Algorithms and Data Structures Analysis (ADSA)

P and NP

Overview

- Complexity of Problems
- Classes P and NP

Efficient Algorithms

Major Questions:

- When do we call an algorithm efficient?
- Are there problems for which there is no efficient algorithm?

Efficient Algorithms

- An algorithm runs in polynomial time (is a polynomial time algorithm), if there is a polynomial p(n) such that its execution time on inputs of size n is O(p(n)).

- A problem can be solved in polynomial time if there is a polynomial time algorithm that solves it.

We call an algorithm efficient iff it runs in polynomial time.

Examples

Problems that can be solved in polynomial time:

- Integer Addition - O(n)
- Integer Multiplication – O(n^2)
- Test whether a graph is acyclic
- Shortest paths – Dijkstra O(m+n^2)
- Minimal spanning trees –Kruskal O(m log m).
- Almost all problems that we consider in this course.

Difficult Problems

There are many problems for which no efficient algorithm is known.

Examples (see Mehlhorn/Sanders page 54):

- Hamiltonian cycle problem
- Traveling Salesman Problem
- Boolean Satisfiability Problem
- Clique Problem
- Graph Coloring Problem

Hamiltonian Path Problem

- Given: Undirected graph G=(V,E).
- Decide whether G contains a Hamiltonian path. A Hamiltonian path is path that visits each node exactly once. (A spanning tree where each node has degree at most 2.)

Hamiltonian Cycle Problem

- Given: Undirected graph $G=(V,E)$.
- Decide whether it contains a Hamiltonian cycle. A Hamiltonian cycle is cycle that visits each node exactly once and returns to the start vertex.

Traveling Salesman Problem

- Given: Complete edge-weighted undirected graph G=(V,E) and an integer C.
- Decide whether G contains a Hamiltonian cycle of cost at most C.

Graph Coloring Problem

- Given: Undirected graph G=(V,E) and an integer k.
- Decide whether there is a coloring of the nodes with k color such that any two adjacent nodes are colored differently.

Multi-objective Minimum Spanning Trees

- Given: Undirected connected graph G=(V,E) with two weight functions w_1 and w_2 on the edges, and two numbers k_1 and k_2.
- Decide whether there is a spanning tree T of G for which

$$w_1(T) \leq k_1 \text{ and } w_2(T) \leq k_2$$

holds.

Boolean Satisfiability Problem (SAT)

- Given: A Boolean expression in conjunctive normal form.
- Decide whether it has a satisfying assignment.

Conjunctive normal form is conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_k$

Clause is disjunction of literals $l_1 \vee l_2 \vee \ldots \vee l_h$.

Literal is variable or a negated variable.

Formal setting

- Inputs are encoded in some fixed alphabet Σ.
- A decision problem is a subset $L \subseteq \Sigma^*$.
- Characteristic function χ_L of L.

$$\chi_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$$

Σ^* : Set of all possible strings over the alphabet Σ.

Class NP

A decision problem L is in NP iff there is a predicate Q(x,y) and a polynomial p such that

1. for any $x \in \Sigma^*$, $x \in L$ iff there is a $y \in \Sigma^*$ with $|y| \leq p(|x|)$ and $Q(x, y)$, and
2. Q is computable in polynomial time

y is a witness that x belongs to L (guess such a witness y). The predicate Q(x,y) is a function that returns true iff y is a witness that x belongs to L.

Verify y in polynomial time using Q.

Class NP

- How can we prove a problem is NP?
  - Guess an answer
  - Verify the answer true/false
  - If we can use polynomial time to verify the answer is true
  - Then the problem is NP.

Example: Class NP

The Hamiltonian Cycle Problem is in NP:

- We can guess a Hamiltonian cycle y in the input graph x.
- Given such a cycle y we can check in polynomial time whether it is a Hamiltonian cycle in x.

Class NP

- How can we prove a problem is NP?
  - A problem is not in NP if its solution cannot be verified in polynomial time.
- Any example?
  - All optimisation problems, whose answers cannot be checked in polynomial time.
  - TSP
  - Bin Packing
  - Timetabling

NP-Complete Problems

- We don’t know whether polynomial time algorithms exists for the mentioned problems.
- It is very likely (and almost all people in computer science believe) that there are no polynomial time algorithms for these problems.
- They belong to a class of equivalent problems known as NP-complete problems. (NP stands for “nondeterministic polynomial time”)

Class **P**

- A decision problem is polynomial solvable iff its characteristic function is polynomial-time computable.
- We use **P** to denote the class of polynomial-time-solvable decision problems.

Class **P**

- How can we prove a problem is P?
  - Prove that the problem can be solved in polynomial time.
- Can you prove that the MST problem is in P?
  - Hint: Upperbound on Kruskal…