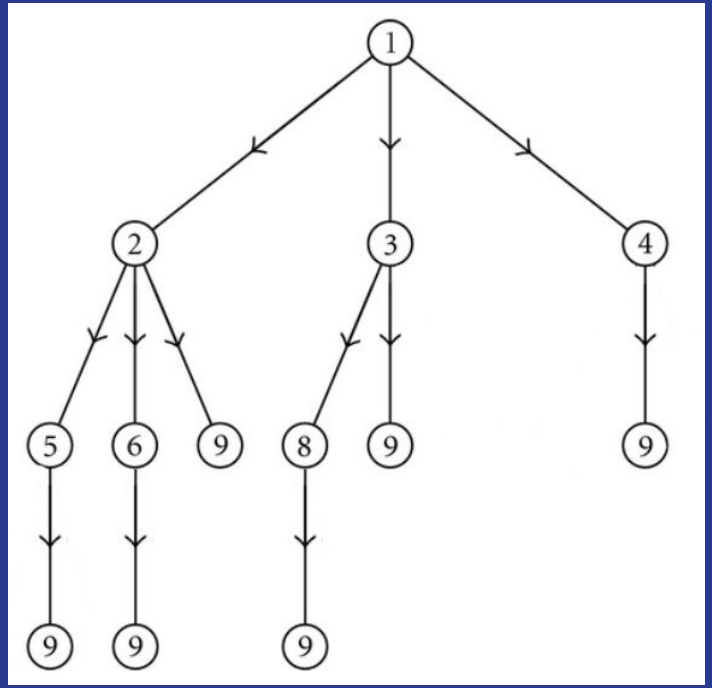
Minimum Cost Arborescence

Optimum branching (Tarjan)
Directed minimum spanning tree
Algoritmo de Chu-Liu/Edmonds



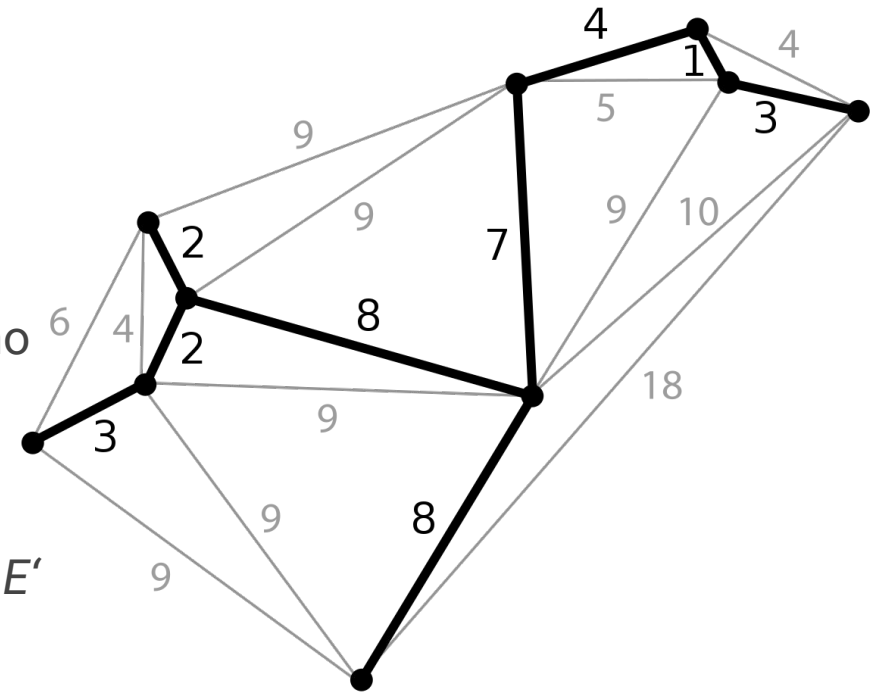
MST Problem

Sea *G = (V , E)* un grafo no dirigido compuesto por *N* vértices y *M* aristas.

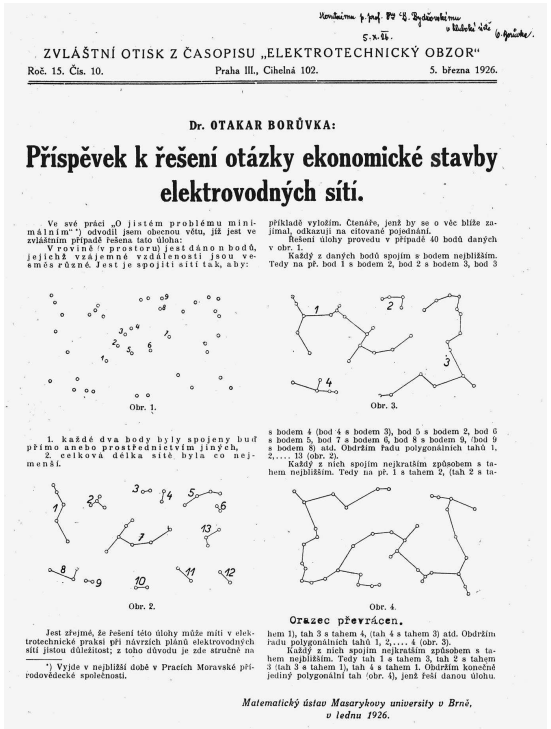
Por cada arista e ∈ E , definimos w(e) como el peso de la arista.

Hallar un árbol de expansión *T = (V , E')* donde la sumatoria de *w(e)* para todo *e ∈ E'* es mínima.

Un árbol de expansión de *G* es un subconjunto de *N - 1* aristas que forman un árbol en los *N* vértices



Un poco de historia del MST problem



ZVLÁŠTNÍ OTISK Z ČASOPISU „ELEKTROTECHNICKÝ OBZOR"

Roč. 15. Čís. 10. Praha III., Cihelná 102. 5. března 1926.

Dr. OTAKAR BORŮVKA:

Příspěvek k řešení otázky ekonomické stavby elektrovodných sítí.

Ve své práci „O jistém problému minimálním"[*]) odvodil jsem obecnou větu, jíž jest ve zvláštním případě řešena tato úloha:

V rovině v prostoru) jest dáno n bodů, jejichž vzájemné vzdálenosti jsou všechny různé. Jest je spojiti sítí tak, aby:

1. každé dva body byly spojeny buď přímo anebo prostřednictvím jiných,
2. celková délka sítě byla co nejmenší.

Obr. 1.

Obr. 2.

Jest zřejmé, že řešení této úlohy může míti v elektrotechnické praksi při návrzích plánů elektrovodných sítí jistou důležitost; z toho důvodu je zde stručně na příkladě vyložím. Čtenáře, jenž by se o věc blíže zajímal, odkazuji na citované pojednání.

Řešení úlohy provedu v případě 40 bodů daných v obr. 1.

Každý z daných bodů spojím s bodem nejbližším. Tedy na př. bod 1 s bodem 2, bod 2 s bodem 3, bod 3 s bodem 4 (bod 4 s bodem 3), bod 5 s bodem 2, bod 6 s bodem 5, bod 7 s bodem 6, bod 8 s bodem 9, bod 9 s bodem 8) atd. Obdržím řadu polygonálních tahů 1, 2, ..., 13 (obr. 2).

Obr. 3.

Každý z nich spojím nejkratším způsobem s tahem nejbližším. Tedy na př. 1 s tahem 2, (tah 2 s tahem 1), tah 3 s tahem 4, (tah 4 s tahem 3) atd. Obdržím řadu polygonálních tahů 1, 2, ..., 4 (obr. 3).

Obr. 4.

Orazec převrácen.

Každý z nich spojím nejkratším způsobem s tahem nejbližším. Tedy tah 1 s tahem 3, tah 2 s tahem 3 (tah 3 s tahem 1), tah 4 s tahem 1. Obdržím konečně jediný polygonální tah (obr. 4), jenž řeší danou úlohu.

[*]) Vyjde v nejbližší době v Pracích Moravské přírodovědecké společnosti.

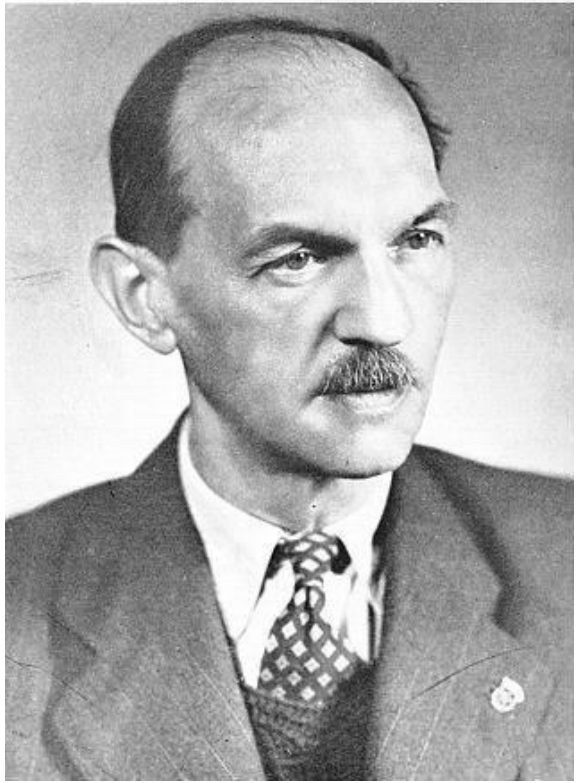
Matematický ústav Masarykovy university v Brně, v lednu 1926.

1. **Boruvka (1926):**

"A CONTRIBUTION TO THE SOLUTION OF A PROBLEM OF ECONOMIC CONSTRUCTION OF ELECTRIC POWER-LINE NETWORKS"

Boruvka Algorithm

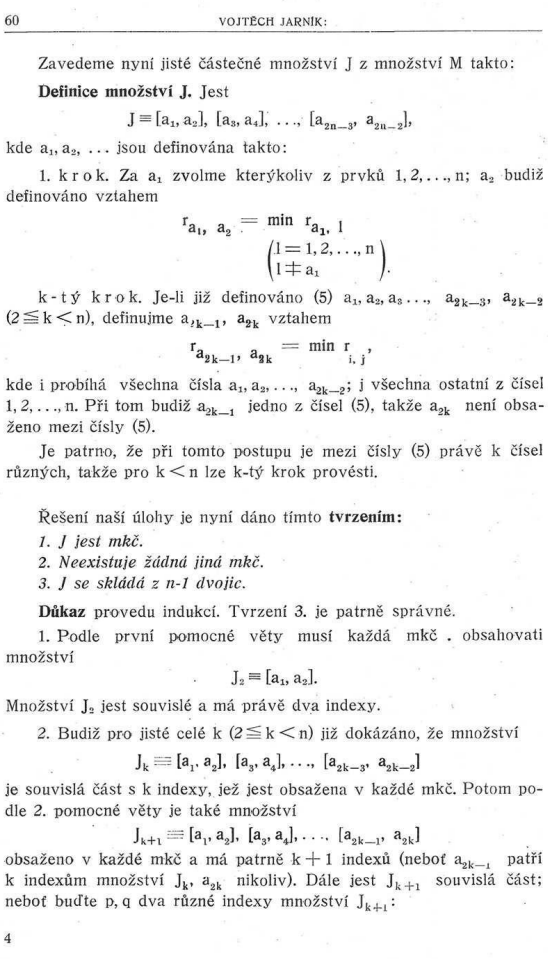
- For each vertex, select the minimum-weight edge incident to the vertex.
- Replace by a single vertex each connected component defined by the selected edges.
- Delete all resulting isolated vertices, loops, and all but the lowest-weight edge among each set of multiple edges.



2. Vojtech Jarnik (1930)

“O jistém problem minimálním”

Así se titula el artículo del matemático checo *Vojtěch Jarník* que escribe como respuesta al paper de *Boruvka*. En ella establece otra solución novedosa al problema del MST



60 VOJTĚCH JARNÍK:

Zavedeme nyní jisté částečné množství J z množství M takto:

Definice množství J. Jest

$$J \equiv [a_1, a_2], [a_3, a_4], \ldots, [a_{2n-3}, a_{2n-2}],$$

kde $a_1, a_2, \ldots$ jsou definována takto:

1. k r o k. Za a_1 zvolme kterýkoliv z prvků $1, 2, \ldots, n$; a_2 budiž definováno vztahem

$$r_{a_1, a_2} = \min_{\substack{l = 1, 2, \ldots, n \\ l \neq a_1}} r_{a_1, l}.$$

k - t ý k r o k. Je-li již definováno (5) $a_1, a_2, a_3 \ldots, a_{2k-3}, a_{2k-2}$ $(2 \leqq k < n)$, definujme a_{2k-1}, a_{2k} vztahem

$$r_{a_{2k-1}, a_{2k}} = \min_{i, j} r_{i, j},$$

kde i probíhá všechna čísla $a_1, a_2, \ldots, a_{2k-2}$; j všechna ostatní z čísel $1, 2, \ldots, n$. Při tom budiž a_{2k-1} jedno z čísel (5), takže a_{2k} není obsaženo mezi čísly (5).

Je patrno, že při tomto postupu je mezi čísly (5) právě k čísel různých, takže pro $k < n$ lze k-tý krok provésti.

Řešení naší úlohy je nyní dáno tímto **tvrzením:**

1. J jest mkč.
2. Neexistuje žádná jiná mkč.
3. J se skládá z n-1 dvojic.

Důkaz provedu indukcí. Tvrzení 3. je patrně správné.

1. Podle první pomocné věty musí každá mkč . obsahovati množství

$$J_2 \equiv [a_1, a_2].$$

Množství J_2 jest souvislé a má právě dva indexy.

2. Budiž pro jisté celé k $(2 \leqq k < n)$ již dokázáno, že množství

$$J_k \equiv [a_1, a_2], [a_3, a_4], \ldots, [a_{2k-3}, a_{2k-2}]$$

je souvislá část s k indexy, jež jest obsažena v každé mkč. Potom podle 2. pomocné věty je také množství

$$J_{k+1} \equiv [a_1, a_2], [a_3, a_4], \ldots, [a_{2k-1}, a_{2k}]$$

obsaženo v každé mkč a má patrně $k + 1$ indexů (neboť a_{2k-1} patří k indexům množství J_k, a_{2k} nikoliv). Dále jest J_{k+1} souvislá část; neboť buďte p, q dva různé indexy množství J_{k+1}:

4

Dato curioso:

No solo fue este trabajo que no se le atribuyó su descubrimiento.

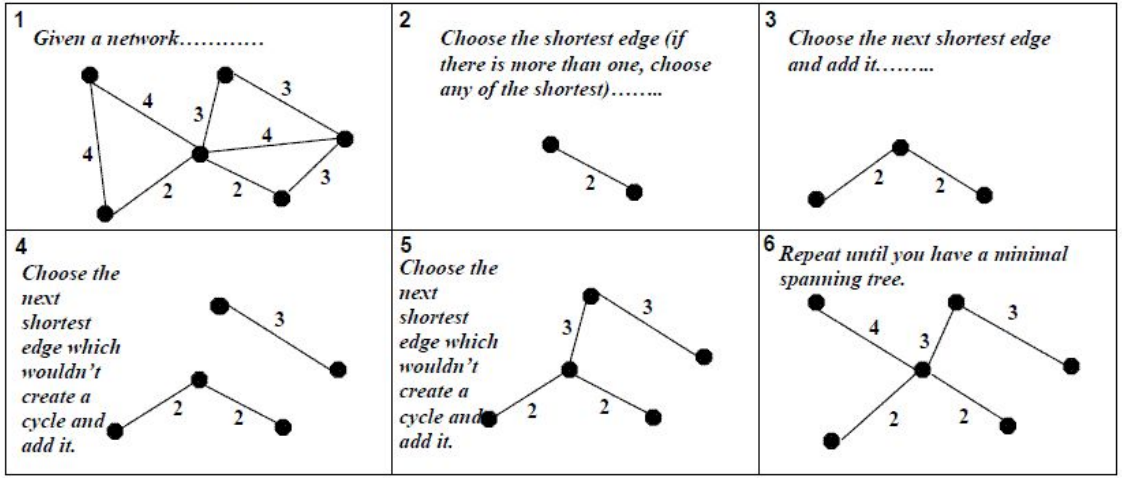
Uno de sus trabajos "*O minimálních grafech obsahujících n daných bodů*" es probablemente el más antiguo artículo sobre el Steiner Tree Problem.

El Steiner Tree Problem se puede interpretar como una generalización del MST.

3. Joseph Kruskal (1956)

En *“On the shortest spanning subtree of a graph and the travelling salesman problem”* , el científico *Joseph Kruskal* desarrolla un bello algoritmo que soluciona el MST.

Kruskal’s Algorithm



Demostración:

Es trivial demostrar que el árbol generado por el algoritmo es un Spanning Tree.

Supongamos que el árbol generado no es un MST. Sea $T = (V, E')$ el MST. Hacemos que las aristas sean de pesos diferentes.

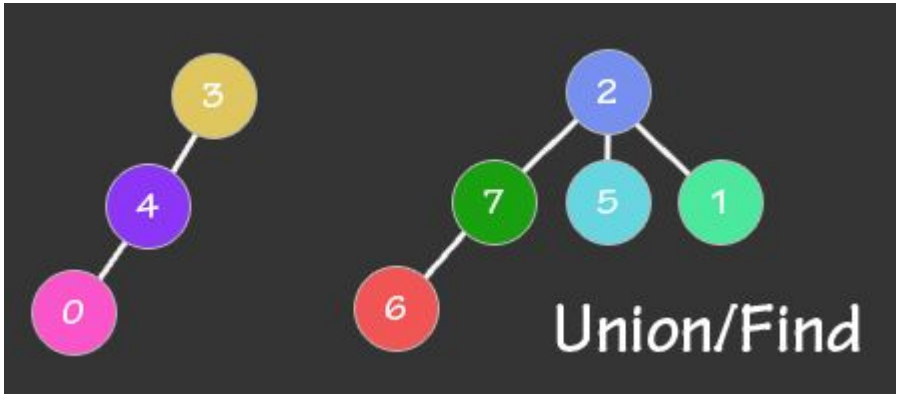
Sea $e \in E$ la primera arista que es añadida en el algoritmo pero no pertenece a T. Entonces $T + e$ contiene un ciclo (si a un árbol le añades una arista tendrá al menos 1 ciclo) . Sea otra arista $e' \in T$ que pertenece a ese ciclo.

Si $e' < e$ entonces se debió escoger e' antes que e ya que siempre se escoge las aristas en orden creciente. Si $e' > e$ entonces $T + e - e'$ es un spanning tree cuyo peso total es menor que el de T (que es un MST).

Entonces se demuestra que el árbol generado es un MST.

Disjoint Set Union (DSU)

Es una estructura de datos que te permite representar conjuntos disjuntos, unir conjuntos y hacer consultas sobre elementos del conjunto.



The Union-Find Data Structure

- Purpose:
  - Great for disjoint sets
  - Operations:

Union (S_1, S_2)	Performs a union of two disjoint sets (S_1 U S_2)
Find (x)	Returns a pointer to the set containing element *x*

Q) Under what scenarios would one need these operations?

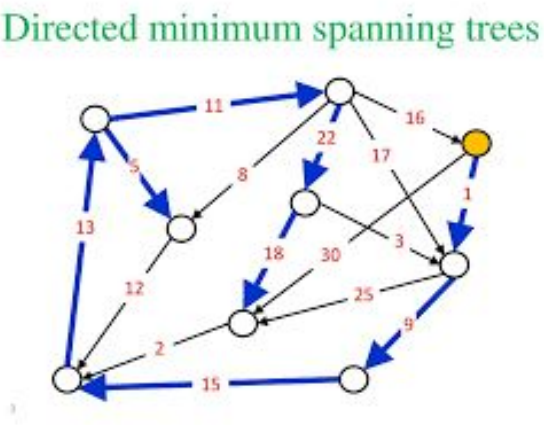
```
int parent[MAX];

int find(int u){
    if(parent[u] == u) return u;
    parent[u] = find(parent[u])
}

void uni(int u,int v){
    int parent_u = parent[u];
    int parent_v = parent[v];
    if(parent_u == parent_v)
        return;
    parent[v] = parent_u;
}
```

Minimum Directed Spanning Tree

Sea $G = (V , E)$ un grafo dirigido compuesto por N vértices y M aristas.

Por cada arista $e \in E$, definimos $w(e)$ como el peso de la arista.

Hallar la arborescencia para alguna raíz r $T = (V , E')$ donde para para todo vértice $v \in V$ existe un camino dirigido de r hacia v, además la sumatoria de $w(e)$ para todo $e \in E'$ es mínima.



Optimum Branching

A *branching* B of G is a set of edges such that

(i) if (x_1,y_1), (x_2,y_2) are distinct edges of B then $y_1 \neq y_2$;

(ii) B does not contain a cycle.

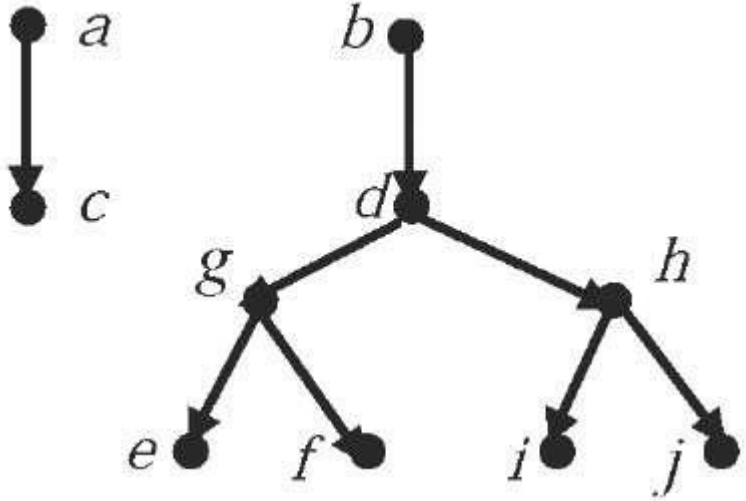
Given a real value c(v,w) defined for each edge of G, we desire a branching B such that $\sum_{(v,w)\in B} c(v,w)$ is maximum. Such a set B is called an *optimum branching*.

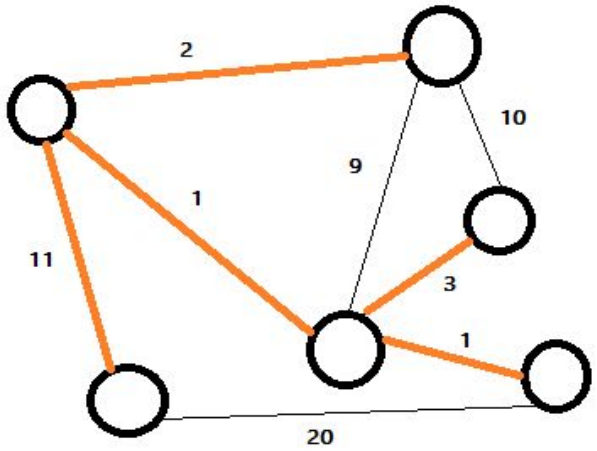
Equivalencia entre MDST y Optimum branching

Demuestre que ambos problemas son equivalentes.

Además demuestre que hallar el MDST para una raíz fija *u* es equivalente a calcular el MDST

¿Cómo?

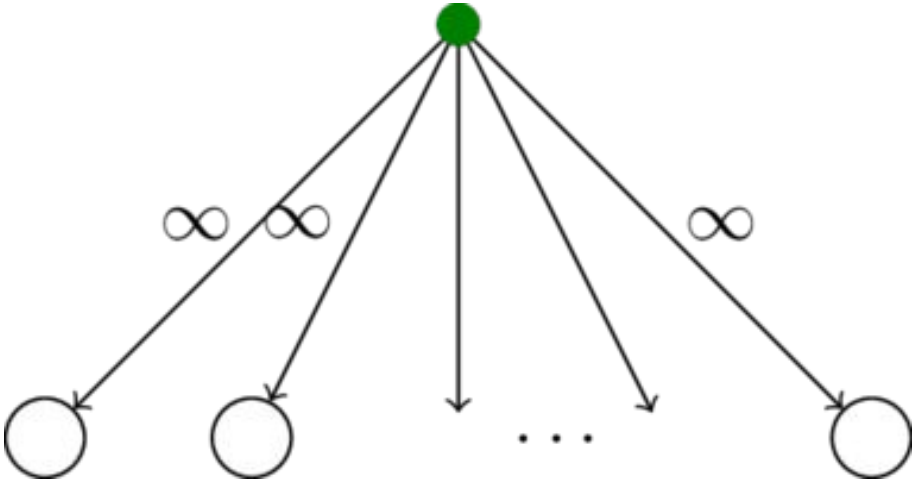




Minimum Spanning Tree (organge colored edges denote the edges in the MST)

OpenGenus Foundation

Con mucho amor creamos vértices y aristas



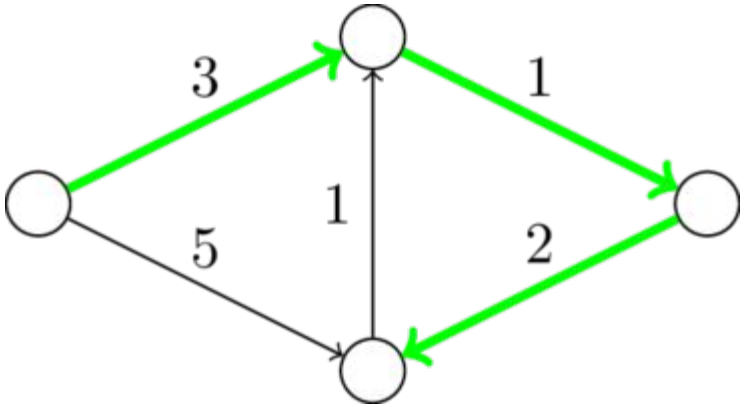
Mínimo equivale a máximo (en la mayoría de casos)

Creamos un vértice auxiliar desde el cual salen n aristas de peso +/- infinito.

El MDST enraizado en el nuevo vértice contiene el *optimum branching*

El MDST enraizado en el nuevo vértice contiene el MDST desde cualquier vértice del grafo anterior

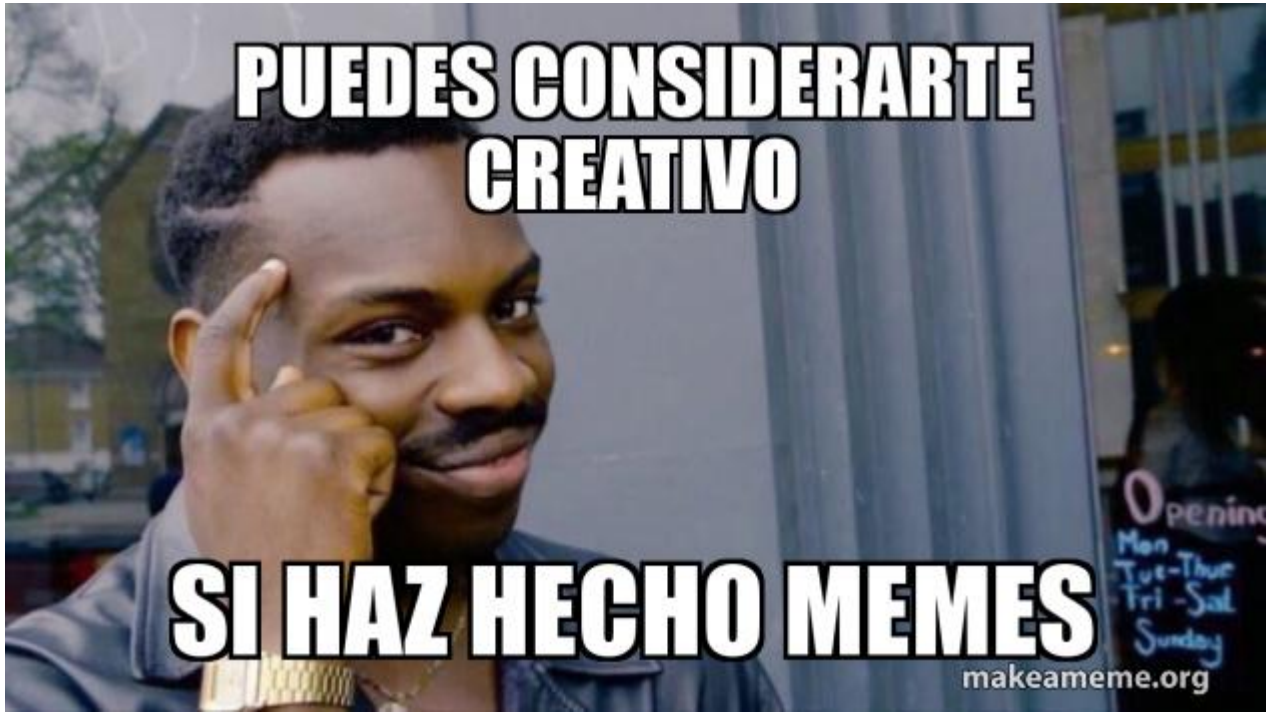
No es tan simple



La idea de Kruskal o Jarnik - Prim no funciona para el caso dirigido

¿Qué hacemos?

Tenemos que ser más ambiciosos aún.



Algoritmo de Edmonds

Calcularemos la arborescencia de peso mínimo enraizada en *r* entonces eliminamos todas las aristas entrantes a *r*.

Definimos una compresión del grafo $G = (V, E)$ a $C = (V, E')$ donde $E' = \{ (min\_edge(v)) : para\ todo\ v \in V \}$

Donde *min_edge(v)* es la arista entrante a *v* de peso mínimo.

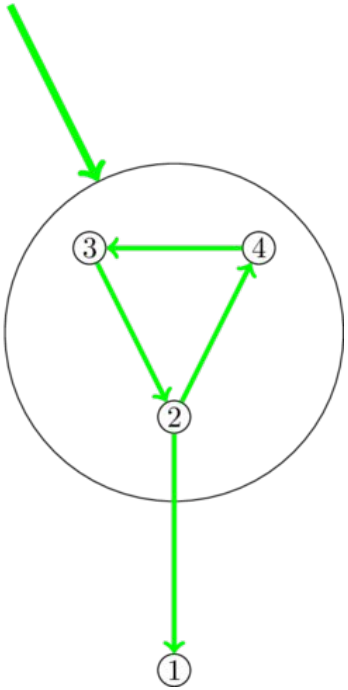
Si *E'* contiene no contiene ciclos entonces *C* es el MDST.

Si contiene al menos un ciclo, comprimimos el grafo $G = (V, E)$ a $G' = (V', E')$. Cada ciclo o vértice simple de C será un vértice en G'. y definimos una función $w(e)$, para todo $e \in E'$.

Donde $E' = E$ - loops que se forman al comprimir los ciclos.

Esta fase es la expansión

La magia de la compresión y expansión

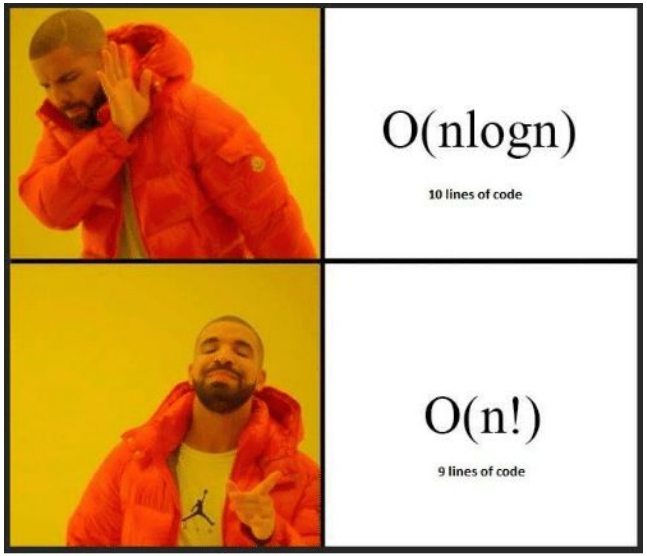


$$\bar{w}(e) = \begin{cases} w(e) - w(e_C) & \text{if } e \text{ enters } C, \\ w(e) & \text{otherwise.} \end{cases}$$

Al definir la nueva función de pesos, obtenemos que el DMST de G está en función del DMST de G'.

Se demuestra que el DMST de G contiene a cada ciclo formado en la comprensión menos exactamente una arista

Complejidad



After discovering that complexicity of the algorithm won't be taken into consideration on the exam...

Las implementaciones más simples tienen una complejidad de *O(n*m)*.

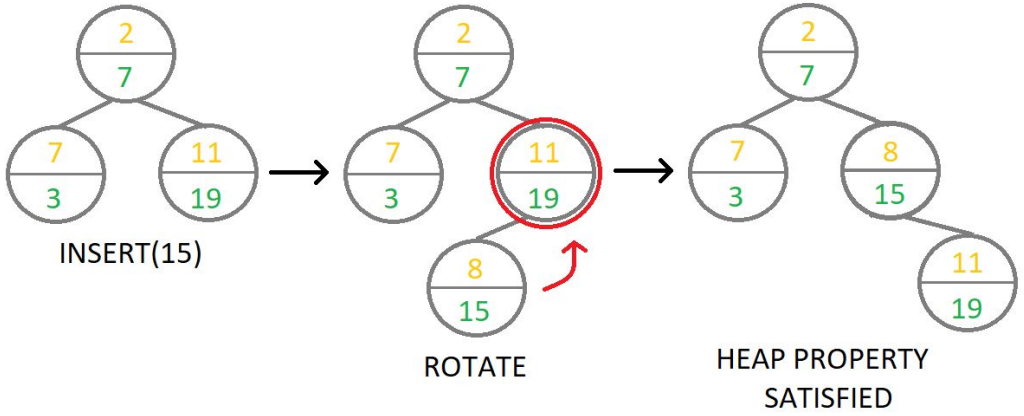
¿Es posible mejorar esta complejidad?

¿Qué necesitamos?

Una estructura que nos permita:

1) Obtener el mínimo de un conjunto.
2) Eliminar dicho mínimo.
3) Combinar estructuras.
4) Restar un número a todos los elementos del conjunto

¿Alternativas?



Soluciones

En O(N*M) : https://github.com/mhunicken/icpc-team-notebook-el-vasito/blob/master/graphs/chu_liu.cpp

Con treap en O(N + M log(M)) : https://pastebin.com/WUqyxrLh

Recursos

https://codeforces.com/blog/entry/82410

https://codeforces.com/blog/entry/20079 (en ruso)

https://www.math.uni-bielefeld.de/documenta/vol-ismp/30_nesetril-nesetrilova.pdf

https://cw.fel.cvut.cz/old/_media/courses/a4m33pal/cviceni/tarjan-finding-optimum-branchings.pdf

http://www.cs.tau.ac.il/~zwick/grad-algo-13/directed-mst.pdf