Johan Karlander, KTH, CSC

Teoritenta i Algoritmer (datastrukturer) och komplexitet för KTH DD1352–2352 2016-05-30, klockan 14.00–17.00 Solutions

No aids are allowed. 12 points are required for grade E, 15 points for grade D and 18 points for grade C.

If you have done the labs you can get up to 4 bonus points. If you have got bonus points, please indicate it in your solutions.

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. The problem of findning a minimal spanning tree in a weighted graph can be solved in time $O(n^3)$ where n is the number of vertices in the graph.

      TRUE. An algorithm like Kruskal's algorithm works in time $O(E \log E)$. Since $E \in O(n^2)$ and $|logE \in n$, we get $O(E \log E) \in O(n^3)$.

   b. If a Divide and Conquer-algorithm has a time complexity $T(n)$ given by the recursion formula

      $$T(n) = 2T(\frac{n}{2}) + cn$$

      then $T(n) \in O(n^2)$.

      TRUE. The Master Theorem gives us $T(n) \in O(n?ogn) \in O(n^2)$.

   c. The problem of deciding if a program P halts on every input is decidable.

      FALSE. We can reduce the basic undecidable problem HALTING can be reduced to this problem, which shows that the problem is undecidable.

   d. If an NPO-problem can be approximated within a factor 2, it can also be approximated within a factor 3.

      TRUE. If we have a maximization problem that is approximable within a factor 2, we have $\frac{OPT}{APP} < 2$. Then we also have $\frac{OPT}{APP} < 3$. The reasoning for minimization problems is similar.

2. (3 p)

   Describe in detail how Ford-Fulkerson's Max Flow Min Cut -algorithm works. Let us assume that all capacities are positive integers. Explain why the algorithm then returns a flow with integer values for the flow in each edge.

Solution: For a description of the algorithm, see the book or the lecture notes. If all capacities are positive integers we can use induction. If all flows in all edges are integers we see that the value δ (as defined by the algorithm) for an augmenting path is an integer. The increase in the flow is then an integer too. So at each step we get a flow with integers in each edge.

3. (3 p)

   Let us assume that we have an undirected, connected graph G. If we have a path P in G, we say that the length of the path is the number of edges in the path. We will consider the problem of finding *longest* paths between pair of nodes. Given two nodes a and b, we define $L(a, b)$ to be the longest lenght of paths going from a to b. Let us know consider the following recursion formula:

$$\begin{cases} L(x,x) = 0 \text{ for all } x \\ L(x,y) = \underset{i}{max}\,(L(x,i)+1) \text{ where } i \text{ runs over all vertices adjacent to } y \end{cases}$$

   Does this recursion formula work (is it correct)? Give proof or counter-example.

   What happens if we replace "longest path" with "shortest path" and replace "*max*" with "*min*" in the recursion formula. Does this recursion formula work? Give proof or counter-example.

   Solutions: The formula for longest paths is not correct. We can take a very simple example: Take a triangel with vertices a, b, c. We see for instance that $L(a, b) = L(b, c) = 2$ (correct value). But then the formula says that $L(a, c) = L(a, b) + L(b, c) = 2 + 2 = 4$ which is incorrect.

   The formula for shortest paths is correct. There is a shortest path from x to y. On this path there must be a last vertex before y. Let us call it z. ($x = z$ is possible.) But then the path from x to z must be a shortest path too. By induction we get that it is $L(x, z)$. So $L(x, y) = L(x, z) + 1$ where z is chosen so that the righthand side of the equation is minimal.

4. (3 p)

   In the graph bellow the nodes are problems. An arrow like $A \to B$ indicates that there is a polynomial time reduction fram A to B. Observe that there could be more reductions than those indicated. Let us assume that A is NP-Complete. Answer these questions:

   a. Which problems must be NP-Complete?
   b. Which problems could be outside NP?
   c. Given P $\neq$ NP, which problems could then be in P?

   Solutions:

   a. Which problems must be NP-Complete? : A,B,C.

b. Which problems could be outside NP?: E, F.

c. Given P $\neq$ NP, which problems could then be in P?: D, E.

5. (3p We want to approximate the problem PARTITIONING. In order to do that we first formulate an optimization variant of the problem. Let $x_1, x_2, ..., x_n$ (positive integers) be an instance of the problem. Let us assume that we have partitioned the set into two parts with sums S_1, S_2. (A solution to the decision problem has $S_1 = S_2$.) Let $M = Max\,(S_1, S_2)$. Our optimization problem consists of minimizing M.

   We use the following approximation algorithm: We start with assigning x_1 to the first set. After that, we assign $x_2, x_3, ..., x_n$ (in that order) to the set with the temporarily least sum.

   Show that this algorithm approximates the optimization problem within a factor 2. Give an example of an instance with $APP > OPT$.

   Solution: A proof of this is given in the course book ch 11.1 (or rather a more general case of the problem) and in lecture notes to lecture 11.

   A very simple example is the sequence $2, 1, 3$ where the algorithm generates two sets with sums $2, 4$ and $M = 4$. The optimal solution is of course sets with sums $3, 3$ and $M = 3$.