INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

	Stamp / Signature of the Invigilator

EXAMINATION (End Semester)	**SEMESTER (Autumn)**

Roll Number									**Section**		**Name**	
Subject Number	C	S	6	0	0	3	5	**Subject Name**	*Selected Topics in Algorithms*			

Department / Center of the Student		**Additional sheets**	

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

CS60035/CS60086 Selected Topics in Algorithms, Autumn 2015–2016

End-Semester Test

26–November–2015 CSE-107, 2:00–5:00pm ($+\varepsilon$) Maximum marks: 50

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. You start with the array $A = [1,2,\ldots,n]$. Your task is to return a random permutation of $1,2,\ldots,n$ in A itself (that is, using only $O(1)$ additional space). Propose an efficient algorithm to solve your problem. Argue that your algorithm is capable of generating any permutation of $1,2,\ldots,n$ with equal probability. **(6)**

Solution Assume that array indexing is zero-based. The following algorithm solves the given problem in $\Theta(n)$ time.

```
for $i = n-1, n-2, \ldots, 1$ (in that order), repeat:
    Choose $j \in_U \{0,1,\ldots,i\}$.
    Swap $A[i]$ with $A[j]$.
```

Start with any permutation π of $1,2,\ldots,n$ stored in A, and run selection sort on A. Conversely, if you start with $A = [1,2,\ldots,n]$ and apply the swaps of the selection sort in the reverse sequence, you have π stored in A. Therefore the above algorithm is capable of generating any permutation of $1,2,\ldots,n$. Different permutations are generated by different choices of j in the loop, and the number of choices is exactly $n(n-1)\cdots 2 = n!$, that is, each permutation of $1,2,\ldots,n$ is generated with probability $1/n!$.

2. Let $G = (V, E)$ be an undirected graph. A subset $C \subseteq V$ is called a vertex cover of G if each edge $e \in E$ has at least one endpoint in C. We want to compute a vertex cover C of G with as few vertices as possible (the *minimum vertex cover problem*). For simplicity, assume that G is connected.

We make a DFS traversal in G (starting from any arbitrary vertex). Denote by T the DFS tree produced by the traversal. Let C consist of all the non-leaf vertices in T.

(a) Prove that C is a vertex cover of G. **(6)**

Solution Let $e = (u, v) \in E$. Then e is either a tree edge (an edge of T) or a back edge. In both the cases, at least one of u and v must be a non-leaf node in T, that is, e is covered by C.

(b) Prove that this computation of C using the DFS traversal is a 2-approximation algorithm. **(6)**

Solution Let t be the number of internal nodes in T, that is, $|C| = t$. Let M be any matching in G. The absence of cross edges in a DFS traversal implies that for each edge $(u, v) \in M$, either u or v is an internal node (or both are), that is, $t \leqslant 2|M|$. Moreover, $\text{OPT} \geqslant |M|$, so $t \leqslant 2 \times \text{OPT}$.

(c) Suppose that each vertex $v \in V$ is associated with a positive weight $w(v)$. The *weighted vertex cover problem* deals with the computation of a vertex cover C of G such that $\sum_{v \in C} w(v)$ is as small as possible. Demonstrate that given any positive constant Δ, there exist infinitely many graphs, in which any DFS traversal leads to an approximation factor $> \Delta$. **(6)**

Solution Let $G = T$ with $|V| = n \geqslant 3$ be a star graph with the center having weight $(n-1)\Delta + 1$, and with each of the remaining vertices having weight 1. If we start the DFS traversal from the center, only the center is output as the vertex cover, and the weight of this cover is $(n-1)\Delta + 1$. If we start the DFS traversal from a non-center vertex, then that vertex and the center constitute the vertex cover having a weight of $(n-1)\Delta + 2$. On the contrary, if we constructed a vertex cover consisting of all vertices other than the center, that vertex cover would have a weight of $n-1$.

3. Let $A = [a_1, a_2, \dots, a_n]$ be an array of n positive integers with $S = \sum_{i=1}^{n} a_i$. Let I_0 and I_1 be two subsets of $I := \{1, 2, \dots, n\}$ with $I_0 \cap I_1 = \emptyset$, and $I_0 \cup I_1 = I$. Denote by A_0 and A_1 the following subcollections of A: $A_0 = [a_i]_{i \in I_0}$ and $A_1 = [a_i]_{i \in I_1}$. The *partition problem* deals with deciding the existence of I_0, I_1 satisfying: $S_0 := \sum_{i \in I_0} a_i = \sum_{i \in I_1} a_i =: S_1 = \dfrac{S}{2}$. Here, we deal with an optimization version of the problem. We want to find a partitioning of A, that is as balanced as possible, that is, we construct a partition I_0, I_1 such that $S_0 \leqslant S_1$, and S_1 is as small as possible.

(a) For $i \in \{1, 2, \dots, n\}$, let x_i denote the variable such that $x_i = \begin{cases} 0 & \text{if } i \in I_0, \\ 1 & \text{if } i \in I_1. \end{cases}$ Formulate the balanced set partitioning problem explained above as a $0,1$-LP (linear programming) problem. Also mention how this problem can be relaxed to an LP problem. **(6)**

Solution We have $S_1 = \sum_{i=1}^{n} a_i x_i$. Moreover, the condition $S_0 \leqslant S_1$ is equivalent to the condition $S_1 \geqslant \frac{S}{2}$. Therefore the $0,1$-LP formulation is:

$$\text{Minimize } \sum_{i=1}^{n} a_i x_i \text{ subject to } \sum_{i=1}^{n} a_i x_i \geqslant \frac{S}{2}, \text{ and } x_i \in \{0,1\} \text{ for all } i = 1, 2, \dots, n.$$

The relaxed LP formulation replaces each non-linear constraint $x_i \in \{0,1\}$ by the two linear constraints:

$$x_i \geqslant 0, \text{ and } x_i \leqslant 1.$$

(b) We run an LP-solver to get an optimum solution $(x_1^*, x_2^*, \ldots, x_n^*) \in [0,1]^n$ of the relaxed LP instance. We make a deterministic rounding as follows: Take $i \in I_0$ if $x_i^* < \frac{1}{2}$, and $i \in I_1$ if $x_i^* \geqslant \frac{1}{2}$. Demonstrate by examples that this rounding may produce an infeasible solution (that is, $S_1 < \frac{S}{2}$), and a solution that can be as bad as possible (that is, $S_1 = S$). **(6)**

Solution Infeasible solution: Take $a_1 = a_2 = \cdots = a_{n-1} = 1$, and $a_n = n$. Here, the optimal solution to the discrete optimization problem is OPT $= n$. The relaxed LP instance has a solution with $S_1 = n - \frac{1}{2}$ achieved by $x_1^* = x_2^* = \cdots = x_{n-1}^* = 1$, and $x_n^* = \frac{1}{2n}$. But then, rounding gives $S_1 = \{1, 2, \ldots, n-1\}$ leading to $S_1 = n - 1 < \frac{S}{2}$.

Worst possible solution: Take $a_1 = a_2 = \cdots = a_n$. The LP solver may output $x_1^* = x_2^* = \cdots = x_n^* = \frac{1}{2}$. Rounding gives $I_1 = \{1, 2, \ldots, n\}$, so $S_1 = S$.

(c) Let us do randomized rounding instead. For each $i \in I$, let us independently take $x_i = 1$ with probability x_i^*. Denote $X = \sum_{i=1}^{n} a_i x_i$. Prove that $\Pr\left(X \geqslant \frac{3}{2} \times \mathrm{OPT}\right) \leqslant 4\left(\sum_{i=1}^{n} a_i^2\right) \Big/ \left(\sum_{i=1}^{n} a_i\right)^2$, where OPT is the optimal solution of the discrete optimization problem. **(6)**

Solution We have $\mathrm{E}(X) = \sum_{i=1}^{n} a_i x_i^*$, and $\frac{S}{2} \leqslant \mathrm{E}(X) \leqslant \mathrm{OPT}$. But then, $\Pr\left(X \geqslant \frac{3}{2} \times \mathrm{OPT}\right) = \Pr\left(X - \mathrm{OPT} \geqslant \mathrm{OPT}/2\right) \leqslant \Pr\left(X - \mathrm{E}(X) \geqslant \mathrm{OPT}/2\right) \leqslant \Pr\left(|X - \mathrm{E}(X)| \geqslant \mathrm{OPT}/2\right) \leqslant 4\mathrm{Var}(X)/\mathrm{OPT}^2 \leqslant 16\mathrm{Var}(X)/S^2$. For each i, we have $\mathrm{E}(x_i) = x_i^*$ and $\mathrm{E}(x_i^2) = x_i^*$, so $\mathrm{Var}(x_i) = \mathrm{E}(x_i^2) - \mathrm{E}(x_i)^2 = x_i^*(1 - x_i^*) \leqslant \frac{1}{4}$. Since x_i are independent of one another, we have $\mathrm{Var}(X) = \sum_{i=1}^{n} \mathrm{Var}(a_i x_i) = \sum_{i=1}^{n} a_i^2 \mathrm{Var}(x_i) = \sum_{i=1}^{n} a_i^2 x_i(1 - x_i) \leqslant \frac{1}{4}\sum_{i=1}^{n} a_i^2$. It follows that $\Pr\left(X \geqslant \frac{3}{2} \times \mathrm{OPT}\right) \leqslant 4(\sum_{i=1}^{n} a_i^2)/S^2 = 4(\sum_{i=1}^{n} a_i^2)/(\sum_{i=1}^{n} a_i)^2$.

Hint: For any $a > 0$, we have *Chebyshev's inequality*: $\Pr\left(|X - \mathrm{E}(X)| \geqslant a\right) \leqslant \frac{\mathrm{Var}(X)}{a^2}$.

Remarks: The probability bound of Part (c) is not always good. The trouble associated with infeasible solutions can be overcome by switching the roles of I_0, I_1 for both deterministic and randomized rounding.

4. Recall that in the bin packing problem, n items of weights $a_1, a_2, \ldots, a_n \in (0,1]$, and bins of capacity one are given. Our task is to pack *all* of the given items in as few bins as possible. Consider the strategy that keeps on adding items to the most recently opened bins so long as possible. When further placements are not possible, the last bin is closed, a new bin is opened, and the next object is put in the newly opened bin. One calls this the *next fit* or the *linear bin packing* strategy.

(a) Prove that the next fit strategy is a 2-approximation algorithm for the bin packing problem. **(6)**

Solution Let m be the number of bins used by the next-fit algorithm. Suppose that a bin j is at most half full in this packing. But then, the next bin $j+1$ must be more than half full. Otherwise, the opening of bin $j+1$ is not justified. More importantly, the total weight of the two bins j and $j+1$ must be larger than one. In fact, the sum of the weights of the items placed in bin j and the weight of the first item put in bin $j+1$ must be larger than one. It follows that among the first $m-1$ bins, at most $m/2$ bins can be at most half full, each followed by a bin more than half full that makes the average weight of the two consecutive bins larger than half. Therefore the total weight packed is $\sum_{i=1}^{n} a_i > m/2$. Moreover, $\text{OPT} \geqslant \sum_{i=1}^{n} a_i$, that is, $m < 2 \times \text{OPT}$. Since m and OPT are integers, we have $m \leqslant 2 \times \text{OPT} - 1$.

(b) Prove that the approximation ratio 2 is tight, that is, given any $\varepsilon > 0$, there exists an input instance for which the algorithm achieves an approximation ratio $> 2 - \varepsilon$. **(6)**

Solution Given $\varepsilon > 0$, choose a positive integer $k > \frac{1}{\varepsilon}$. Let there be $k-1$ items of weight one, and k items of weight $\frac{1}{k}$. Suppose that these items occur in the sequence $\frac{1}{k}, 1, \frac{1}{k}, 1, \ldots, \frac{1}{k}, 1, \frac{1}{k}$. We have $\mathrm{OPT} = k$ (place all the objects of weight $\frac{1}{k}$ in one bin, and each item of weight one in a single bin; this strategy must be optimal since all the k bins are filled to the capacity). The next-fit algorithm uses $m = 2k - 1$ bins. The approximation ratio is therefore $2 - \frac{1}{k} > 2 - \varepsilon$.

For leftover answers and rough work

If you optimize everything, you will always be unhappy.
— Donald E. Knuth