Computer Science CSC263H
St. George Campus

February 15, 2018
University of Toronto

Homework Assignment #4

Due: March 1, 2018, by 5:30 pm

- You must submit your assignment as a PDF file of a typed (**not** handwritten) document through the MarkUs system by logging in with your CDF account at:

  `https://markus.teach.cs.toronto.edu/csc263-2018-01`

  To work with one or two partners, you and your partner(s) must form a group on MarkUs.

- The PDF file that you submit must be clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.

- If this assignment is submitted by a group of two or three students, the PDF file that you submit should contain for each assignment question:

  1. The name of the student who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.

- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy that is stated in the csc263 course web page: `http://www.cs.toronto.edu/~sam/teaching/263/#HomeworkCollaboration`.

- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbook, by referring to it.

- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision, and conciseness of your presentation.

- Your submission should be no more than 2.5 pages long in a 10pt font.

Question 1. (1 marks) Consider the sorting algorithm given by the pseudocode below. It takes an array $A[1\,..\,n]$ of size n, and outputs A with its elements in sorted (non-decreasing) order.

```
1  for i = 2 to n
2      j = i − 1
3      while A[j + 1] < A[j]  &  j ≥ 1
4          swap A[j] and A[j + 1]
5          j = j − 1
```

In the following subquestions, assume that the array A contains a uniformly chosen random permutations of the integers $1, \ldots, n$.

a. Let S_i be the number of swaps performed by the algorithm in the i-th iteration of the for-loop. What is the ***exact expected value*** of S_i as a function of n and i? Justify your answer.

b. Let $S = S_1 + \ldots + S_{n-1}$ be the total number of swaps performed by the algorithm. What is ***exact expected value*** of S as a function of n? Justify your answer.

Question 2. (1 marks) Let $G = (V, E)$ be an undirected graph on the nodes $V = \{1, \ldots, n\}$. Suppose the edges of the graph are removed one by one in some order $e_1, \ldots, e_m$. After the removal of some edge e_i the graph will have *no cycles* remaining. Assume the $e_1, \ldots, e_m$ edges are given as an array $A[1\,..\,m]$, where $A[i]$ contains the tuple of vertices $e_i = (u, v)$. Give an algorithm that, when given as input n and the array A, outputs the smallest positive integer i such that the graph $G_i = (V, E - \{e_1, \ldots, e_{i-1}\})$ has no cycles. (Define $G_1 = G$.) **The running time of your algorithm must be asymptotically better than $O(mn)$.** Give pseudocode for your algorithm, analyze its running time, and prove it is correct.

[***The questions below will not be corrected/graded. They are given here as interesting problems that use material that you learned in class.***]

Question 3. (0 marks)

Let $p_2, \ldots, p_n$ be real numbers in $[0, 1]$, to be specified later. Consider the following randomized algorithm:

```
1  x = 1
2  for i = 2 to n
3      With probability p_i, x = i; otherwise x is unchanged.
4  return x
```

a. Consider the value of x returned by the above procedure. Compute $\Pr[x = i]$ for each $1 \le i \le n$, under the assumption that $p_2 = p_3 = \ldots = p_n = 1/2$. Give a precise mathematical derivation of these probabilities.

b. Give values for $p_2, \ldots, p_n$ so that for any $1 \le i \le n$, $\Pr[x = i] = 1/n$, i.e. x is a uniform sample from $1, \ldots, n$. Prove that, with the values of $p_2, \ldots, p_n$ that you give, the probability of $x = i$ is $1/n$, as required.

Question 4. (0 marks) Assume you have a biased coin, which, when flipped, falls on Heads with probability p, where $0 < p < 1$, and on Tails with probability $1 - p$. However, you do not know p. How can you use the coin to simulate an unbiased coin? Formally, you have access to a procedure FlipBiasedCoin(), which returns either 1 or 0 at random. FlipBiasedCoin() returns 1 with probability p, and 0 with probability $1 - p$, but you do not know p. Design an algorithm that, without making any other random choices except calling FlipBiasedCoin(), returns 1 with probability 1/2 and 0 with probability 1/2. Your algorithm should not use p. You can assume that each time you call FlipBiasedCoin(), the value it returns is independent of all other calls to it.

a. Describe the algorithm in clear and concise English, and prove that it outputs 0 with probability 1/2 and 1 with probability 1/2.

b. Analyze the expected running time of your algorithm. Note that while the algorithm itself does not use p, the expected running time should be expressed in terms of p.

Question 5. (0 marks) Consider the following "Graph Dismantling Problem". Let $G = (V, E)$ be a connected undirected graph with $n \geq 4$ nodes $V = \{1, 2, \ldots, n\}$ and m edges. Let $e_1, e_2, \ldots, e_m$ be all the edges of G listed in some specific order. Suppose that we ***remove*** the edges from G one at a time, ***in this order***. Initially the graph is connected, and at the end of this process the graph is disconnected. Therefore, there is an edge e_i, $1 \leq i \leq m$, such that just before removing e_i the graph has at least one connected component with more than $\lfloor n/4 \rfloor$ nodes, but after removing e_i every connected component of the graph has at most $\lfloor n/4 \rfloor$ nodes. Give an efficient algorithm that determines this edge e_i. Assume that G is given to the algorithm as a (plain) linked list of the edges appearing in the order $e_1, e_2, \ldots, e_m$.

The worst-case running time of your algorithm ***must be asymptotically better than*** $O(mn)$***.***

Hint: See "An application of disjoint-set data structures" in pages 562-564 of CLRS (Chapter 21).