Operations Research

Minimum Spanning Trees

Safia Kedad-Sidhoum

safia.kedad_sidhoum@cnam.fr

Cnam

2025-2026

Minimum Spanning Tree

Problem Statement

- we want to link a list of objects so that the total length of the links is minimal
- We consider a weighted graph in which :
  - the vertices are the objects to be linked
  - an edge represents a possible link
  - the weight of an edge is the length of the corresponding link

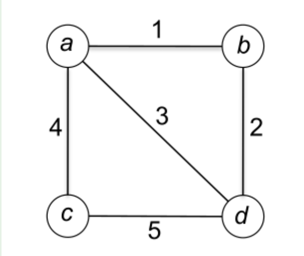
Problem Definition

- Input : A connected undirected graph $G = (V, E)$ and a real-valued cost c_e for each edge $e \in E$.
- Output : A spanning tree $T \subseteq E$ of G with the minimum possible sum $\sum_{c_e \in T} c_e$ of edge costs.
  A *spanning* tree is a tree that covers all the vertices of G.
- Assumption : The input graph $G = (V, E)$ is connected, with at least one path between each pair of vertices.

Example

What is the minimum sum of edge costs of a spanning tree of the following graph ?

Example



Prim's algorithm

- named after Robert C. Prim, who discovered the algorithm in 1957
- greedy algorithm (can be seen as a graph search algorithm)
- Rationale :
  - Prim's algorithm begins by choosing an arbitrary vertex,
  - construct a tree one edge at a time,
  - In each iteration, add the cheapest edge that extends the reach of the tree-so-far

Example

Compute a minimum spanning tree of the graph of the previous example

Prim's algorithm

Pseudo-code

Prim

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

// Initialization
$X := \{s\}$ // s is an arbitrarily chosen vertex
$T := \emptyset$ // invariant: the edges in T span X
// Main loop
while there is an edge (v, w) with $v \in X, w \notin X$ **do**
  $(v^*, w^*) :=$ a minimum-cost such edge
  add vertex w^* to X
  add edge (v^*, w^*) to T
return T

Running time ?

Prim's algorithm

Running time

- Straightforward : $O(n^2)$
- Heap based : $O((n + m) \log n)$

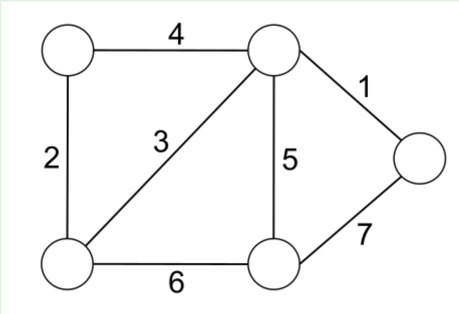
Kruskal's algorithm

- Greedy algorithm
- Constructs a spanning tree one edge at a time
- Rather than growing a single tree from a starting vertex, Kruskal's algorithm can grow multiple trees in parallel until the end of the algorithm where a single tree is obtained
- Kruskal's algorithm considers the edges of the input graph one by one, from cheapest to most expensive

Example

What is the minimum spanning tree of the following graph ?

Example



Kruskal's algorithm

Pseudo-code

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G

```
// Preprocessing
T := ∅
sort edges of E by cost  // e.g., using MergeSort
// Main loop
for each e ∈ E, in nondecreasing order of cost do
    if T ∪ {e} is acyclic then
        T := T ∪ {e}
return T
```

Running time ?

Kruskal's algorithm

Running time

- Straightforward : $O(m \log m + nm)$
- Union-Find based : $O(m \log m + (n + m) \log n)$

Correctness

- Prim's algorithm
- Kruskal algorithm

on the blackboard