Computability of the problem. Decision and optimization problems - dependencies

- A problem is said to be combinatorial is it has a finite solution space
- Decision problems are a kind of yes or no problems — the objective is to determine if the input satisfies a particular constrain. An example of that kind of problem is Hamiltonian Path.
- Optimization problems are designed to find the best solutions among a set of feasible solutions. They try to minimize/maximize a certain objective function. An example of that kind of problem is the traveling salesman problem or the knapsack problem.

Reducing an optimization problem into a decision problem and vice versa

- Decision problems and optimization problems might seems very different from each other, however in many cases it is actually possible to reduce an optimization problem into a decision problem and vice versa.
- Example: Traveling Salesman problem. An optimization version of this problem asks what is the shortest possible tour that visits each city exactly once and returns to the origin city? We can reduce this problem into a decision problem by asking: is there a tour that visits each city exactly once and has a total distance less than k? From there we can use binary search with complexity $\Theta(\log n)$ to find the minimum k for which the decision problem gives the answer yes. Since the binary search has logarithmic complexity it is negligible compared to the complexity of the main problem.
- Some decision problem can also be reduced into optimization problems. Let's consider the partition problem: given a set of integers can it be partitioned into two subsets such that the sum of the elements in each subset is equal? An optimization problem is phrased as: find a subset of given list of integers

whose sum is as close as possible to the total sum divided by 2. If the result of the optimization problem gives a subset whose sum is exactly equal to the total sum over 2, we have solved the decision problem.

- The general procedure of reducing an optimization problem into a decision problem is as follows:
  - Given an optimization problem, determine the objective function it tries to minimize/maximize
  - Consider a question of whether there exist a solution for which the objective function is either equal or less or equal to a given value k (depending on the problem)
  - Use binary search to minimize/maximize the given value k

Dependency between decision problems and optimization problems

- Difficulty of a decision problem influences the difficulty of the corresponding optimization problem (and vice versa)
- If we can reduce an optimization problem into a decision problem or a decision problem into an optimization problem the complexities of the corresponding problems are the same. Conversion of those problems only requires the use of binary search with logarithmic time complexity which is negligible.
- We can summarize the dependency of the complexity of corresponding decision/optimization problems as follows:
  - decision problem is difficult ⇒ optimization is difficult
  - optimization is easy ⇒ decision problem is easy

Computability of a problem

- **Computability** is the ability to solve a problem in an effective manner.
- There are various models of computability, such as Turing-computability, lambda calculus or combinatorial logic (afaik we will only focus on Turing).

- One goal of computability theory is to determine which problems, or classes of problems, can be solved in each model of computation.
- A problem that is not computable is called **undecidable**.
- Definition:

> 💡 The computability of a problem refers to whether there exists an algorithm that can provide a solution to the problem for any valid input in a finite amount of time. A problem is considered computable if such an algorithm exists; otherwise, it is deemed non-computable.