Algorithms, 2023 Fall, Homework 5
(Due: Oct 26)

October 16, 2023

required reading: section 6.10.

Problem 1 : (@) KT book pp 320. problem 9

Problem 2 : (@) KT book pp 324 problem 13

Problem 3 : (@) KT book pp 324 problem 14

Problem 4 : In the minimum coloring problem, we need to assign a color to each node of the given undirected graph, such that any two adjacent nodes are assigned different colors. The goal is to use as few colors as possible. Solve the minimum coloring problem on graphs with bounded treewidth (i.e., the treewidth of the graph is $O(1)$) in polynomial time. You can assume a tree decomposition with constant treewidth is given.

Problem 5 : The definition of a chordal graph is the following (why the name is so called): A chordal graph is an undirected graph with the following property: every cycle of length four and greater has a chord (a chord is an edge that is not part of the cycle but joins two vertices of the cycle).

Prove the following statement. We are given a tree decomposition T of the graph G. If we make every bag in T a clique (a complete subgraph) by adding edges to G, the resulting graph is a chordal graph.

Problem 6 :(Minimum Dominating Set) Given a undirected graph $G(V, E)$, a subset S of vertices is a dominating set if for every $v \in V \setminus S$, there exists a vertex $u \in S$ which is a neighbor of v. The decision version of the minimum dominating set problem asks whether there is a dominating set of size at most k for a given undirected graph $G(V, E)$ and integer k. Design an efficient verifier for the problem.

Problem 7 : In formal complexity theory, we should understand a decision problem L (formally it is called a language) as an (infinite) collection of yes instances (the instances for which we should output yet). Hence, we can formally define *the complement* $\bar{L}$ of a language L, which consists of all problem instances not in L (or all no instances). Define the class co-NP to be all languages that are complement of some NP languages. For example, the problem of deciding whether the graph does not contain a vertex cover of size at most k is a co-NP problem.

It may appears that NP and co-NP are equivalent to each other since for any input instance we only need to change a yes answer to a no answer. But (people believe) there is fundamental difference between these two classes. For an NP problem L and a YES instance (a string $X \in L$), there exists a certificate (or "solution", the input t in $B(s,t)$ mentioned in the class) that we can verify in poly-time. For example, we verify whether an assignment satisfies a 3CNF. However, for a co-NP problem $\bar{L}$ and a string $X \in \bar{L}$), it is unclear such a certificate exists. For example, how can you quickly verify whether the graph does not contain a vertex cover of size at most k (without trying all possibilities). Make sure you understand the difference.

1. Prove P$\subseteq$ NP $\cap$ co-NP.

2. Define PRIME as the problem of deciding whether a positive number is a prime. The problem is known in P (through a celebrated algorithm called AKS primality test). Hence PRIME is in NP $\cap$ co-NP. (note that in this problem, poly-time means poly in the length of the input, encoded in binary bits).

   Now consider the FACTOR problem, which generalize the above problem. The input is of the form (x, r) and the goal is to decide whether the given number x has a prime factor that is smaller than r. Show FACTOR is in NP $\cap$ co-NP.