TECHNICAL UNIVERSITY OF DENMARK

Written exam, December 14, 2020

Course name: Computationally Hard Problems

Course No.: 02249

Aids allowed: all aids are allowed

Duration: 4 hours

Weighting: Exercise 1 – 10%, Exercise 2 – 15%, Exercise 3 – 10%,
Exercise 4 – 20%, Exercise 5 – 15%, Exercise 6 – 10%, Exercise 7 – 20%.

All exercises should be answered by filling out the boxes below the statement of the assignment or ticking boxes in the multiple-choice exercise. As your solution to the exam, just hand in the page with your answer and the following pages. If you need more space, you may use extra pieces of paper and hand these in along with your solution.

Name: ____________________________

Student id: ________________________

Exercise 1: Let $A = \{1, 2, \ldots, n\}$ for some positive integer n. A *set system* over A is a pair $(A, \mathcal{S})$, where $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a set of subsets of A, i. e., $S_j \subseteq A$ for $j = 1, 2, \ldots, m$. For example: $A = \{1, 2, 3, 4, 5\}$, $S_1 = \{1, 3\}$, $S_2 = \{1, 2, 4\}$, $S_3 = \{2, 3\}$, $S_4 = \emptyset$ (the empty set). The task is to design a formal language L_{set} for set systems.

a) Specify the alphabet Σ_{set} you use.

b) Specify how the language L_{set} is defined and describe how the above example is coded in your language.

c) Describe how one can check whether a given word $w \in \Sigma_{\text{set}}^*$ is a legal description of a set system and, if so, how the set system can be reconstructed.

Exercise 2: Consider the following problem:

Problem [LONGESTPATH]
Input: An undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$ for a positive integer n, and (only for the decision problem) a positive integer $k \leq n$.
Output: YES if there is a sequence $i_1, \dots, i_k$ of pairwise distinct (i. e. $i_a \neq i_b$ for $a \neq b$) integers from $\{1, \dots, n\}$ such that G contains a path from i_1 to i_k, i. e., $\{i_j, i_{j+1}\} \in E$ for $j = 1, \dots, k-1$. NO otherwise.
Output for the optimization version: A sequence $i_1, \dots, i_k$ of the above kind where k is as large as possible.

We would like to convert an algorithm A_d solving the decision version of LONGESTPATH into a polynomial-time algorithm A_o solving the optimization version of the problem.

a) Describe an algorithm A_o which solves the optimization problem as described above. The running time of A_o has to be polynomial in the input size and A_o may make calls to A_d. Recall that such a call counts as one step.

   We suggest that you divide the description of your algorithm into two subsequent phases.

   a1) Describe the first phase of your algorithm.

a2) Describe the second phase of your algorithm.

b) Argue that your algorithm is correct.

c) Show the polynomial running time of your algorithm.

Exercise 3: Consider the following problem.

> **Problem** [PartitionInto3Sets]
> **Input:** A sequence $X = x_1, x_2, \dots, x_{3n}$ of $3n$ positive integers, and a positive integer number B, such that $(B/4) < x_i < (B/2)$ for all $i \in \{1, 2, ..., 3n\}$ and $\sum_{i=1}^{3n} x_i = nB$.
> **Output:** YES if X can be partitioned into n disjoint sets $X_1, \dots, X_n$ such that for all $j \in \{1, \dots, n\}$ it holds that both $\sum_{x \in X_j} x = B$ and $|X_j| = 3$. NO otherwise.

Show that PartitionInto3Sets is in the class $\mathcal{NP}$. You need not show that the problem is $\mathcal{NP}$-complete. We suggest the following structure for your proof:

1) Design a deterministic algorithm $A(\boldsymbol{X}, R)$ (in the following just called A) which takes as input a problem instance $\boldsymbol{X}$ and a random sequence R. Especially:
   1a) Specify what the random sequence R consists of: bits, integers in some finite range etc.
   1b) Specify how A interprets R as a guess.
   1c) Specify how A verifies the guess.

2) Show that the following two conditions are met:

2a) If the answer to $\boldsymbol{X}$ is YES, then there is a string R_0 with positive probability such that $A(\boldsymbol{X}, R_0) = \text{YES}$.

2b) If the answer to $\boldsymbol{X}$ is NO, then $A(\boldsymbol{X}, R) = \text{NO}$ for all R.

3) Show that the running time of A is polynomially bounded.

Exercise 4: Consider the following problem.

Problem [LazyTSP]
Input: A positive integer n, a matrix $d(i,j)$ of non-negative integers, where $1 \leq i,j \leq n$, and a positive integer B.
Output: YES if there is a sequence $i_1 = 1, i_2, \ldots, i_{n-1}$ of $n-1$ pairwise distinct positive integers from $\{1,\ldots,n\}$ such that $\left(\sum_{j=1}^{n-2} d(i_j, i_{j+1})\right) + d(i_{n-1}, 1) \leq B$. Otherwise NO.

Prove that LazyTSP is $\mathcal{NP}$-complete. To this end, you may assume that LazyTSP is in $\mathcal{NP}$. Especially show:

(A) Find a suitable problem P_c which is known to be $\mathcal{NP}$-complete.

(B) Prove $P_c \leq_p$ LazyTSP, especially:

(B.1) Describe a transformation T which transforms every instance $\boldsymbol{X}$ of P_c into an instance $T(\boldsymbol{X})$ of LazyTSP and which runs polynomial in the size $\|\boldsymbol{X}\|$ of $\boldsymbol{X}$.
Hint: use a simple transformation that adds one component.

(B.2) Show that if the answer to $\boldsymbol{X}$ is YES then so is the answer to $T(\boldsymbol{X})$.

(B.3) Show that if the answer to $T(\boldsymbol{X})$ is YES then so is the answer to $\boldsymbol{X}$.

Exercise 5: Suppose you run the algorithm from Section 5.2 of the lecture notes for computing an independent set on the undirected *star* graph $G_{\text{star}} = (V, E)$, defined by $V = \{1, \ldots, n\}$ and $E = \{\{1, i\} \mid i \in \{2, \ldots, n\}\}$ for some positive integer n.

a) Define a choice of the p_i (not necessarily all equal) that makes the algorithm return an independent set of maximal size, and explain why this is the case.

b) Determine the bound on the expected size of the independent set constructed by the algorithm in the following three cases. Use the analysis from the notes and simplify the formulas such that they finally depend only on n (or no variable). For example, you may find that the expected size is at least $n - 2 + 1/(n-1)$.

b.1) $p_i = 1/d_{\text{avg}}$ for $i \in \{1, \ldots, n\}$, where d_{avg} is the average degree of G?

b.2) $p_i = 1/d_i$ for $i \in \{1, \ldots, n\}$, where d_i is the degree of vertex i?

b.3) $p_i = 1$ for $i \in \{1, \ldots, n\}$?

c) Still using $p_i = 1$ for $i \in \{1, \ldots, n\}$: Prove that the algorithm in fact returns a largest independent set with probability $1/2$ and that the expected size of the independent set returned is at least $n - 2$.

Note: when the algorithm finds for $\{i, j\} \in E$ that both $i \in I$ and $j \in I$, it removes either i or j from I, chosen uniformly at random.

Exercise 6: Consider the following five clauses over the boolean variables $\{x_1, x_2, x_3\}$:

$$\begin{aligned}
c_1 &= x_1 \vee \overline{x_2} \\
c_2 &= x_3 \\
c_3 &= x_1 \vee x_2 \vee \overline{x_3} \\
c_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
c_5 &= \overline{x_1} \vee x_2
\end{aligned}$$

a) Suppose you draw a truth assignment uniformly at random. Compute for each clause the probability of being satisfied and derive the expected number of satisfied clauses.

b) Compute for each x_i, where $i \in \{1, 2, 3\}$, the fraction

$$p_i := \frac{\text{total number of unnegated } x_i \text{ in clauses}}{\text{total number of } x_i \text{ or } \overline{x_i} \text{ in clauses}},$$

i. e., the relative frequency of positive literals in the literals involving x_i.

Suppose you apply randomized rounding as described in Section 5.3 of the lecture notes, using the p_i instead of the $\hat{y}_i$ as probabilities to determine the truth settings of the corresponding variables. Compute the expected number of clauses satisfied by this kind of randomized rounding. Compare it to the expected number you obtained in part a).

Exercise 7: [**Multiple choice**] Answer the multiple choice questions below. There are 10 questions in total. An incorrect answer is counted negatively, that is, it cancels out a correct answer. However, the total number of points will be at least 0.

1.) If a problem is in the class $\mathcal{RP}$ then it is also in the class $\mathcal{NP}$. ☐ True ☐ False

2.) The expression $rand(1,2) \neq rand(2,3)$ is always true. ☐ True ☐ False

3.) Las Vegas algorithms always give a YES/NO-answer. ☐ True ☐ False

4.) If a randomized algorithm for a decision problem has one-sided error and gives the correct answer with probability exactly 1/2, then it can be converted to one that gives the correct answer with probability exactly 15/16. ☐ True ☐ False

5.) Section 5.4 of the lecture notes describes a pseudo-polynomial algorithm for 3-SAT. ☐ True ☐ False

6.) The expected value printed by following program snippet is 6. ☐ True ☐ False

```
$i \leftarrow 3$
**while** $(rand(3,4) + rand(1,2) \neq 6)$ **do**
  $i \leftarrow i + 1$
**end while**
print($i$);
```

7.) The probability that $(rand(2,3) = rand(2,3))$ is true is 1/2. ☐ True ☐ False

8.) The TSP tour $(1,2,3,4,5,6)$ can be transformed into the tour $(1,5,3,4,2,6)$ by a single 2-OPT mutation. ☐ True ☐ False

9.) The randomized search heuristic RLS optimizes every function $f\colon \{0,1\}^n \to \mathbb{R}$ in expected time $O(n^n)$. ☐ True ☐ False

10.) Suppose that the approximation algorithm for Knapsack from Section 4.5.3 of the notes is applied to an instance where all weights, values and the capacity are odd numbers and that $k = 2$ is chosen. Then the output will in no case be an optimal solution. ☐ True ☐ False