3.1 Algorithm complexity

Consider two alternative algorithms A and B for solving a given problem. Suppose A is $O(n^2)$ and B is $O(2^n)$, where n is the size of the instance. Let n_0^A be the size of the largest instance that can be solved in one hour with algorithm A on a given computer, and n_0^B be the corresponding size for algorithm B. Denote by n^A e n^B the size of the largest instances that can be solved in one hour on a computer that is 100 times faster. How large is n^A with respect to n_0^A and n^B with respect to n_0^B?

3.2 Size of a problem instance

Determine the size of an instance of the minimum spanning tree problem.

3.3 Complexity of longest path problem and shortest simple path problem

A *simple path* in a directed graph is a path that visits each intermediate node at most once. Consider the MAX-SIMPLEPATH problem: Given a directed graph $G = (N, A)$ with a rational length associated to each arc, and a pair of nodes s and t, find a simple path of maximum total length from s to t. Show that this problem is $\mathcal{NP}$-hard.

To do so, show that the following recognition version of MAX-SIMPLEPATH is $\mathcal{NP}$-complete.

> MAX-SIMPLEPATH-r: Given a directed graph $G = (N, A)$ with a rational length associated to each arc, a pair of nodes s, t, and an integer K, does there exist a *simple path* from s to t of length at least K?

[*Hint*: Consider the HAMILTONIAN-CIRCUIT-r problem where, given a directed graph, we have to decide whether it contains a Hamiltonian circuit. Describe a polynomial-time transformation from HAMILTONIAN-CIRCUIT-r to MAX-SIMPLEPATH-r.]

Now consider the MIN-SIMPLEPATH problem: Given a directed graph with rational arc costs, and a pair of nodes s, t, we look for a shortest simple path from s to t. Why is it also $\mathcal{NP}$-hard?

3.4 Minimum spanning tree problem: complexity and ILP formulation size

Give an Integer Linear Programming formulation for the problem of finding a minimum cost spanning tree in an undirected graph $G = (V, E)$. How does the number of constraints grow with the number of the nodes $n = |V|$? Is there a direct relationship between the size of an ILP formulation (number of constraints and variables) and the difficulty of the corresponding problem?