Johan Karlander, KTH, CSC

Theory exam in i Algorithms (Data Structures)and Complexity, DD2352/DD1352 2013-05-20, 10.00–13.00

No aids are allowed.

For students taking course DD2352 12 points are required for grade E, 15 points for grade D and 18 points for grade C.
You can have up to 4 bonus points.

For students taking course DD1352 14 points are required for grade E, 17 points for grade D and 20 points for grade C.
You can have up to 8 bonus points.

The solutions can be written in Swedish.

1. (8 p)

   Are these statements true or false? For each sub-task a correct answer gives 1 point and an answer with convincing justification gives 2 points.

   a. An algorithm with time complexity $O(\log n!)$ runs in polynomial time.
   b. The problem K-coloring of graphs is in NP.
   c. In a graph G with n nodes and m edges we always have $m \in \Theta(n^2)$.
   d. Kruskal's algorithm is an example of a decomposition algorithm.

   Solution:

   a. TRUE A well-known estimate gives us $\log n! \sim n \log n$. This means for instance that $O(\log n!) \in O(n^2)$.
   b. TRUE A certificate is just a K-coloring of the graph and by checking each edge we can tell if the coloring is correct or not. This can be done in polynomial time.
   c. FALSE $m \in \Theta(n^2)$ means roughly that m is of the same size as n^2.This is obviously false for some graphs. For instance, we could have $m = 0$.
   d. FALSE Kruskal's algorithm is an example of a Greedy Algorithm.

2. (3 p)

   If we have an undirected graph G with positive edge weights, a node s and a tree T in G we can say that T is a *shortest path tree* based at s if T contains all nodes and if, for every node t, the unique path from s to t is a shortest path between s and t in the graph.

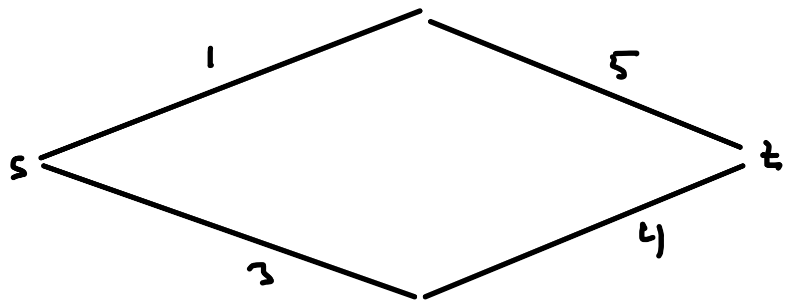
Here is an algorithm:

```
F($V, E, w, s$)
(1)    $S \leftarrow \{s\}$
(2)    $T \leftarrow \emptyset$
(3)    while $S \neq V$
(4)        Let $A$ be the set of edges $(u, v)$ such that $u \in S$ and $v \notin S$
(5)        Let $(x, y)$ be a smallest edge in $A$
(6)        $S \leftarrow S \cup \{x\}$, $T \leftarrow T \cup \{(x, y)\}$
(7)    return $T$
```

a. Show that this algorithm always returns a tree.
b. Show, by giving a counter example, that this algorithm doesn't always return a shortest path tree.
c. There is another famous problem the algorithm actually solves. What problem?

Solution:

a. We see that at the start of the algorithm, the sets S and T always form a connected subgraph that contains no cycles, i.e. they form a tree. This property is constant while running the loop. Indeed, the edge (x, y) connects y to a node in S and (x, y) cannot form a cycle with any of the edges in T.
b. We could take the following graph:



c. It is an algorithm for finding minimal spanning trees. It is in fact Prim's algorithm.

3. (3 p)

In the graph bellow the nodes are problems. An arrow like $A \rightarrow B$ indicates that there is a polynomial time reduction (Karp-reduction) from A to B. Let us assume that A is NP-Complete. What do we then know about the rest of the problems? Answer the following:

a. Which problems must be NP-Complete?

b. Which problems must be in NP?

c. *If* we would know that P $\neq$ NP, which problems could then possibly be in P?

Solution:

a. The principles we can use is that if $A \rightarrow B$, then if A is NP-Hard then B must be as well and if B is in NP then A must also be in NP. This gives us that A, D and E must be NP-Complete.

b. The same principle show that A, B, C, D and E must be in NP.

c. The problems that could not be in P are the problems such that we can form a chain of reductions from A to the problem. This means that the rest, i.e. B and C could be in P.

4. (3 p)

A propositional logical formula ϕ is on Disjunctive Normal Form (DNF) if

$\phi = c_1 \vee c_2 \vee ... \vee c_k$ where each clause c_i is a variable or a negated variable or a conjunction ($\wedge$) of several variables and/or negated variables. An example is

$$(x_1 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge x_2 \wedge x_4) \vee \bar{x}_3 \vee (x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_6)$$

.

Describe an efficient algorithm that decides if a formula on DNF is satisfiable or not. You don't have to write code. Just give an informal description of an algorithm and explain why it is efficient.

It is well known that any propositional logical formula can be converted to DNF. It seems that we could decide efficiently if a formula ϕ is satisfiable or not by first converting it to DNF and then run your algorithm. Explain what is wrong with this argument.

Solution:

All you have to do is to check the clauses and see if there is an occurrence of a variable x_i and its negate $\bar{x}_i$. If there is a clause without any such occurrences then the formula is satisfiable, otherwise it is not. Obviously this check can be done in polynomial time.

But the reason why we cannot get an efficient algorithm is that we cannot generally transform a formula to DNF in polynomial time.

5. (3 p)

We now study an optimization problem defined like this:

Input: A list of positive integers $p_1, p_2, ..., p_n$.

Goal: Partition the numbers into two set S_1 and S_2 such that if $Sum(S_i)$ is the sum of the elements in S_i, the $Max(Sum(S_1), Sum(S_2))$ is as small as possible.

a. What important NP-Complete problem corresponds to this optimization problem?

b. We define an algorithm for solving the problem:
   Start with S_1 and S_2 empty. Place p_1 in S_1. From now on $p_2, p_3, ..., p_n$ in turn are placed in the S_i with $Sum(S_i)$ smallest (or in S_1 if the sums are equal.).

   Show by an example that this algorithm doesn't always return an optimal solution.

c. Show that the algorithm approximates the problem within $B = 2$.

Solution:

a. The PARTITIONING problem.

b. A very simple counterexample is $1, 2, 3$. The algorithm places them into $S_1 = \{1, 3\}, S_2 = \{2\}$, but the optimal solution is $S_1 = \{1, 2\}, S_2 = \{3\}$.

c. Obviously, if $P = p_1 + p_2 + ... + p_n$, we have $\frac{P}{2} \leq OPT$ and $APP \leq P$. So we get $\frac{APP}{OPT} \leq \frac{P}{\frac{P}{2}} = 2$. So the algorithm approximates within $B = 2$.