W_e < W_{e'}$

Por lo que el costo del árbol recubridor resultante T' será menor.

En conclusión

Cualquier árbol recubridor mínimo de G deberá tener a e como puente entre S y V-S

(para cualquier S que tenga a e en la frontera)



Algoritmos tipo greedy

Para encontrar un árbol recubridor mínimo

- Existen varios algoritmos greedy (todos ellos óptimos)

Algunos de ellos

- Algoritmo Kruskal
- Algoritmo Prim
- Algoritmo de eliminación inversa (Reverse-delete)
- ...

Todos ellos son iterativos

- Funcionan agregando/quitando un eje a la vez
- Basándose en la propiedad de corte

Algoritmo Kruskal

Algoritmo propuesto por Joseph Kruskal en 1956

En la publicación “On the shortest spanning subtree and the traveling salesman problem”

Inicialmente

Todos los nodos de G=(V,E) conforman arboles en un bosque

Iterativamente en orden creciente de costo

recorre los ejes de E

En una iteración, analizando el eje e=(u,v)

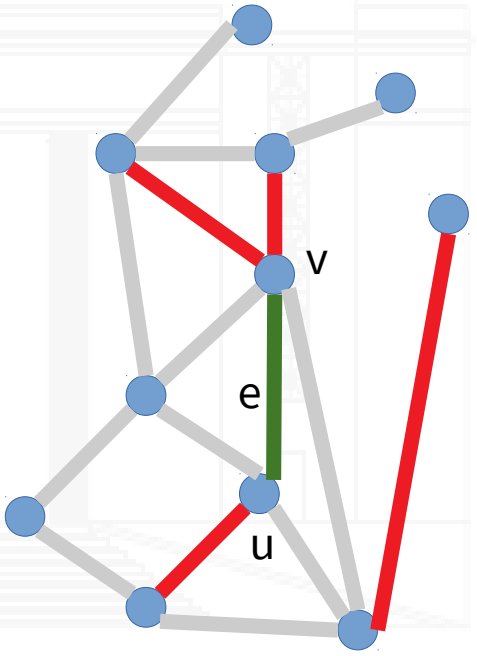
Evalúa unir los arboles que contienen a u y v en uno solo.

Desecha la acción Si el resultado de esa operación

se realiza en nodos u y v que ya pertenecen al mismo árbol

(agregarlo generaría un ciclo)

El resultado es un árbol recubridor mínimo



Kruskal - Optimalidad

Queremos ver que

Cada vez que el algoritmo agrega un eje, lo hace de a cuerdo a la propiedad de corte

Sea

e=(v,w) eje agregado en una iteración

S el subconjunto de nodos a los cuales v tiene un camino antes de agregar el eje e

Sabemos que

$v \in V$ y que $w \notin V$ (sino se crearía un ciclo al agregar el eje e)

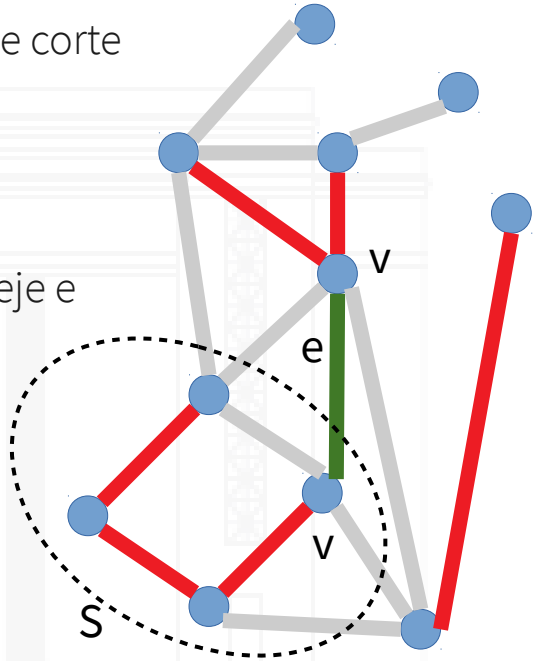
No hay ejes seleccionados que vayan de S a V-S

Por lo tanto

Por como se selecciona e, corresponde al eje menos costoso que une S con V-S

En conclusión

El eje e pertenece a cualquier árbol recubridor mínimo de G (según propiedad corte)



Algoritmo Prim

Algoritmo fue diseñado en forma independiente por

Vojtech Jarnik (1930), Robert C. Prim (1957) y Dijkstra (1959)

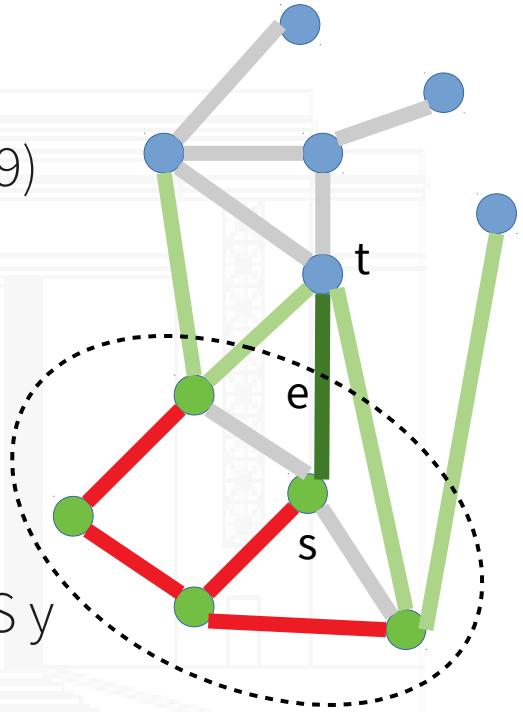
Inicialmente

Se selecciona un u nodo del grafo y S={u}

Repetimos mientras V-S ≠ ∅

Seleccionamos el eje e=(s,t) con el menor costo y con s ∈ S y t ∈ V-S

Agregamos t a S (S=S+{t})



Prim - Optimalidad

Queremos ver que

Cada vez que el algoritmo agrega un eje, lo hace de a cuerdo a la propiedad de corte

Sea

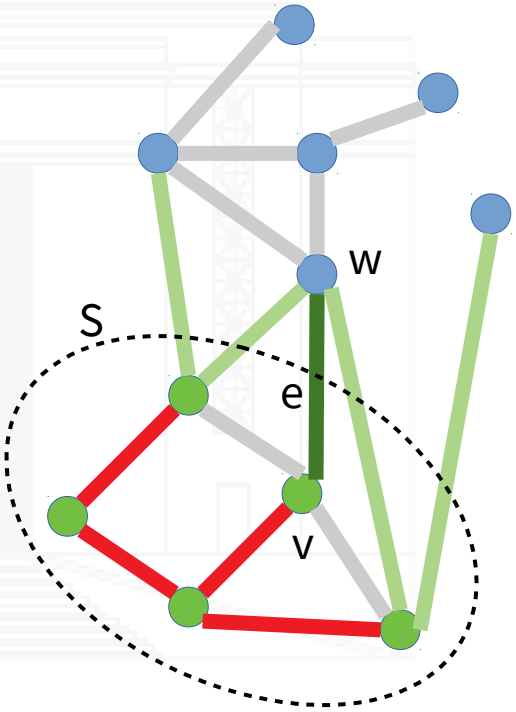
e=(v,w) eje agregado en una iteración

S por funcionamiento del algoritmo es un árbol y $v \in S$

V-S son todos nodos sueltos que aun no se unen al arbol S

Según algoritmo $w \in V-S$

El eje e es aquel de menor costo que va de V a V-S



En conclusión

El eje e pertenece a cualquier árbol recubridor mínimo de G (según propiedad corte)

Algoritmo de eliminación inversa

Inicialmente

Comenzamos con el grafo completo

Iterativamente en orden decreciente de costo

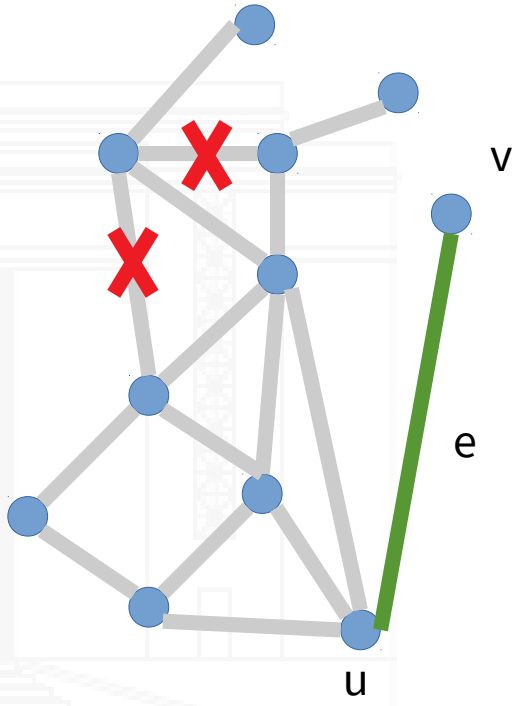
recorre los ejes de E

En una iteración, analizando el eje e=(u,v)

Eliminamos el eje si al realizarlo el grafo resultante continua siendo conexo

Sino lo mantenemos

El resultado es un árbol recubridor mínimo



Propiedad de ciclo

Supongamos por un instante

Que todos los costos de los ejes son diferentes

Sea

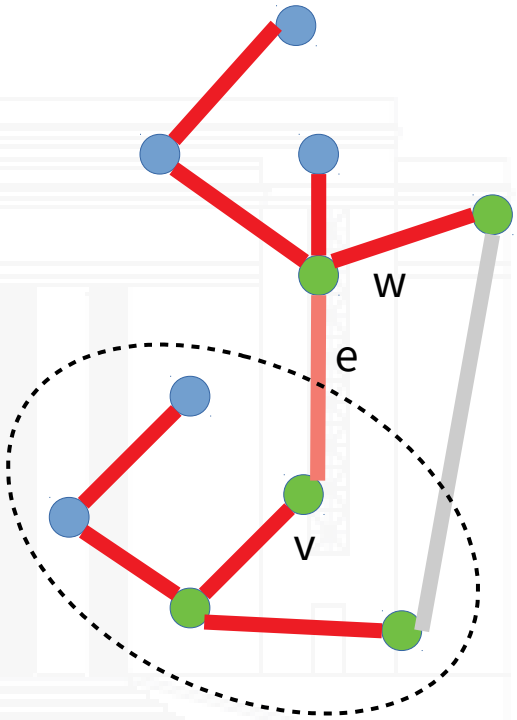
C un ciclo en G

e=(v,w) el eje mas costoso dentro de C

T un árbol recubridor que contiene a e

Si eliminamos e de T

Nos quedarán 2 componentes conexos S y V-S



Propiedad de ciclo (cont)

Para volver a generar el árbol

debemos seleccionar un eje e'=(v',w') con v' ∈ S y w' ∈ V-S

Como e pertenece al ciclo C,

existe en el gráfo otro eje con menor costo que construye un camino entre v y w

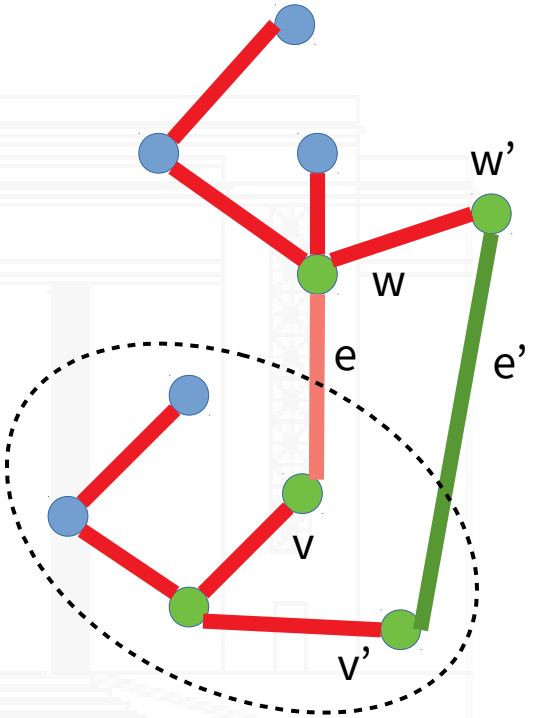
Ese eje debe ser e'!

Al agregar e'

Se genera un nuevo árbol recubridor T' de menor costo que T

Por lo tanto

Si existe un ciclo C en el Grafo G, el eje de mayor costo en C no pertenece al árbol recubridor mínimo



Eliminación inversa - Optimalidad

Sea

El eje e=(v,w) eliminado por el algoritmo en una iteración

Como

Es el eje de mayor costo que aun no se ha removido (obviando a aquellos que quitarlo generarán una desconexión del grafo)

Y al quitar el eje el algoritmo se asegura que el resultante siga siendo conexo

Entonces

El eje e pertenece a un ciclo inmediatamente antes de su remoción.

Por lo tanto

El algoritmo elimina aquellos ejes que no pertenecen al árbol recubridor mínimo (por propiedad de ciclo)

¿Qué pasa si tengo ejes con costos iguales?

Si

Varios ejes pueden compartir el mismo costo

Seleccionar

Cualquiera de ellos nos lleva al árbol recubridor mínimo

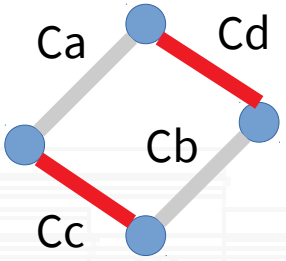
De hecho

Esto hace que puedan existir varios árboles recubridores mínimos

Necesitamos

poder desempatar entre entre los ejes de igual valor para ver cual quitar

(puede ser simplemente por orden en el que vienen)



Cc < Cd < Ca = Cb

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Septiembre de 2020