Johan Karlander, KTH, CSC

Teoritenta i Algoritmer (datastrukturer) och komplexitet
för KTH DD1352–2352 2015-06-01, 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.

In all solutions you can assume that $P \neq NP$.

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. There are known efficient algorithms fore deciding if a graph is 2-colorable.

      TRUE. For instance, it can be shown that a graph is 2-colorable if and only if it is bipartite. There is a modified version of BFS decides in polynomial time if a graph is bipartite or not.

   b. It is possible to find the product of two n-degree polynomials in time $O(n \log n)$.

      TRUE. You can use FFT. Transform the two polynomials, multiply their transforms and make an inverse transformation. Can be done in time $O(n \log n)$.

   c. Let us assume that Divide and Conquer-algorithm has a time complexity $T(n)$ given by the equation

      $$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

      Then the time complexity is smaller than $\Theta(n^2)$.

      TRUE. We can use the master theorem with $a = 3$ and $b = 2$. We get $1 < k < 2$. That gives us $T(n) \in \Theta(n^k)$.

   d. Every problem in PSPACE is also a problem in P.

      FALSE. It can be shown that NP $\subseteq$ PSPACE. If all problems in PSPACE would belong to P we would have P = NP. (And we assume it is not.)

2. (3 p)

   Describe in detail how the Depth First Search algorithm works. A graph can be represented in different ways. Show what the difference in time complexity is when you run

the algorithm with a graph represented with an adjacency matrix and with a graph represented with adjacency lists.

Solution: For a description of DFS, see lecture notes or the course book. If we use adjacency list we see that the time complexity is $O(|E|)$. If we use an adjacency matrix, we find that the time complexity is proportional to the number of times we have to check for neighbors to nodes. This number is $O(|V|^2)$. When we have *sparse* graphs, we see that adjacency lists work better.

3. (3 p)

   Let G be a directed graph with no directed cycles (i.e. a DAG). Let us assume that we have a topological numbering of the nodes. We have a weight w_i for each node i. If there is a path from node i to node j (this path is then unique) we say that the node-weight of the path is the sum of the weights of all nodes in the path, including node i and node j. Describe a Dynamical Programming algorithm for finding the node-weight of the path between node i and node j (if there is any). Give the time-complexity for your algorithm.

   Solution: There are several possible solutions. One is to define $d[a, b]$ = node-weight of the path from a to b. If there is no such path we set da, b. We can compute these values recursivelly. We initialize $d[a, b] = \infty$ for all a, b. Then set $d[a, a] = w_a$ for all a. Then, for all $a < b$ we check for $a \leq k < b$ such that $d[a, k] < \infty$ and there is an edge (k, b) in G. If there is such a k we set $d[a, b] = d[a, k] + w_b$. In the simplest implementation this algorithm has time-complexity $O(n^3)$.

4. (3p)

   In the graph bellow the nodes are problems. An array like $A \rightarrow B$ indicates that there is a polynomial time reduction fram A to B. Observe that there could be more reductions than those given. 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?

Solution:

We use the facts:

1. If $U \to V$ and V is in NP, then U is in NP.
2. If U is NP-Hard, then V is NP-Hard.
3. I V is in P, then U is in P.

a. In NP: A, B, C.

b. Possibly outside NP: D, E, F, H.

c. Possibly in P: D, G.

5. (3 p)

The two problems VERTEX COVER and INDEPENDENT SET are related in the sense that $A \subseteq V(G)$ is a vertex cover if and only if $V(G) - A$ is an independent set. We know that VERTEX COVER can be approximated. The question here is if INDEPENDENT SET also can be approximated.

a. Describe the approximation algorithm for VERTEX COVER (i.e. the one given in the course).

This algorithm has an approximation quotient $B = 2$. Now let A' be a vertex cover given by the algorithm. Let $C' = V(G) - A'$. Then we will take C' as an approximative solution to INDEPENDENT SET.

b. Can you find an upper bound for $\frac{OPT_{IS}}{|C'|}$, that is, can you find an approximation quotient for the suggested algorithm?

Solution: In fact, there is no approximation quotient. To see this, take any integer $n > 2$. Form a graph G with nodes $\{0, 1, 2, ..., 2n\}$. Let there be edges $\{(0, 1), (0, 2), ..., (0, 2n), (1, 2), (3, 4), ..., (2n - 1, 2n)\}$. It can then be seen that one possible solution given by the vertex cover algorithm is the set $\{1, 2, 3, 4, ..., 2n\}$. This gives the one node 0 as an approximation for a maximal independent set. But we can easily see that the size of a maximal independent set is n. So we get a quotient n. Since n can be chosen arbitrarily large, there is no approximation quotient.