Parallel algorithms: adapting for multiple processors.

Most of the algorithms we considered in this course were sequential: there was just one processor doing the job step-by-step. However, there is a lot of demand for parallel and distributed algorithms that can be executed in parallel on multiple processors/computers. Here we will briefly see a few examples of how to adapt a sequential algorithm to this setting.

Merge sort: Suppose you have 4 processors. The algorithm starts by splitting work into two and then two more pieces. Think of one processor doing the first split (and the following merge) and distributing work to two others, which again do the split and distribute the work among all four. Now, the four processors can sort the quarters of the input in parallel, and then merge the answers.

Borøuvka's algorithm: Borøuvka's algorithm starts by putting each vertex in a separate connected component (cluster) and then adds the best edge out of every cluster. This can be done in parallel among clusters.

Kruskal's algorithm: Provided edges are sorted, an edge e_i is in the minimal spanning tree iff it does not form a cycle in a graph consisting of the first $i-1$ edges (note that we don't consider the minimal spanning tree on the first $i-1$ edges, just the graph itself, which is faster). So for every i checking if e_i is in the min. spanning tree can be done in parallel.