Árboles generadores mínimos (MST)

comp-420

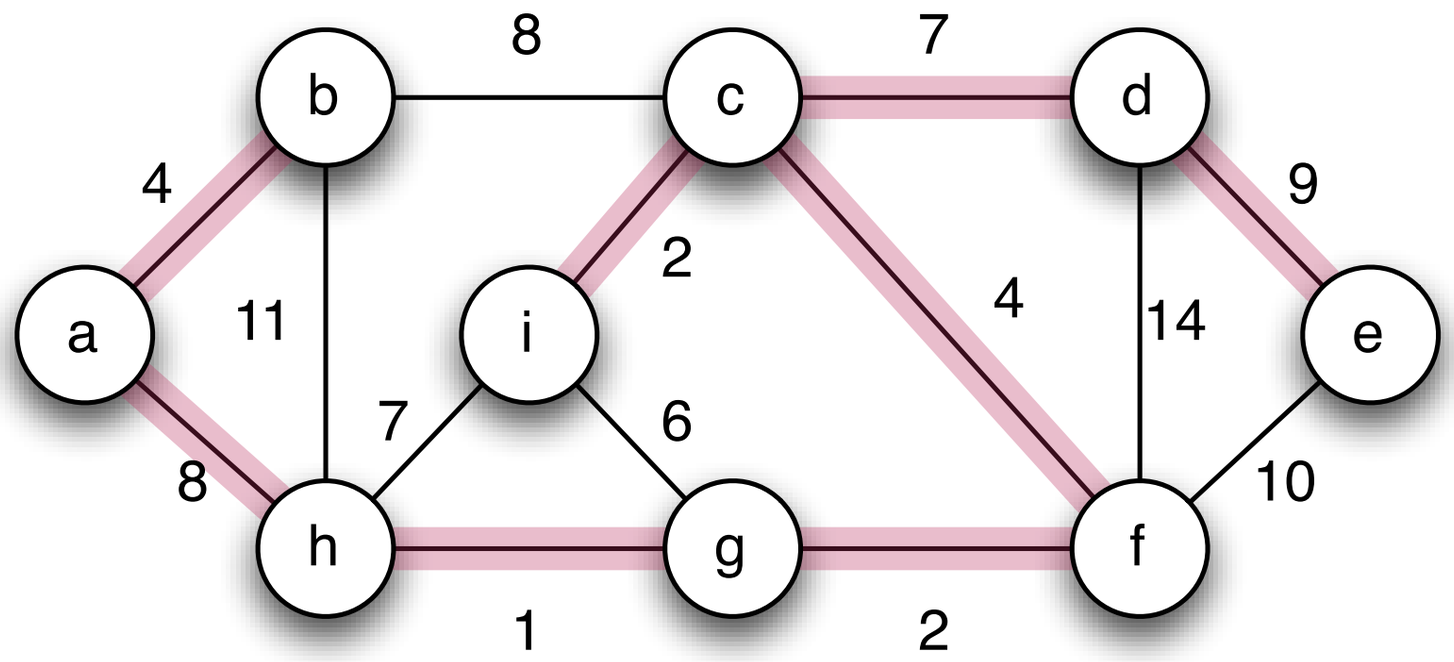
Árboles generadores mínimos (MST)

- Sea $G=(V,E)$ una gráfica conectada y no-dirigida con V vértices y E aristas.
- Cada arista $(u,v)\in E$ tiene un peso $w(u,v)$, que es el costo de conectar u y v.
- La meta es encontrar un subconjunto acíclico $T\subseteq E$ que conecte todos los vértices y que minimice el costo total

$$w(T) = \sum_{(u,v)\in T} w(u,v).$$

- ¿Qué tipo de estructura debe ser T para ser acíclico y conectar todos los nodos?
  - un árbol (llamado árbol generador)
- En inglés "minimum spanning trees".

Árboles generadores mínimos (MST)

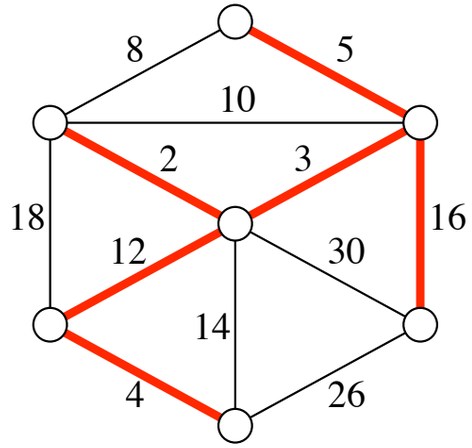


- árbol generador mínimo con un costo de 37.
- árbol que no es único. Podemos cambiar por ejemplo (b,c) por (a,h) y tendrá el mismo costo.

Árboles generadores mínimos (MST)

- Por simplicidad supondremos que todas las aristas tienen pesos distintos w(e) ≠ w(e') para cualquier par de aristas e y e'.

- Pesos distintos garantizan que el MST de una gráfica sea única.



- Podemos utilizar un algoritmo que suponga pesos iguales en ciertas aristas en pero con un método de desempate.

- Un ejemplo de método de desempate es SHORTEREDGE(i,j,k,l) que decide, entre dos aristas (i,j) y (k,l) cuál tiene el menor peso.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

Árboles generadores mínimos (MST)

```
ShorterEdge(i, j, k, l)
    if w(i, j) < w(k, l) return (i, j)
    if w(i, j) > w(k, l) return (k, l)
    if min(i, j) < min(k, l) return (i, j)
    if min(i, j) > min(k, l) return (k, l)
    if max(i, j) < max(k, l) return (i, j)
    ⟪if max(i,j) < max(k,l)⟫ return (k, l)
```

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

Árboles generadores mínimos (MST)

- ¿Cómo encontrarían el árbol generador mínimo?
- IDEA: algoritmo de fuerza bruta.
  - calcular todos los árboles generadores.
  - encontrar el mínimo.
- PROBLEMAS:
  - no es fácil de implementar.
  - pueden ser muchísimos árboles generadores.

Árboles generadores mínimos (MST)

- Existen varias maneras de calcular el MST pero todas son instancias de un algoritmos genérico.
- La situación es similar a la de los algoritmos de recorridos de gráficas (dfs y bfs).
- El algoritmo genérico para encontrar el MST mantiene una subgráfica acíclica F de la gráfica de entrada G.
- Esta subgráfica se llama bósque generador intermedio (intermediate spanning forest)
- F es una subgráfica del árbol generador mínimo de G.
- Cada componente de F es un árbol generador mínimo de sus vértices.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

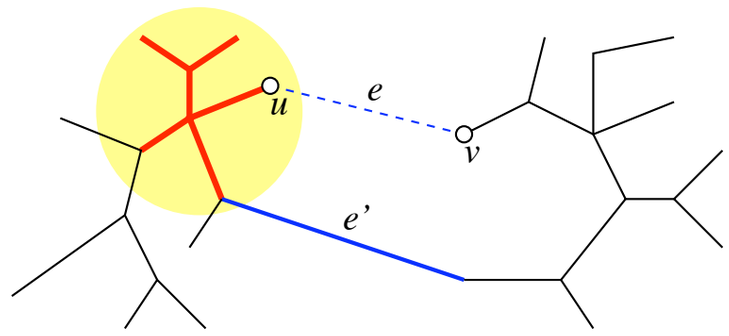
Árboles generadores mínimos (MST)

- Inicialmente, la subgráfica F consta de n árboles de un nodo cada uno.
- El algoritmo genérico une los árboles agregando ciertas aristas entre esos nodos:
  - USELESS edges.
  - SAFE edges.
  - UNDECIDED edges
- Cuando el algoritmo se detiene, F consta de un sólo arbol de n vértices: el árbol generador mínimo.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

Árboles generadores mínimos (MST)

- El MST contiene cada arista SEGURA.

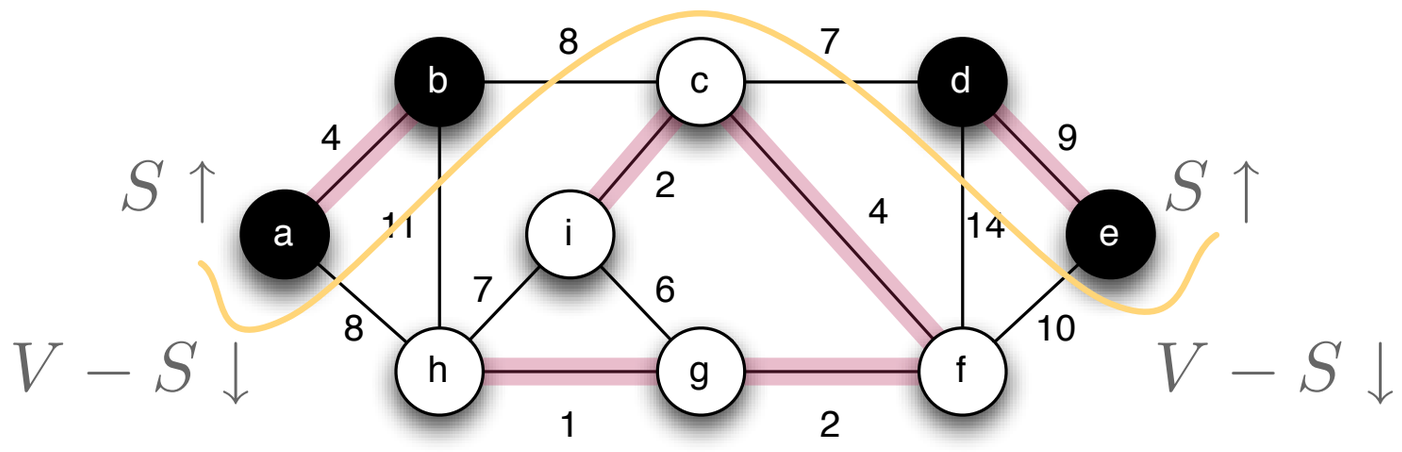


- El MST no tiene aristas INÚTILES.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

MST: algoritmo genérico

- Problema: identificar aristas seguras (que respeten la invariante)
- Un corte (cut) (S, V-S) de una gráfica no dirigida es una partición de V.



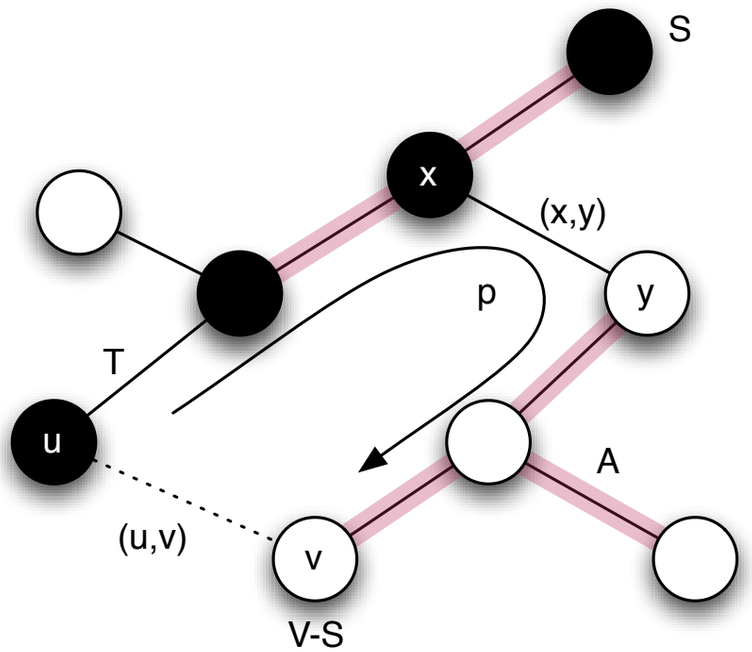
- Decimos que la arista (u,v) ∈ E cruza el corte (S, V-S) si uno de sus extremos está en S y el otro en V-S.
- ¿Qué arista cruza el corte aquí?

MST: algoritmo genérico

- Decimos que un corte respeta un conjunto A de aristas si ninguna arista de A cruza el corte.
- Una arista que cruza un corte se llama arista clara (light edge) si su costo es el mínimo entre cualquier otra arista que cruza el corte.
- Puede haber más de una arista clara en el caso que el peso sea el mismo entre estas aristas.
- La regla para reconocer aristas seguras es entonces:

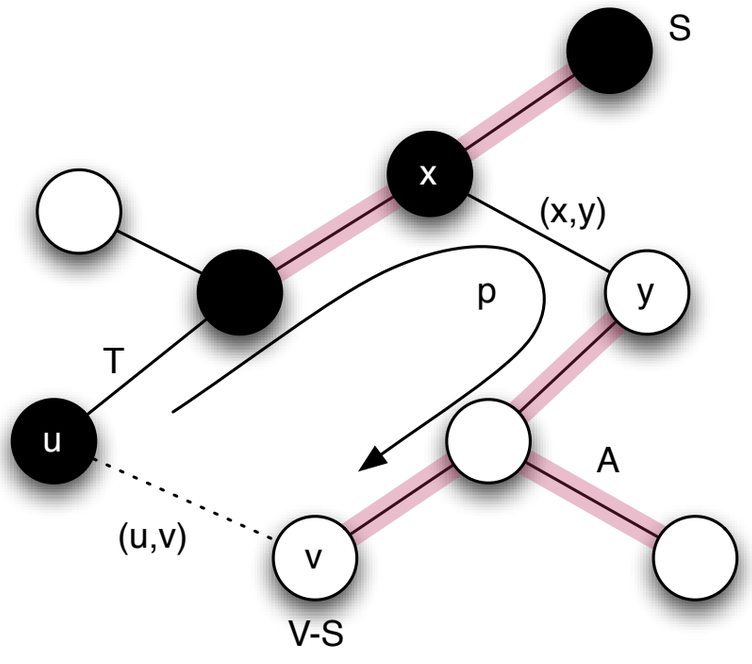
Sea G=(V,E) una gráfica conectada, no dirigida con una función real w de costos definida en E. Sea A un subconjunto de E incluido en un MST para G, sea (S,V-S) cualquier corte de G que respete A, y sea (u,v) una arista clara que cruce (S,V-S). Entonces, la arista (u,v) es segura para A.

MST: aristas seguras



- Sea T un MST que incluye A y supongamos que T no contiene a la arista clara (u,v).
- Queremos construir otro MST, T' que incluya A ∪ {(u,v)} mostrando que (u,v) es una arista segura.
- La arista (u,v) forma un cíclo con las aristas del camino p desde u hasta v en T.
- Ya que u y v están en lados opuestos del corte (S,V-S), hay al menos una arista en T en el camino p que también cruza el corte.
- (x,y) no está en A porque sabemos que el corte respeta A.

MST: aristas seguras



- Ya que (x,y) está en el único camino p desde u hasta v en T, remover (x,y) desconecta T en dos componentes.

- Agregar (u,v) le reconecta para formar un nuevo MST:

  - $T' = T - \{(x,y)\} \cup \{(u,v)\}$.

MST: aristas seguras

- Probamos que T′ es un MST:
- Ya que (u,v) es una arista clara que cruza (S, V–S) y (x,y) también cruza el corte, w(u,v) ≤ w(x,y).
  - Entonces:

    w(T′) = w(T) – w(x,y) + w(u,v)

    ≤ w(T)

- Como T es un MST, w(T′) ≤ w(T), entonces T′ debe ser también un MST.
- Ya que T′ es un MST, (u,v) es una arista segura para A.

MST: algoritmo genérico

- Mientras el algoritmo avanza, el conjunto A es siempre acíclico, de otra forma el MST que incluye A contendría un cíclo, lo que es contradictorio.
- En cualquier momento de la ejecución del algoritmo, la gráfica $G_A = (V,A)$ es un bosque, en donde cada componente de G_A es un árbol.
- Algunos de estos árboles pueden contener un solo vértice (al inicio del algoritmo por ejemplo).
- Cualquier arista segura conecta dos componentes de G_A.
- El cíclo de las líneas 2 a 4 de GENERIC-MST se ejecuta, ¿cuántas veces?
  - V-1 veces.
- ¿Cuándo termina el algoritmo?
  - cuando el bosque contiene sólo un árbol.

MST: Algoritmo de Borůvka

- Ideado por Otakar Borůvka (1899-1995) en 1926.
- En su artículo "On a certain minimal problem" describió un algoritmo para encontrar el MST de una red eléctrica (1926).
  - ¿Qué buscaría Borůvka en esta aplicación?
    - encontrar la construcción más económica de la red eléctrica.
- Problema concreto de ingeniería que se convirtió en un modelo de problema para la optimización combinatoria.
- Es el algoritmo más antiguo y posiblemente el más simple para encontrar MST.
- Redescubierto por Choquet, Florek, Łukaziewicz, Perkal, Steinhaus, Zubrzycki y Sollin.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

MST: Algoritmo de Borůvka / Choquet / Florek / Łukaziewicz / Perkal / Stienhaus / Zubrzycki / Sollin

Allie Brosh, "This is Why I'll Never be an Adult", Hyperbole and a Hlaf. June 17, 2010.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

¿Cómo encontrar todas las aristas seguras en una gráfica en tiempo O(E)?

- contar las componentes de F usando cualquier whatever-first-search.
- mientras contamos etiquetamos cada vértice con su número de componente.
- Si F solo tiene un componente, terminamos.
- Si no, calculamos un arreglo S[1...V] de aristas, donde S[i] es la arista con menor peso con un endpoint en el componente i (o el valor sentinela NULL si hay menos de i componentes).
  - Para esto consideramos cada arista uv en la gráfica de entrada G.
  - Si los endpoints u y v tienen la misma etiqueta, uv es inútil.
  - Si no, comparar el peso de un a los de S[label[u]] y S[label[v]], actualizando su valor si es necesario.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

MST: Algoritmo de Borůvka

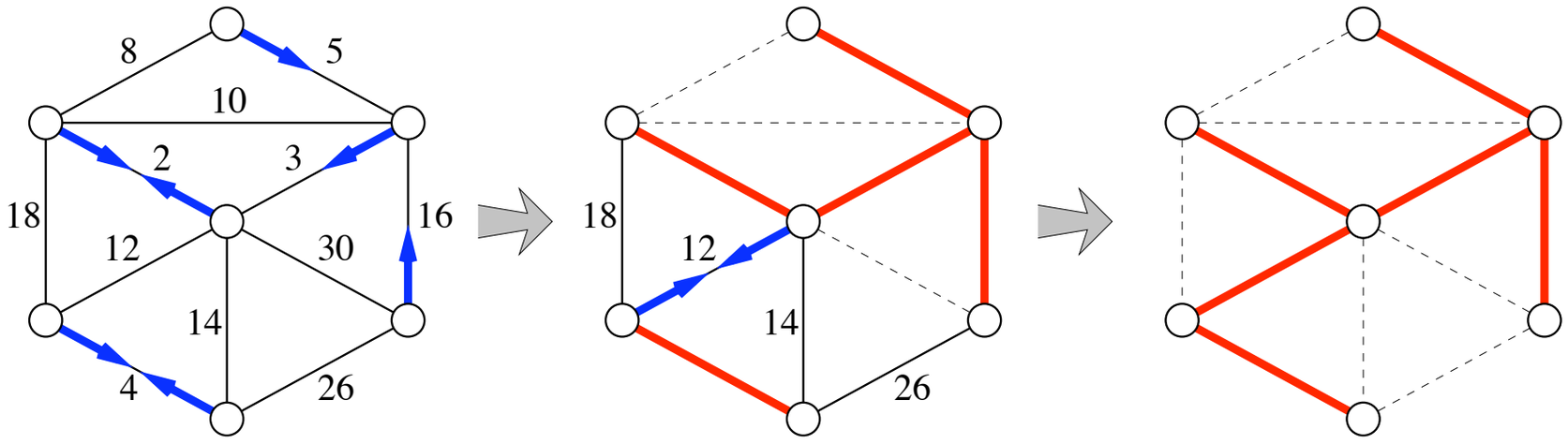
```
BORŮVKA(V, E):
  F = (V, ∅)
  count ← COUNTANDLABEL(F)
  while count > 1
      ADDALLSAFEEDGES(E, F, count)
      count ← COUNTANDLABEL(F)
  return F
```

```
ADDALLSAFEEDGES(E, F, count):
  for i ← 1 to count
      S[i] ← NULL        ⟨⟨sentinel: w(NULL) := ∞⟩⟩
  for each edge uv ∈ E
      if label(u) ≠ label(v)
          if w(uv) < w(S[label(u)])
              S[label(u)] ← uv
          if w(uv) < w(S[label(v)])
              S[label(v)] ← uv
  for i ← 1 to count
      if S[i] ≠ NULL
          add S[i] to F
```

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

MST: Algoritmo de Borůvka

BORŮVKA: Add all the safe edges and recurse.



- El algoritmo elige arbitrariamente un vértice líder.
- La manera más simple para encontrar a este líder es con una búsqueda en profundidad (DFS) en A. El primer nodo que visitamos en cualquier componente será el líder del componente.
- Una vez elegidos los líderes encontramos las aristas seguras para cada componente, por fuerza bruta y estas son agregadas a A.

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf

MST: Algoritmo de Borůvka

- Cada llamada a CountAndLabel toma ...
  - O(V) porque el bosque F tiene a lo más V-1 aristas.
- Suponiendo que la gráfica se representa con listas de adyacencia, el resto de cada iteración del main while loop toma
  - O(E) tiempo porque pasamos tiempo constante en cada arista.
- Como la gráfica está conectada, tenemos $V \leq E+1$, así que cada iteración del ciclo while toma O(E).
- Cada iteración reduce el número de componente de F por al menos un factor de dos.
  - El peor caso ocurre cuando los componentes se forman en pares.
  - Ya que F tiene inicialmente V componentes, el cíclo while itera a lo más O(logV) veces.
  - El tiempo total de ejecución del algoritmo es
    - O(ElogV).

http://web.engr.illinois.edu/~jeffe/teaching/algorithms/notes/20-mst.pdf