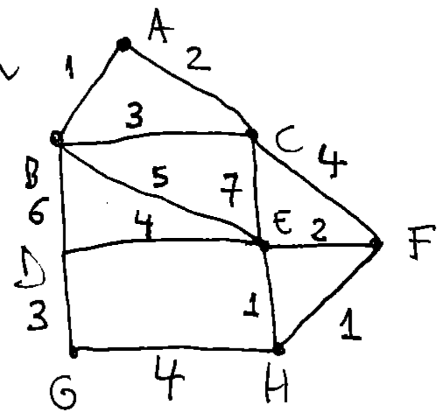
Kruskal's Algorithm for finding minimal spanning trees:

Let (V, E) be a connected graph w/ an associated cost function $c: E \to \mathbb{R}$.

1. Start w/ $(V, \emptyset)$, the subgraph of (V, E) consisting of all the vertices of (V, E) and no edges.
2. List all edges in E in a queue so that the cost of the edges is non-decreasing in the queue, i.e. if $e, e' \in E$ and if $c(e) < c(e')$, then e precedes e' in the queue.
3. Take edges successively from the front of the queue, and determine whether or not the addition of that edge to the current subgraph will create a cycle (circuit). If a circuit is created by this addition, discard the edge; otherwise, add it to the subgraph. Continue until the queue is emptied.

We will first do an example, and after the example we will prove Kruskal's algorithm yields a spanning tree that is indeed minimal.

Example: Consider



The cost function here is

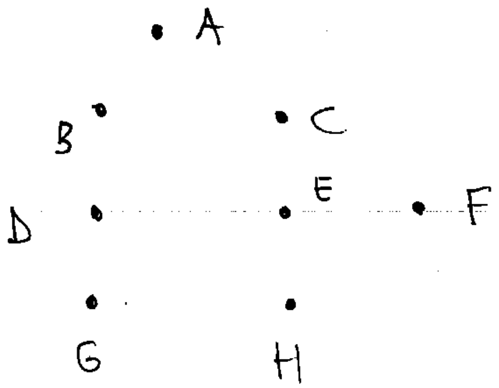
$c(AB) = 1$ $\quad c(CE) = 7$
$c(AC) = 2$ $\quad c(CF) = 4$
$c(BC) = 3$ $\quad c(EF) = 2$
$c(BD) = 6$ $\quad c(EH) = 1$
$c(BE) = 5$ $\quad c(FH) = 1$
$c(DE) = 4$ $\quad c(DG) = 3$

$c(GH) = 4$

We can also use a table format to write down the cost function $c : E \longrightarrow \mathbb{R}$

AB	AC	BC	BD	BE	DE	CE	CF	EF	EH	FH	DG	GH
1	2	3	6	5	4	7	4	2	1	1	3	4

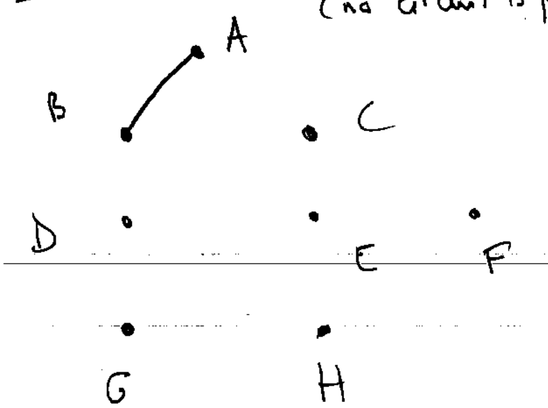
We start w/ $(V, \emptyset)$. This is step 0 of the algorithm. (The starting state).



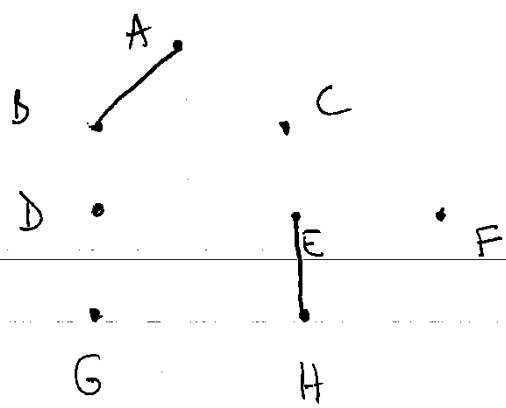
We list the edges in a queue so that the cost is non-decreasing

AB, EH, FH, AC, EF, BC, DG, DE, CF, GH, BE, BD, CE

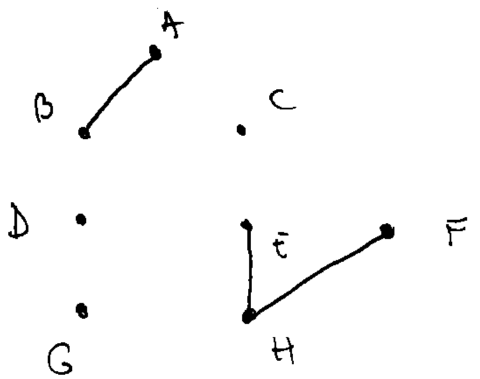
<u>Step 1</u> We can add AB (no circuit is formed)



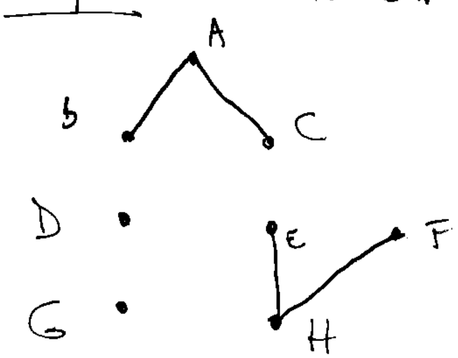
<u>Step 2</u> We can add EH



<u>Step 3</u> We can add FH



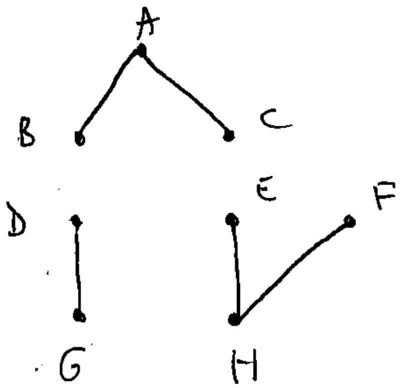
<u>Step 4</u> We can add AC

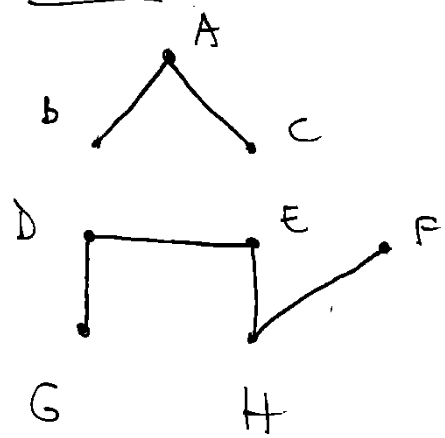


Step 5 We cannot add edge EF because we would create circuit EFHE, so EF gets discarded.

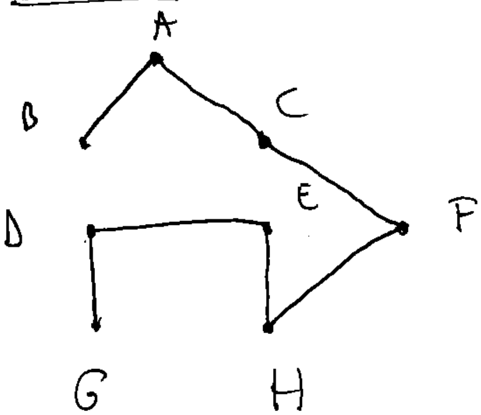
Step 6 We cannot add edge BC because we would create circuit ABCA, so BC gets discarded.

Step 7 We can add DG. Step 8 We can add DE.





Step 9 We can add CF.



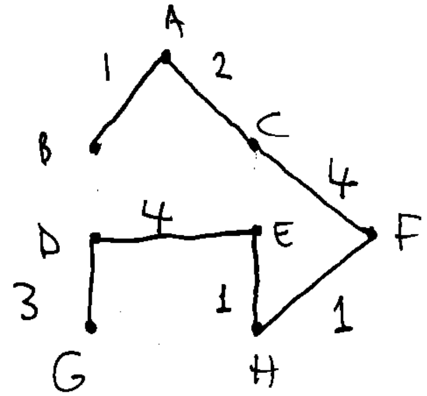
Step 10 We cannot add GH because we would create circuit DEHGD.

Step 11 We cannot add BE because we would create circuit BEHFCAB.

Step 12 We cannot add BD because we would create circuit BDEHFCAB.

Step 13 We cannot add edge CE because we would create circuit CEHFC.

The minimal spanning tree given by Kruskal's algorithm is thus:



min Cost $c(E_T) = 1+2+4+1+1+4+3 = 16$

Now that we have some intuition about the Kruskal algorithm, let us prove that it always yield a spanning tree that is indeed minimal.

Proposition Let (V,E) be a connected graph with associated cost function $c: E \to \mathbb{R}$. Kruskal's algorithm yields a spanning tree of (V,E).

Proof Since an edge is added from the queue only if no circuit is formed, we conclude the subgraph (V,E') of (V,E) produced by the Kruskal algorithm must be acyclical (i.e. contain no circuits.). To prove (V,E') is a spanning tree of (V,E), we must show (V,E') is connected. Assume not, Then (V,E') has components $(V_1, E_1'), (V_2, E_2'), \ldots, (V_m, E_m')$ for $m \geq 2$. (V,E) is connected, however $\Rightarrow \exists$ edge $e_{ij} \in E$ for $1 \leq i,j \leq m$, $i \neq j$ s.t. adding edge e_{ij} connects (V_i, E'_i) and (V_j, E'_j), but edge e_{ij} could not have possibly created a circuit when considered in the queue $\Rightarrow \Leftarrow$ (V,E') cannot have more than one connected component $\Rightarrow (V,E')$ is connected. (q.e.d.)

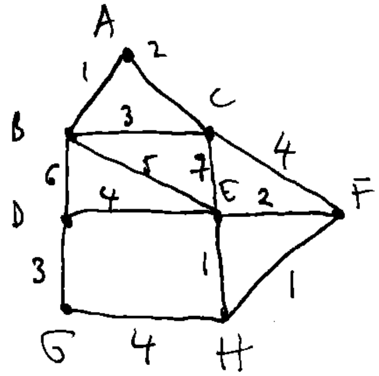
<u>Proposition</u> Let (V, E) be a connected graph w/ associated cost function $c: E \to \mathbb{R}$. Kruskal's algorithm yields a minimal spanning tree of (V, E).

<u>Proof</u> We already showed in the previous proposition that Kruskal's algorithm yields a spanning tree. Now we have to show that spanning tree is minimal w.r.t. $c: E \to \mathbb{R}$. Let (V, E') be the spanning tree given by the algorithm. If $(V, E') = (V, E)$, i.e. if the original connected graph is a tree, then there is nothing to prove. Assume $(V, E') \neq (V, E)$ i.e. (V, E) contains some circuit. Let all the edges of (V, E) be $e_1, e_2, \dots, e_m$ that we label s.t. $c(e_i) \leq c(e_j)$ $\forall$ $1 \leq i < j \leq m$. In other words, $c(e_1) \leq c(e_2) \leq \dots \leq c(e_{m-1}) \leq c(e_m)$. Kruskal's algorithm chooses the lowest cost $\#(V) - 1$ edges from $e_1, e_2, \dots, e_m$ such that the resulting subgraph is a spanning tree of (V, E). Therefore, if (V, E'') is any other spanning tree of (V, E), then $c(E') \leq c(E'')$. (q.e.d.)

<u>Def</u>: Let (V, E) be a connected graph w/ associated cost function $c: E \to \mathbb{R}$. Let (V, E') be the minimal spanning tree of (V, E) produced by Kruskal's algorithm. (V, E') is called the <u>Kruskal tree</u>.

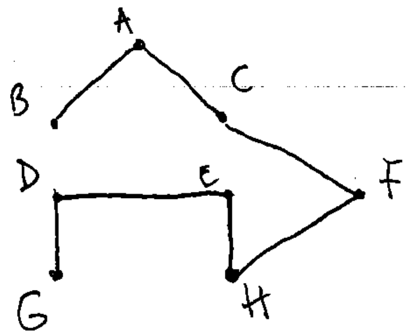
<u>NB</u> If two or more edges have the same cost, then we can reshuffle them in the queue used to determine the Kruskal tree. Therefore, the Kruskal tree might not be unique. In the example we used to illustrate Kruskal's algorithm

We see this scenario at work:

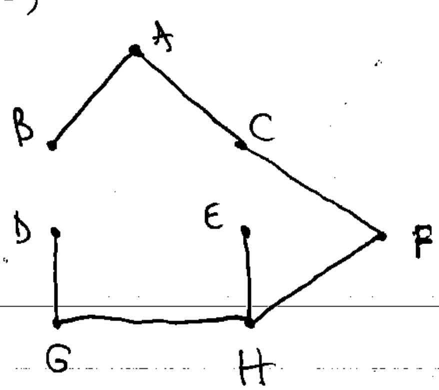


$C(DE) = C(GH) = 4.$

We used the queue AB, EH, FH, AC, EF, BC, DG, DE, CF, GH, BE, BD, CE to produce the Kruskal tree



whereas the queue AB, EH, FH, AC, EF, BC, DG, GH, CF, DE, BE, BD, CE would have produced the Kruskal tree



which has the same cost.

<u>Remarks</u> ① Joseph Kruskal published this algorithm that bears his name in 1956, two years after he finished his PhD at Princeton. Kruskal is known for work in computer science, combinatorics, and statistics.

② The cost of an edge is sometimes called the <u>weight</u> of that edge.

(3) Kruskal's algorithm starts w/ a disconnected graph $(V, \emptyset)$ and adds edges until the graph becomes connected and a tree, thus a spanning tree. In other words, until the last addition of an edge, the graph is disconnected.

$\boxed{\text{Prim's Algorithm}}$

<u>Task</u> Describe another algorithm for constructing the minimal spanning tree, which is characterized by the fact that at each step of the algorithm, the subgraph is a tree. This algorithm is called Prim's Algorithm.

Vojtěch Jarník first discovered and published this algorithm in 1930. Robert Prim subsequently rediscovered and published it in 1957. It was once again rediscovered by Edsger Dijkstra in 1959.

$\boxed{\text{Moral of the story}}$

The idea behind this algorithm is very natural. We apply procedure (2) for constructing a spanning tree that we discussed before using the same queue of edges ordered by cost as in Kruskal's algorithm. The result at each step is a tree, and at the end we get a minimal spanning tree.

<u>Prim's algorithm</u>

Let (V, E) be a connected graph w/ an associated cost function $c: E \longrightarrow \mathbb{R}$.

1. Start by choosing some vertex $v \in V$. Our starting subgraph is $(\{v\}, \emptyset)$.
2. List all edges in E in a <u>queue</u> so that the cost of the edges is non-decreasing in the queue, i.e. if $e, e' \in E$ and if $c(e) < c(e')$, then e precedes e' in the queue.

3. We identify the first edge in the queue, which has one vertex included in the current subgraph and the other vertex not included in the subgraph. We add that edge to the current subgraph as well as the vertex not already included. Since the subgraph with which we started was a tree, the resulting subgraph is a tree (we added one vertex and one edge). Continue this process until it is not possible to proceed any further, i.e. we have added all vertices in V. (51)