Chapter 4- Minimum spanning trees

We deal with undirected graph.

Definition

- let $G = (V, E)$ be a graph, a **partial subgraph** of G is a graph $G' = (V, F)$ with $F \subset E$
- a tree is an acyclic conneted graph
- a spanning tree of $G = (V, E)$ is a partial subgraph of G that is a tree.

Theorem

Theorem

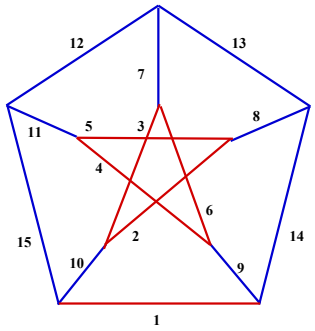
The following conditions are equivalent:

1. *T is a tree*
2. *T is a connected graph and $m = n - 1$ (m number of edges, n number of vertices)*
3. *T is an acyclic graph and $m = n - 1$*
4. *T is a connected graph and every edge of T is a* **bridge**
5. *T is an acyclic graph and if we link any two non adjacent vertices of T then we create a cycle (a unique cycce)*
6. *For any two vertices a et b, it exists a unique path linking a and b.*

Fundamental cycle

Definition

Let $G = (V, E)$ be graph and $T = (V, F)$ a spanning, the fundamental cycle associated to $e \in E \setminus F$ is the unique cycle $C_T(e)$ of $T \cup e$



Minimum spanning tree - Problem

Definition

Let $G = (V, E)$ be a graph $p : E \to \mathbb{R}$ be a weight funcion. Let $T = (V, F)$ be a spanning tree of G then:
$p(T) = \sum_{e \in F} p(e)$

Problem

Let G be a graph with a weighted function, find a minimum (weighted spanning) tree for G.

Kruskal Algorithm (1956)

MST-Kruskal(G,p)

/* the edges are sorted into the nondecreasing order by weight */

/* $p(e1) \leq p(e_2) \leq \cdots \leq p(e_m)$ */

1 $S \leftarrow \emptyset$

2 $k \leftarrow 0$

3 **while** $k < n$ **and** $|S| < n-1$

4 **do if** $(V, S \cup e_{k+1})$ is acyclic.

5 **then** $S \leftarrow S \cup e_{k+1}$

6 $k \leftarrow k+1$

Analysis-Complexity

We prove by contradition that:

Theorem

Any spaning tree $T^* = \{e_1, , e_2, \cdots, e_{n-1}\}$ obtained by Kruskal algorithm is optimal.

For the acyclic test, we give a numbering to the vertices from 1 to n and we take an edge iff the number of the extremities of the edge are distinct. Then all the vertices having the same number as one of the extremity is renumbered with the samllest nuber of the extremities.

Complexity:

1. to sort the edges: $m \log m$
2. to compare the extremities of the edges $\theta(m)$
3. to renumber the vertices $O(n^2)$

The complexity is: $O(n^2 + m \log m)$

Prim Algorithm (1956)

MST-Prim (G,p,r)

1 $S \leftarrow \emptyset$

2 $A \leftarrow r$ /*A is the set of marked vertices*/

3 **while** $|S| < n-1$

4 **do** $u \leftarrow xy$ s.t. $x \in A$, $y \notin A$ **and** $p(xy)$ is minimum.

5 $S \leftarrow S \cup u$

6 $A \leftarrow S \cup y$

Analysis

We prove by induction on the number of steps that:

Theorem

If S is the set of selected edges then it exists a minimum spanning tree containg S

Prim Algorithm with priority queue

MST-Prim-Q(G,p,r)

```
1     for each u ∈ V
2         do key[u] ← ∞
3            Π[u] ← NIL
4     Key[r] ← 0
5     Q ← V
6     while Q ≠ ∅
7     do u ← Extract-Min(Q)
8         for each v ∈ Adj[u]
9             do if v ∈ Q and p(uv) < Key[v]
10                then Π[v] ← u
11                     Key[v] ← p(uv)
```

Complexity

- Build the heap $O(n)$
- Extract minimum $\log n$, we do it n times
- line 8 to 11 $O(m) \times O(\log n)$
- Total: $O(n \log n + m \log n)$, it is $O(m \log n)$

Chapter 5- Shortest Paths
1- Definitions

Let $G = (V, E)$ be a directed graph.
Let $w : E \to \mathbb{R}$ be a weight function. For $u \in E$ $w(u)$ is the weight of u

Definition

- A path is a alternated sequence of vertices and edges $x_1, u_1, x_2, u_2, \ldots, u_k, x_{k+1}$ such that $u_i = \overrightarrow{x_k x_{k+1}}$.
- If P is a path, the weight of P is $w(P) = \sum_{u \in P} w(u)$
- An **elementary path** does not go through the same vertex twice.
- A root of G is a vertex s of G such that for each $x \in V \setminus \{s\}$ it exists a path from s to x.
- A **circuit** is a closed path.

1- Definitions

Definition

- A circuit C is said **an absorbing circuit** if : $w(C) = \sum_{u \in C} p(u) < 0$
- An **arborescence with root** r in a directed graph $G = (V, E)$ is a sub-graph such that the underlying undirected graph is a tree and it exists a path from s to any vertex of the arborescence.
- A **shortest path** linking two vertices x and y is a path linking x to y with a smallest weight.

Theorem

Let $G = (V, E)$ be a directed weighted graph. Let $s \in V$, then for any $x \in V \setminus \{s\}$ it exists a path linking s to x iff:

1. *s is a root of G*
2. *G does not contain an absorbing circuit.*

Single-source shortest path
1: The weights are nonnegative: Dijkstra Algorithm

Remark. There is no absorbing circuit

Idea. We start from the vertes s. we build a set of edges by taking successively edges with initial vertex in D (the set of discovered vertices) and the final extremity not in D.

Dijkstra Algorithm

Dijkstra(G,s)

```
1   D ← s; A(s) ← ϵ; x_1 ← s; d(s) ← 0; k ← 1
2   for each (x ∈ V and x ≠ s)
3       do d(x) ← ∞
4   while (k < n and d(x_k) < ∞)
5       do for each u ∈ E s.t. I(u) = x_k and T(u) ∉ D
6           x ← T(u)
7           if d(x) > d(x_k) + w(u)
8               then d(x) ← d(x_k) + w(u)
9                    A(x) ← u
10      Choose x ∉ D s.t. d(x) = min{d(y), y ∉ D}
11      k ← k + 1
12      x_k ← x
13  D ← D ∪ {x_k}
```

Complexity

- line 2-3 $O(n)$
- line 5-9 $O(n)$
- line 10 to find the minimum $O(n^2)$
- total: $O(n^2)$

2- Graphs without circuit. DAG. Bellman Algorithm

Bellman(G,s)

1 $D \leftarrow s$; $A(s) \leftarrow \epsilon$; $x \leftarrow s$; $d(s) \leftarrow 0$
2 **for** each ($x \in V$ **and** $x \neq s$)
3 **do** $d(x) \leftarrow \infty$
4 **while** it exists $x \notin D$ with all its predecessors in D
5 **do** $d(x) \leftarrow \min_{u/T(u)=x, I(u)\in D}\{d(I(u)) + w(u)\}$
7 let $\tilde{u}$ s.t. $d(x) = \min_{u/T(u)=x, I(u)\in D}\{d(I(u)) + w(u)\}$
8 $A(x) \leftarrow \tilde{u}$
9 $D \leftarrow D \cup \{x\}$

Complexity : $O(n^2)$

$G = (V, E, w)$
de $\mathtt{num}(1)$=s

Bellman-with-topological-sort(G,s)

1 $\mathcal{A}(s) \leftarrow \epsilon$
2 $d(s) \leftarrow 0$
3 **for** each $x \in V$, $x \neq s$
4 **do** $d(x) \leftarrow +\infty$
5 **for** $j \leftarrow 2$ to n
6 **do** $y \leftarrow \mathtt{de\ num}(j)$
7 $d(y) \leftarrow min_{u/\vec{xy}=u}[d(x) + w(u)]$
8 let $\tilde{u}$ such that $d(y) = d(x) + w(\tilde{u})$
9 $\mathcal{A}(y) \leftarrow \tilde{u}$

Complexity : $O(n^2)$

All-pairs shortest paths 1-Floyd Algorithm

We assume that the graph $G = (V, E)$ has a weight function ω. $V = \{1, 2, \ldots, n\}$ and G does not contain absorbing circuit. We proceed step by step. We consider the paths linking i and j such that all intermediate vertices are in $\{1, \cdots, k\}$ at the step k. we denote by:

- $d_k(i, j)$ the weight of a shortest path using only intermediates vertices belonging to $\{1, \cdots, k\}$
- $pred_k(i, j)$ the predecessor of j in such a path.

We use the $n \times n$ matrix A such that :

$$a_{ij} = \begin{cases} \omega(\overrightarrow{ij}) & si\ \overrightarrow{ij} \in E \\ +\infty & si\ \overrightarrow{ij} \notin E \end{cases}$$

At each step k $a_{ij} = d_k(i, j)$

And the $n \times n$ matrix P :

$$P = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{nn} & \cdots & x_{nn} \end{pmatrix}$$

where $P_{ij} = pred_k(i, j)$ At the beginning $P_{ij} = i$

Floyd(G,w)

```
1     for i ← 1 to n
2         do for j ← 1 to n
3                A[i,j] ← ω(i,j)
4                P[i,j] ← i
5     do for k ← 1 to n
6         do for i ← 1 to n
7               do for j ← 1 to n
8                   do if A[i,k] + A[k,j] < A[i,j]
9                      then A[i,j] ← A[i,k] + A[k,j]
10                          P[i,j] ← P[k,j]
```

Complexity : $O(n^3)$

Existence of paths between all pairs of vertices 1-Matrix

Let $G = (V, E)$ be a directed graph with $V = \{1, 2, \ldots, n\}$. We want to know if it exists a path from i to j for each couple (i, j) of vertices of G.

Definition

Let $G = (V, E)$ be a directed graph. We define the boolean matrix:

$$A = (a_{ij}) \begin{array}{l} 1 \leq i \leq n \\ 1 \leq j \leq n \end{array}$$

$$a_{i,j} = 1 \Leftrightarrow \vec{ij} \in E$$

Proposition

Let $p \geq 2$ be a integer, we define

$$A^{(p)} = (a_{ij}^{(p)}) \begin{array}{l} 1 \leq i \leq n \\ 1 \leq j \leq n \end{array}$$

$$A^{(p)} = A^{(p-1)}.A$$

We have : $a_{ij}^{(p)} = 1 \Leftrightarrow$ it exists a path from i to j with a length equal to p

2- Transitive closure- Warshall Algorithm

Definition

Let $G = (V, E)$ be a directed graph. The transitive closure of G is the graph $G^* = (V, E^*)$ such that:

$$\vec{ij} \in E^* \Leftrightarrow \text{it exists a path linking } i \text{ to } j \text{ in } G$$

We will use the boolean matrix:

$$A = (a_{ij}) \begin{array}{l} 1 \leq i \leq n \\ 1 \leq j \leq n \end{array}$$

for $i = j$ $a_{i,j} = 1$

$$a_{i,j} = 1 \Leftrightarrow \vec{ij} \in E$$

Warshall Algorithm

Warshall(G)

```
1   for i ← 1 to n
2       do for j ← 1 to n
3           do A[i,j] ← c(i,j)
4   do for k ← 1 to n
5       do for i ← 1 to n
6           do for j ← 1 to n
7               do A[i,j] ← (A[i,j] ∨ (A[i,k] ∧ A[k,j]))
```

Complexity : $O(n^3)$

Chapter 6- Maximum Flow
1- Definitions

Definition (Network)

Let $G = (V, E)$ be a directed graph.

- $c : E \to \mathbb{R}^+$. For each $e \in E$, $c(e)$ is a nonnegative capacity.
- s and p two vertices of G, s is the source, p the sink of the network.
- a special edge $u_r = ps$ is the back edge, $c(u_r) = +\infty$