Johan Karlander, KTH, CSC

Teoritenta i Algoritmer (datastrukturer) och komplexitet
för KTH DD1352–2352 2014-06-05, klockan 9.00–12.00

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. The problem of deciding if a graph is connected is an NP-Complete problem.

      FALSE. The problem of deciding if a graph is connected is in P and we assume that $P \neq NP$.

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

      $$T(n) = 2T(\frac{n}{4}) + cn$$

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

      TRUE. We can use the Master Theorem. We have $\log_b a = \frac{1}{2}$ and the third case gives us $T(n) \in \Theta(f(n)) = \Theta(n)$, and this also means $T(n) \in O(n)$.

   c. If a problem A can be reduced to a problem B in polynomial time and we know that B is NP-hard, then we know that A is NP-Complete.

      FALSE. For instance, we can take any problem A in P (so A is not NP-Complete) and reduce to SAT (which is NP-hard).

   d. The problem of deciding if a Turing machine M halts on all inputs is an undecidable problem.

      TRUE. It is possible to reduce this problem to the Halting Problem which is known to be undecidable.

2. (3 p)

As you know, Dijkstra's algorithm is an algorithm for finding shortest paths in weighted directed graphs. It works for graphs with positive weights, but not always for graphs with negative weights. Explain how Dijkstra's algorithm works and then explain why it can not handle negative weights.

Solution: See lecture notes or the course book for a description of Dijkstra's algorithm. A simple example that shows that the algorithm can fail for negative weights is this: Take $V(G) = \{1, 2, 3\}$ and $E(G) = \{(1, 2), (1, 3), (2, 3)\}$, $w(1, 2) = -100, w(1, 3) = 2, w(2, 3) = 101$ and $s = 1$.

3. (3 p)

From the lectures you know that the max Max Flow Algorithm can be used to solve the matching problem. Now we look at a similar problem:

Let G be a directed graph, and $A \subseteq V(G), B \subseteq V(G)$ two sets of nodes such that $A \cap B = \emptyset$. We want to find a maximum size set M of edge-disjoint paths from A to B. This means that all paths in M starts in a node in A and ends in a node in B. No paths in M have any common edges (but might have common nodes). Explain why this problem can be seen as a generalization of the matching problem. Show how we can use the Max Flow Algorithm to find such a set M.

Solution: If we demand that $A \cup B = V(G)$ we get the matching problem. So our problem is a generalization. To solve the problem we add to nodes s, t. We add directed edges from s to all nodes in A and edges from all nodes in B to t. These new edges are given capacity ∞, all old edges are given capacity 1. We find a max flow in the graph. The old edges with flow 1 gives us the paths in M.

4. (3 p)

We will here look at a suggested reduction from SUBSET SUM to PARTITIONING. Let $a_1, a_2, ..., a_n$ be positive integers and K a positive integer. We want to know if there is a subset sum equal to K. We construct an instance of PARTITIONING as $a_1, a_2, ..., a_n, X$. Decide what X should be. Prove that the reduction is correct.

Solution: Let us assume that $a_1, a_2, ..., a_n, X$ can be partitioned. Then $a_1, a_2, ..., a_n$ can be partitioned into two sums S_1, S_2 such that $S_1 + S_2 = \sum_i a_i = A$ and $S_1 = S_2 + X$. A simple calculation shows that $S_1 = \frac{A+X}{2}$ and $S_2 = \frac{A-X}{2}$. Now if $K \leq \frac{A}{2}$ we set $X = A - 2K$, otherwise we set $X = 2K - A$. In the first case, we find that

$$\text{There is a subset sum } S_2 = K \Leftrightarrow \text{There is a partitioning } S_1 = S_2 + A - 2K$$

while in the second case we get

$$\text{There is a subset sum } S_1 = K \Leftrightarrow \text{There is a partitioning } S_1 = S_2 + 2K - A$$

5. (3 p)

This problem is about TSP (Traveling Salesperson Problem). we have a complete, undirected graph G with edge weights d_{ij} where d_{ij} is the weight (length) of the edge (i, j). We want to find a shortest Hamiltonian cycle in the graph.

We can actually solve this problem recursively. Let $S \subseteq \{1, , ..., n\} = V(G)$ such that $1 \in S$, let $i \in S$ and let $W(S, i)$ be the length of the shortest possible *path* visiting each node in S exactly once (and no nodes outside S), starting at 1 and ending in i.

We set $W(\{1\}, 1) = 0$ and $W(S, 1) = \infty$ for all $|S| > 1$. Show that for all $j \neq 1$ we have

$$W(S, j) = \min_{i \in S, i \neq j} W(S - \{j\}, i) + d_{ij}.$$

Show how we can use this recursion formula to find an algorithm that solves TSP. What is the time complexity of your algorithm if n is the input size. (Don't expect to find an *efficient* algorithm.)

Solution: Let us consider an optimal path giving length $W(S, j)$. There must be a $i \in S$ which is the last node before j in the path. The path $1 \to i$ must be inside $S - \{j\}$ and it must be optimal, i.e., of length $W(S - \{j\}, i)$. The total length of the path is $W(S - \{j\}, i) + d_{ij}$ and i must be chosen so that this expression is minimal. So we get $W(S, j) = \min_{i \in S, i \neq j} W(S - \{j\}, i) + d_{ij}$.

We will sketch a solution algorithm. Suppose we have computed $W(S, i)$ for all S and i. There are $O(n2^n)$ such values. The recursion formula shows that this can be done in time $O(n^2 2^n)$.

To find the length of the shortest Hamiltonian cycle we compute $\min_j (W(V, j) + d_{j1})$. The total time complexity is $O(n^2 2^n)$.