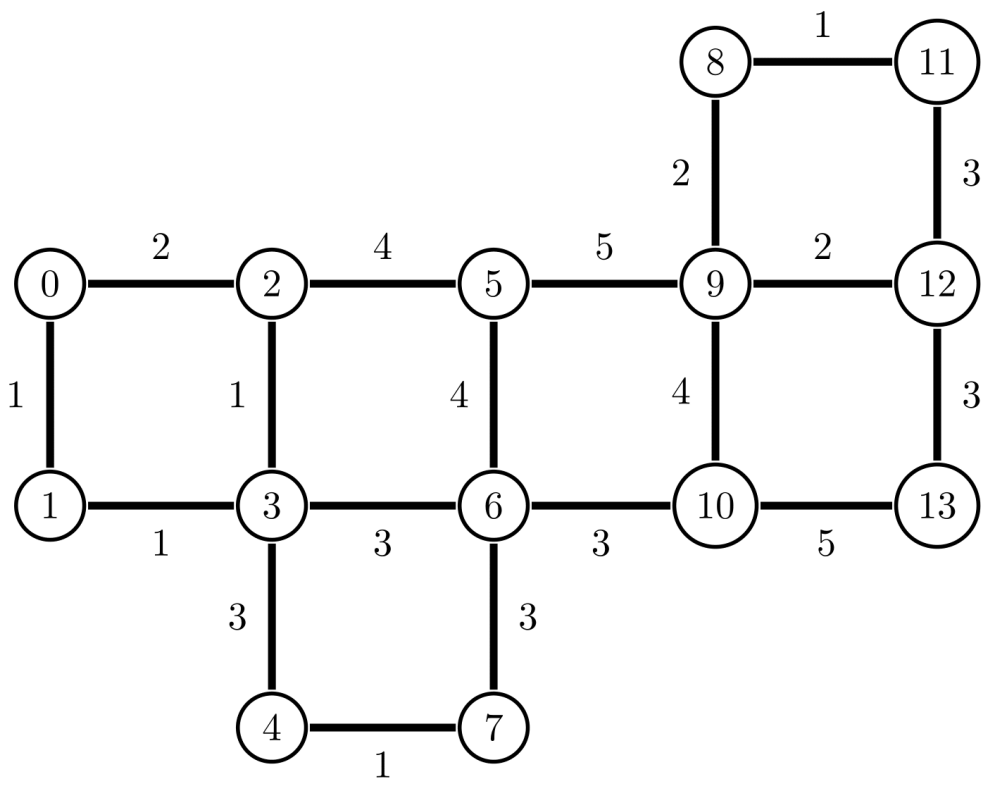
Theory: Kruskal's algorithm

17 minutes 8 / 8 problems solved

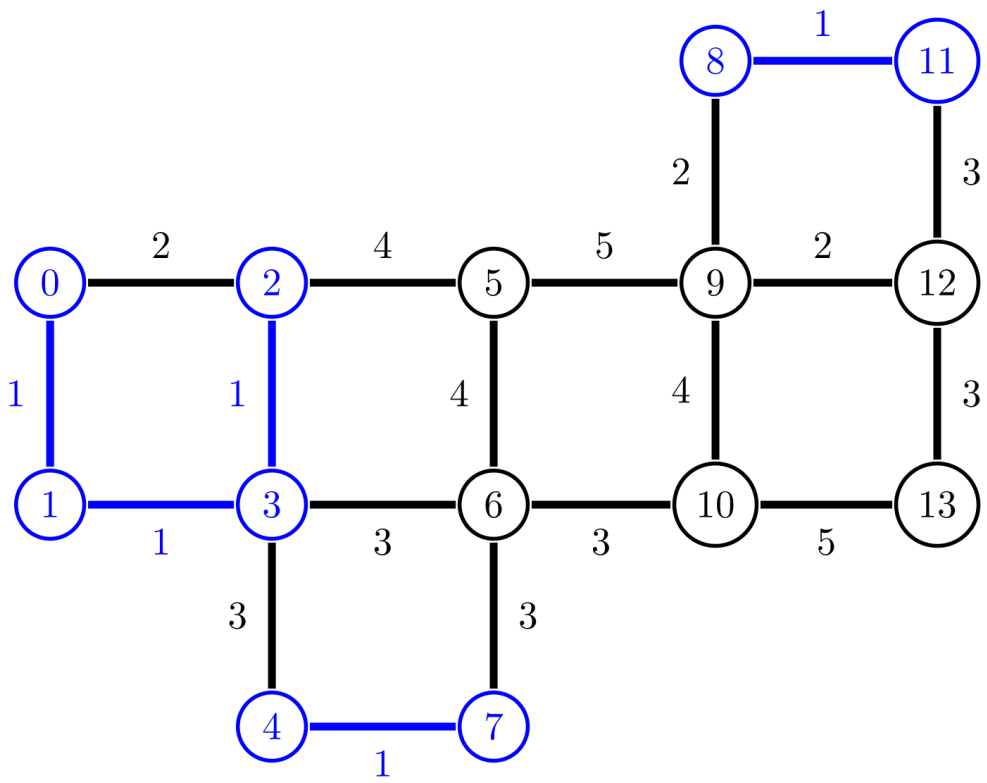
Kruskal's algorithm allows finding the **minimum spanning tree** in a connected undirected weighted graph. The algorithm starts with an empty spanning tree and repeatedly adds an edge with the smallest weight that does not produce a cycle in the current tree. So, it is a **greedy algorithm:** on each step, an edge that gives the best possible advantage is selected.

§1. An example

Let's see how the algorithm works for the following graph:



The weights of the edges range from 1 to 5, so we will start considering the ones whose weight is 1. There are 5 such edges: $\{0, 1\}, \{1, 3\}, \{2, 3\}, \{4, 7\}$ and $\{8, 11\}$. If we start repeatedly including these edges to the current spanning tree, none will produce a cycle. So, all of them have to be added to the spanning tree:



Next, we need to consider all the edges whose weight is 2. These are $\{0, 2\}, \{8, 9\}$and $\{9, 12\}$If we include the edge $\{0, 2\}$ to the current spanning tree, it will produce a cycle, therefore it cannot be used. The remaining edge won't give a cycle, so we will add them to the current spanning tree:

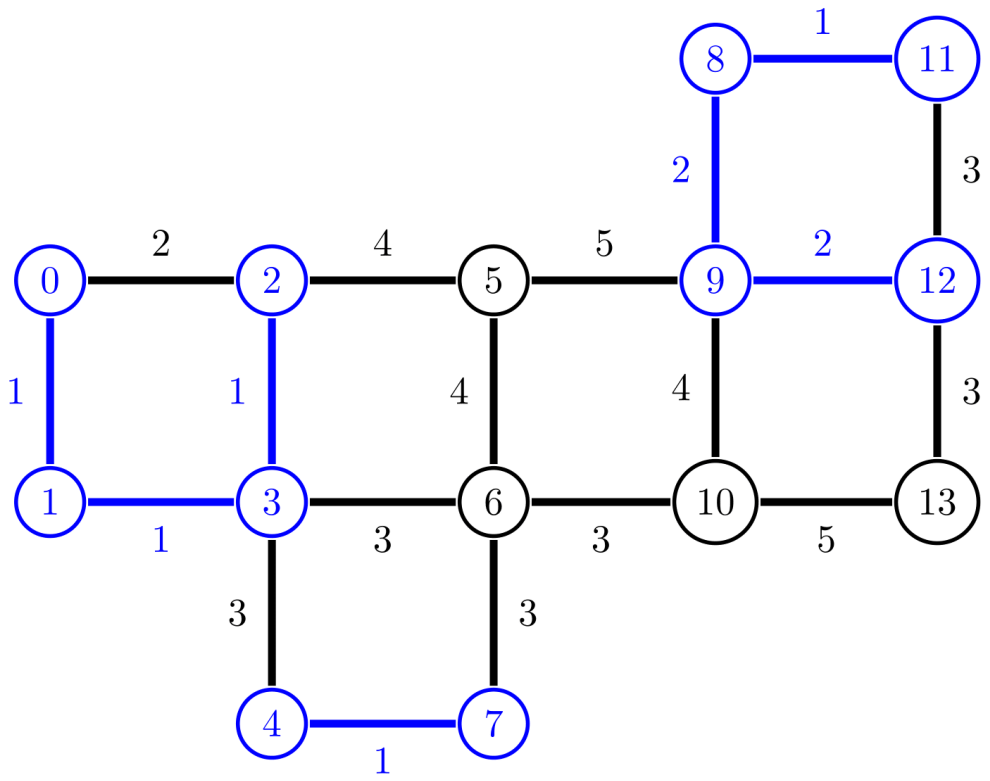
Current topic:

- ✓ Kruskal's algorithm ···

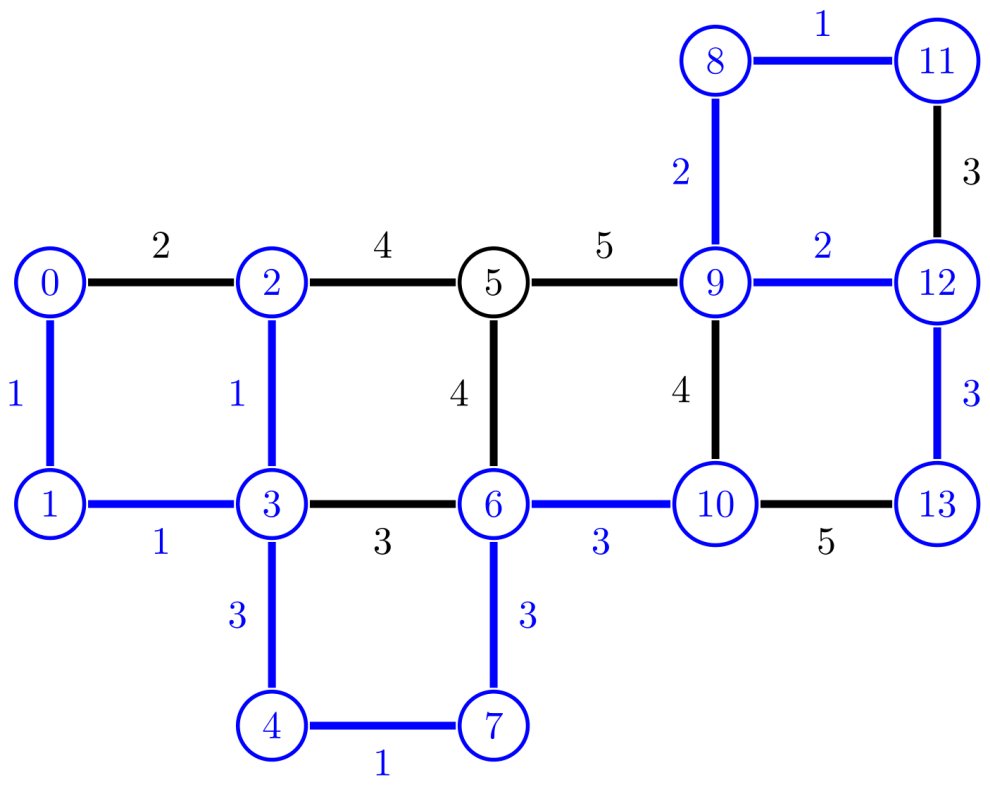
Topic depends on:

- ✓ Spanning trees ···
- ✓ The sorting problem ···

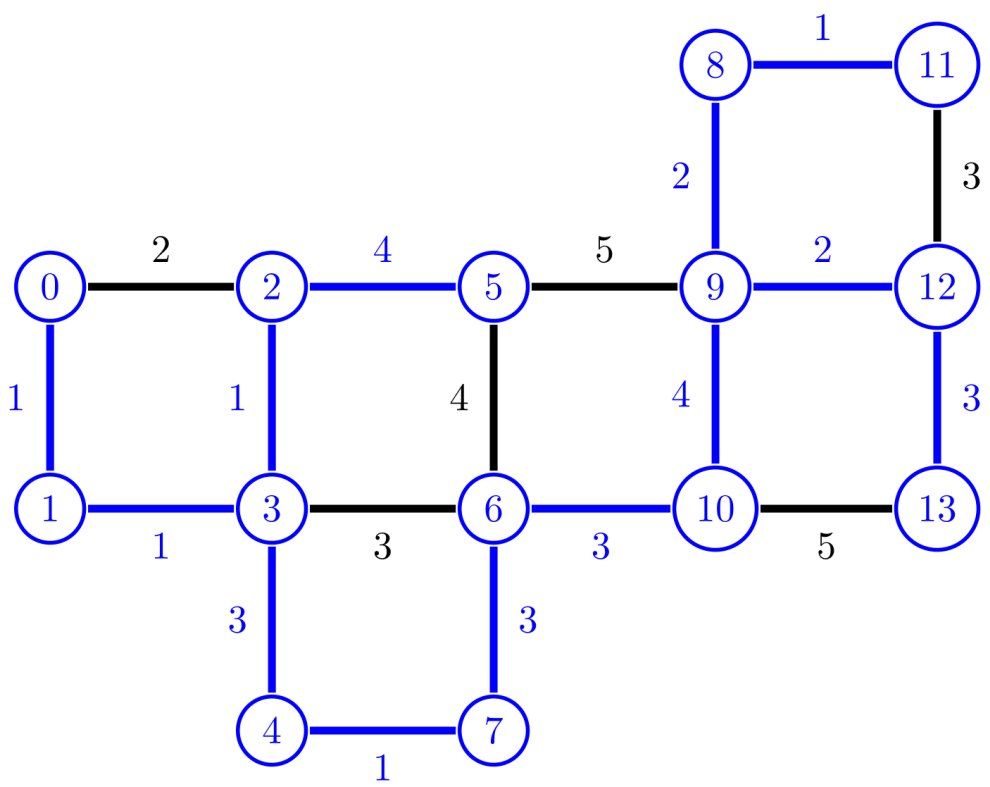
Table of contents:



Then, we will try to add the edges with the weight of 3 in the following order: $\{3, 4\}, \{6, 7\}, \{3, 6\}, \{6, 10\}, \{12, 13\}$ and $\{11, 12\}$. The first two edges can be added to the current spanning tree. But after that, the edge $\{3, 6\}$ cannot be used, since it produces a cycle. Then the edges $\{6, 10\}$ and $\{12, 13\}$ can be successfully added too, but the edge $\{11, 12\}$ cannot, because it gives a cycle. After all these edges are processed, we will get the following:



The edges with the weight of 4 are $\{2, 5\}, \{5, 6\}$ and $\{9, 10\}$. The first edge can be added to the current spanning tree since it does not produce a cycle. The second one cannot be used because it gives a cycle $2 - 5 - 6 - 7 - 4 - 3 - 2$. The last one does not give a cycle, so we will add it to the tree. Finally, we will get the following:



Since there are no edges that can be added to the current spanning tree, we are done. The minimum spanning tree consists of 13 edges and has the weight $1+1+1+1+1+2+2+3+3+3+3+4+4=29$.

§2. Summary

To sum, Kruskal's algorithm constructs a minimum spanning tree in a connected undirected weighted graph using a greedy approach. The steps of the algorithm are:

1. Sort all the edges of a graph by weight in increasing order.
2. Start with an empty spanning tree.
3. Successively consider the sorted edges. If the current edge does not produce a cycle, add it to the spanning tree. Otherwise, move to the next edge.
4. Repeat step 3 until all the edges are processed. Then return the resulting minimum spanning tree.

§3. The complexity analysis

Assume that a graph G has n nodes and m edges. Then, to find a minimum spanning tree in G, we first need to sort all the edges of G, which requires $O(mlogm)$ operations. Next, starting with an empty spanning tree, we need to consider the sorted edges and repeatedly add one that does not produce a cycle in the current tree. For the example above, we checked whether an edge produces a cycle by simply looking at the graph. But when implementing the algorithm using a programming language, one needs to use some data structure for it. Now we need to go to slightly more implementation details to complete the analysis.

To check whether adding an edge produces a cycle, we may store an array A of size n where $A[i]$ is the id of a component the node i belongs to. Initially, all the nodes belong to different components (for example, $A[i]=i$). When we want to add an edge $\{i,j\}$ to the current spanning tree, we may check whether the nodes are in different components by comparing their ids $A[i]$ and $A[j]$. After adding a new edge, we need to update the array A. For example, if $A[i]=a$ and $A[j]=b$ we may iterate through the array and substitute all b by a thereby uniting components.

Since any spanning tree in G contains $n-1$ edges, the total number of updates is no more than $n\cdot(n-1)=O(n^2)$ and therefore the total running time is $O(mlogm+n^2)$. However, to store and unite components, one may use a more sophisticated data structure called a disjoint set, which can decrease the running time to $O(mlogm)$.

Report a typo

28 users liked this theory. **2** didn't like it. **What about you?**

Start practicing

Comments (3) Hints (0) Useful links (0) Show discussion