Minimum Spanning Tree(MST)

CS101 Fall 2024

CS101 Course Team

Nov 2024

Spanning Tree

Why Spanning Tree:

- Tree has a linear scale. ($|E| = \Theta(|V|)$)
- Tree has no cycle. (Unique Path)

Existence of spanning trees $\Leftrightarrow$ Connectedness.

Note: Path on Minimum Spanning Tree $\neq$ Shortest Path!

Another note (by Prof.):

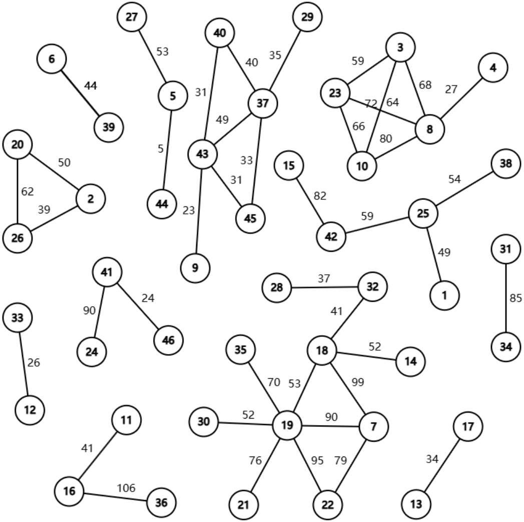
Weights may not be distinct! So MST may not be unique!

Another note (by ta):

Graph with distinct weights has a unique MST.

Application

Analyzing Network.



Number of Spanning Trees/Forests

Multiplication Rule: Count every connected component then multiply.
Naive Method:

- Selection: $\binom{|E|}{|V|-1}$.
- Cut: Enumerate all impossible edge combinations. (e.g. Ones form cycles.)

A universal method: Matrix-Tree Theorem.

Number of Spanning Trees/Forests

Notation

$G = (V, E)$ is a graph.
Degree Matrix: $D = \text{diag}\{d_i\}$, where d_i is degree of vertex i i.e. $\deg(i)$.
Adjacency Matrix: $A_{ij} = 1$ if $(i, j) \in E$ otherwise 0.
Laplace Matrix: $L = D - A$.

Notice that the sum of each line in the Laplace Matrix is 0, which means that $\det(L) = 0$.

Matrix-Tree Theorem

The number of spanning trees of G equals the determinant of (every) $(|V| - 1)$-level major minor of L.
Corollary: Note that L is semi-positive definite. $0 = \lambda_1 \leq \lambda_2 \cdots \lambda_{|V|}$ is the eigenvalue of L. Then the number of spanning trees is $\frac{1}{n}\lambda_2 \cdots \lambda_{|V|}$

One corollary: K_n has n^{n-2} spanning trees.

Minimum Spanning Tree

- From edges: Kruskal's Algorithm
- From vertices: Prim's Algorithm

Public Notation

$G = (V, E)$ is a connected weighted simple graph.
$n = |V|, m = |E|$.
$V = \{v_1, \cdots, v_n\}, E = \{e_1, \cdots, e_m\}, e_i = (x_i, y_i, w_i)$ refers to the edge connected x_i and y_i weighted w_i.
Sometimes also see E as a subset of $V \times V$. (ignore weights)

Cut Property & Cycle Property

Cut: A partition which divides V into 2 disjoint sets: S and $V \backslash S$.
Crossing edges: The edges in $|E| \cap S \times V \backslash S$.

Cut property

For all cuts, the minimum weighted crossing edges must be in an MST.
Dual Proposition: Cycle property. (Maximum weighted cycle edges must not be in any MSTs.

More referred to: `https://piazza.com/class_profile/get_resource/m11nrjwg6r35ku/m3u111zjdd05de`

Prim's Algorithm

Core idea: Using cut property to extend edges.
Initialization: Choosing a vertex v, $S = \{v\}$. $C = \{e | e \text{ contains } v\}$.

Every iteration

1. Find the minimum crossing edge e between S and $V \backslash S$.
2. Denote the crossing edge as $e = (u, v), u \in S$. Add v into S.
3. Add all additional crossing edges into C. $(\{(u, v) | u \in V \backslash S\})$

Naive version: $O(|V|^2)$. (Maintain distance of every vertex.)
Using binary heaps to maintain C: $O(|E| \log |V|)$.
Correctness: Cut property.

Kruskal's Algorithm

Core idea: Always find the legal minimum weighted edges.
Initialization: Make E sorted. $T = \emptyset$
Traverse all edges $e = (u, v)$.
Check whether u and v are connected in T. If so, skip it.
Otherwise, add $e = (u, v)$ in T.

Pseudocode (Cite from HTTPS://OI-WIKI.ORG/GRAPH/MST/)

Input. The edges of the graph e, where each element in e is (u, v, w) denoting that there is an edge between u and v weighted w.
Output. The edges of the MST of the input graph.
Method:
result $\leftarrow \varnothing$
sort e into ascending order by weight w
for each (u, v, w) in the sorted e
    if u and v are not connected in the union-find set
        connect u and v in the union-find set
        $result \leftarrow result \bigcup \{(u, v, w)\}$
return *result*

Kruskal's Algorithm

Time complexity:
Naive version: $O(|V||E|)$.
Optimization with disjoint sets: $O(|E|\log|E|)$.
Correctness: Using contradiction method:
Adding the extra edge and T forms a cycle, using the cycle property, which contradicts to the algorithm.

Example

2023 HW8

In SC101 country, there are n cities and m **broken** roads, with each broken road connecting two different cities. You can consider this as a graph $G = (V, E)$.
Now the government wants to build a traffic net in the SC101 country. There are 2 crucial steps to take in constructing this traffic network:

1. Establish an airport in the i-th city with a cost of a_i
2. Repair the broken road $e_j = (u_j, v_j)$ to connect the city u_j and v_j, cost b_j.

In the final network, every city will either have an airport or be connected to a city with an airport. Your task is to design an algorithm to find the minimum cost to build the network.

Solution: MST.

Network Flow*

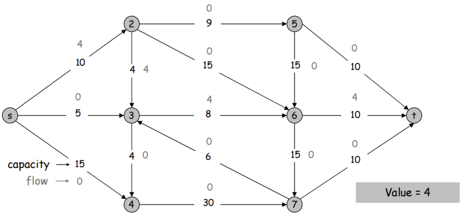
Abstraction for material flowing through the edges.
G = (V, E) = directed graph
Two distinguished nodes: s = source, t = sink.
c(e) = nonnegative capacity of edge e

Flows

Def. An s-t flow is a function that satisfies:

- For each e ∈ E: $0 \leq f(e) \leq c(e)$ (capacity)
- For each v ∈ V − {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

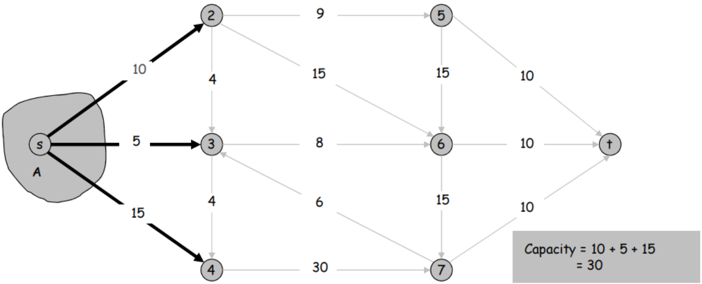
Def. The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.



Cuts

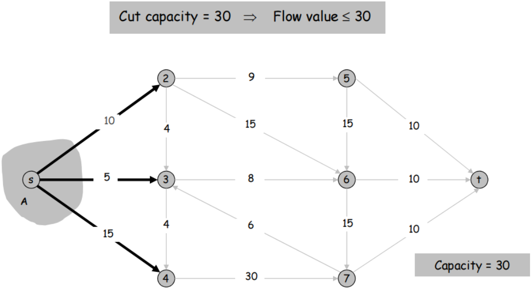
Def. An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$.

Def. The capacity of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$



Network Flow*

Weak duality: Let v be any flow, and let (A, B) be any s-t cut. Then $v \leq \text{cap}(A, B)$
Strong duality(Max-flow min-cut theorem) [Ford-Fulkerson 1956]: The value of the max flow is equal to the value of the min-cut.



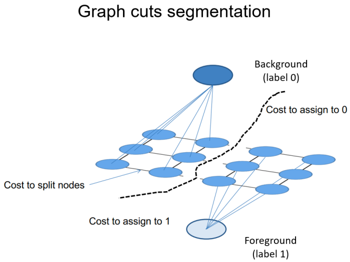
Network Flow*

Algorithms for solving the max-flow / min-cut.
$n = |V|, m = |E|, m = \Omega(n)$

1. Edmonds–Karp(EK) algorithm: $O(nm^2)$
2. Dinic: $O(n^2 m)$
3. Highest Label Preflow Push(HLPP): $O(n^2\sqrt{m})$

Network Flow*

1. Bipartite (Perfect) Matching
2. Survey Design
3. Project Selection
4. Image Segmentation



More details for applications and proofs: welcome to CS240!