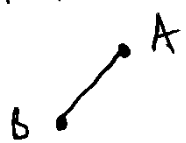
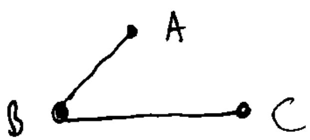
Now we follow procedure (2). We start w/ a vertex in V, and at each step we add an edge from E such that this edge is adjacent to a vertex already in the collection of vertices and also to a vertex that is not already in the collection. In other words, at each step, we add a vertex and an edge such that the resulting graph is connected. We stop once we capture all vertices in V.

We start w/ vertex A.

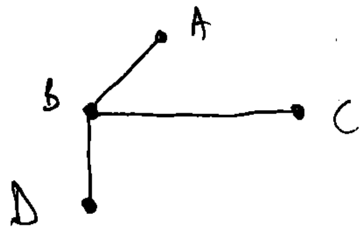
We could add vertex B and edge AB OR we could add vertex C and edge AC. We choose to add vertex B and edge AB.



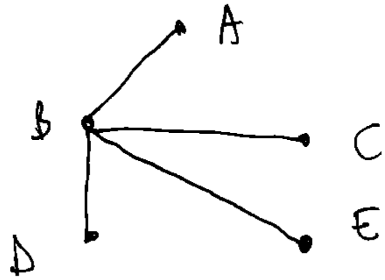
Next, we choose to add vertex C and edge BC



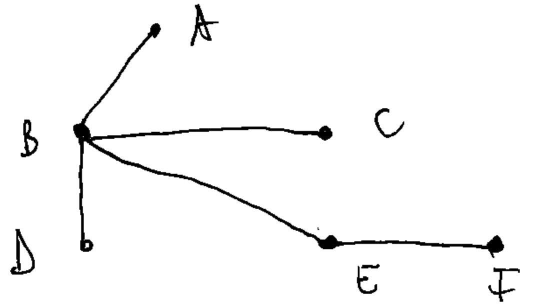
Next, we choose to add vertex D and edge BD.



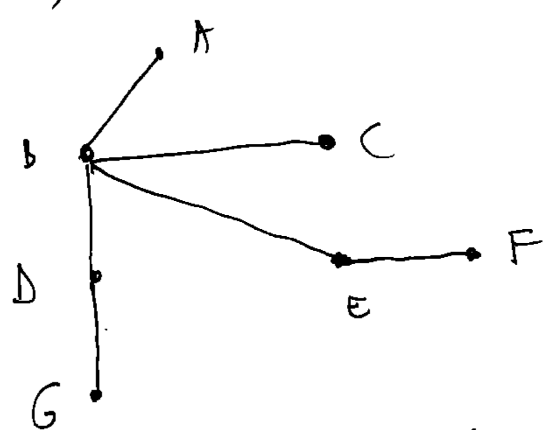
Next, we choose to add vertex E and edge BE.



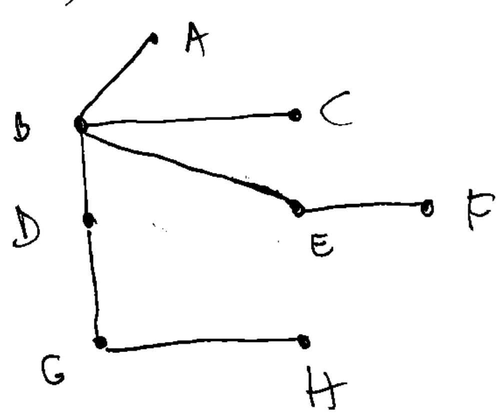
Next, we choose to add vertex F and edge EF.



Next, we choose to add vertex G and edge DG.



Next, we choose to add vertex H and edge GH.



We now have all vertices in $V = \{A, B, C, D, E, F, G, H\}$. We started w/ 1 vertex and 0 edges. At each step we added 1 vertex and 1 edge $\Rightarrow$ at each step i, if V_i is the set of vertices at step i and E_i is the set of edges at step i, we have that $\#(E_i) = \#(V_i) - 1$ for $i = 0, 1, \dots, 7$. In other words, at each step, our subgraph (V_i, E_i) is a tree and by construction it is connected. When $V_i = V$, i.e. for $i = 7$, (V, E_7) is a spanning tree of the original (V, E).

<u>NB</u> Procedure 1 and procedure 2 yielded <u>DIFFERENT</u> spanning trees of (V, E) as we had lots of choices as to which edges to delete or add respectively. We thus see that a spanning tree of a connected graph is <u>not unique</u> unless of course, the original graph is itself a tree.

$\boxed{\text{Kruskal's algorithm}}$

<u>Task</u> If each edge of a connected graph (V, E) comes w/ a particular cost, describe an algorithm that finds the spanning tree of (V, E) w/ minimal cost.

<u>Def</u> Let (V, E) be an undirected graph. A <u>cost function</u> $c: E \to \mathbb{R}$ on the set E of edges of the graph is a function that assigns to each edge e of the graph a real number $c(e)$.

Let $c: E \to \mathbb{R}$ be a cost function on the set E of edges of a graph (V, E). Given any subset $S \subset E$, we define the cost on S to be $c(S) = \sum_{e \in S} c(e)$, the sum of the costs of all elements of S.

<u>Def</u> Let (V,E) be a connected graph w/ cost function $c: E \to \mathbb{R}$. A spanning tree (V, E_M) is said to be <u>minimal</u> (with respect to the cost function) if $\forall\, (V, E_T)$ a spanning tree of (V,E) $c(E_M) \leq c(E_T)$.

<u>Kruskal's Algorithm for finding minimal spanning trees:</u>

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 <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. 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.