

Algorithms and Complexity

Exercise session 9

Repetition

Different exercises from old exams

This is not a standard exam template but just a collection of mixed exercises.

1. PSPACE is the complexity class consisting of all languages for which there exists a *deterministic* Turing machine that recognizes the language in polynomial *space* (memory). EXPTIME consists of all languages for which there exists a *deterministic* Turing machine that recognizes the language of exponential time. Show that $\text{PSPACE} \subseteq \text{EXPTIME}$.
2. (6p) [Classical first exercise in the theory part]
Are these statements true or false? For each sub-task provide a correct answer 1 point and proven correct answer 2 points;
 - a) The problem of determining whether a n -digit number is prime number, is in the complexity class co-NP.
 - b) There exists a constant $c > 1$ such that $n^3 \in O(c^{\log n})$.
 - c) Binary tree is usually implemented by inserting two pointers in each entry (**left** and **right**). When you implement ternary trees (where each node has three children), you can not use less than three pointers in each entry.

3. In a large organization such as KTH there are many groups of people such as teachers at NADA, teachers at F, students of the Algorithms and Complexity course, members of Teknologkoren, etc. Each individual is included in at least one group, but one can be in many groups. Now the president wants to create a group of representatives who can quickly disseminate information to all individuals at KTH. He wants each group to be represented in this group (ie at least one member of each group will be in the representative group), but he wants the representative group to be as small as possible.

This is an example of a general problem which, given a set of groups, finds the smallest group of representatives.

- a) (1p) Formulate the problem mathematically as a set problem and describe it at the same time as a decision problem.
 - b) (5p) Show that the problem is NP-complete.
4. (6p) Input to the optimization problem MAX k -CUT is a graph G . A solution is a cut of the vertices of G in k groups. The problem is to find a solution that maximizes the number of edges in the intersection between the groups, that is, between the vertices belonging to different groups. For $k = 2$ the problem is thus the same as MAX CUT.

Describe a probabilistic approximation algorithm for MAX k -CUT and analyze the expected value of objective function (ie the number of edges that go between the vertices belonging to different groups). Try to develop an algorithm where the expected value is at least $1 - 1/k$ of all edges, which means that approximation factor is $k/(k - 1)$.

5. a) (6p) MAX 2-SAT is an optimization problem which is defined as a decision problem as follows.

INPUT: A positive integer K between 1 and n and n clauses where each clause consists of one or two literals combined by the operator \wedge . Example: $x_1 \wedge \bar{x}_3$, $x_2 \wedge x_3$.

PROBLEM: Is there a variable assignment that satisfies at least K clauses?

Show that the decision version of MAX 2 \wedge SAT problem is NP-complete. You might reduce it to the MAX 2SAT - the equivalent problem with the operator \vee instead of \wedge - which is NP-complete.

- b) (5p) Construct a probabilistic approximation algorithm that approximates the optimization problem MAX 2 \wedge SAT within a factor of 4 (on average).

Solutions to exercises from old exams

1. If a Turing machine uses a polynomial amount of memory, there is a constant k such that the number of locations on the tape is $O(n^k)$ where n is the input data length. If the alphabet consists of three characters (0, 1, blank), the number of different possible configurations of the tape limited by $3^{O(n^k)}$, the number of possible locations for read/write head is $O(n^k)$ and the number of possible states of the Turing machine is finite, ie $O(1)$. The total number of configurations of a Turing machine using polynomial memory is thus $O(n^k) \cdot 3^{O(n^k)}$, ie exponential in n . Since the Turing machine can not return to the same configuration multiple times (then it would go into an infinite loop), this is also an upper bound of time. Thus, all problems that can be solved with polynomial memory can be solved in exponential time.
2. a) *True*. A problem lies in co-NP if its complementary problem is in NP. The complementary problem is in this case to determine if a number with n digits can be factored in at least two factors (greater than 1). This problem is in NP since a solution (ie, a factorization of the number) can be verified in polynomial time (by multiplying together factors and check that the product is the given number).
 b) *True*. if we assume that $\log n$ is the logarithm in base 2, we know that $c^{\log n} = 2^{\log c^{\log n}} = 2^{\log n \log c} = n^{\log c}$. If we choose $c \geq 8$ then $\log c \geq 3$ and $n^3 \in O(n^{\log c}) = O(c^{\log n})$.
 c) *False*. For general tree it is sufficient to have two pointers in each node (**firstson** and **next**).
3. a) Let k be a positive integer, S be the set of persons and $C = \{C_1, \dots, C_m\}$ be those m groups. The problem is to find a subset $S' \subseteq S$ with more than k elements such that $S' \cap C_i \neq \emptyset$ for $1 \leq i \leq m$. In English this problem is called *hitting set*.
 b) The problem is in NP since one can guess k elements that should be in the S' and verify that $S' \cap C_i \neq \emptyset$ for $1 \leq i \leq m$ in polynomial time.
 The problem is NP-hard as it is a generalization of the vertex cover problem which is known to be NP-complete. Given a graph $G = (V, E)$, let $S = V$ and $C = E$. A vertex cover of size k corresponds to a subset $S' \subseteq S$ of size k that contains at least one element from each C_i .
4. Let the algorithm for each of n vertices choose randomly which group it should belong to:

for $i \leftarrow 1$ **to** n **do** $group[i] \leftarrow random(1, k)$

The probability that the endpoints of a certain edge fall into the same group is $1/k$, as a vertex with probability $1/k$ falls in a particular group. The probability that the endpoints of a certain edge do not to fall into the same group is therefore $1 - 1/k$. If there are $|E|$ edges in the graph then the expected value of the number of edges that go between vertices in different groups is $|E|(1 - 1/k)$. Since a solution can never contain more than $|E|$ edges, the algorithm has approximation guarantee $1/(1 - 1/k) = k/(k - 1)$ on average. You can derandomize the algorithm to achieve an approximation factor $k/(k - 1)$ with a deterministic algorithm.

5. a) MAX 2 \wedge SAT is in NP because it is easy to verify that a variable assignment (a solution) satisfies at least K of the clauses. We show that it is NP-hard by reducing MAX 2-SAT.

Each 2-SAT clause with a single literal is also an instance of MAX 2 \wedge SAT. For each clause $l_i \vee l_j$ of a MAX 2-SAT instance we build three clauses of MAX 2 \wedge SAT instance: $l_i \wedge l_j$, $\bar{l}_i \wedge l_j$ and $l_i \wedge \bar{l}_j$. We can easily see that $l_i \vee l_j$ is true if and only if one of the three constructed clauses are true. If $l_i \vee l_j$ is false then all the three clauses are false. So the number of satisfying MAX 2-SAT clauses is exactly the number of satisfying MAX 2 \wedge SAT clauses. Therefore, choose the K for the constructed problem the same value as K for MAX 2-SAT problem and you are done.

- b) Let the algorithm randomly choose the values of each boolean variables with equal probability. What is the probability that a clause $l_i \wedge l_j$ satisfies this variable assignment? Well, since both literals must be true and the probability that one of them is true is $1/2$, a clause is satisfied with probability $(1/2) \cdot (1/2) = 1/4$. The expected number of satisfying clauses becomes $m/4$ if the number of clauses is m . Since the optimal solution satisfies at most m clauses, the approximation factor is at most 4, ie, the algorithm approximates MAX 2 \wedge SAT with a factor 4.