CSOR W4231.001 – Spring, 2023
Eleni Drinea

**Homework 4 (85 points)**

Out: Monday, March 27, 2023
Due: 11:59pm, Monday, April 10, 2023

**Homework Instructions.**

1. For all algorithms that you are asked to "give" or "design", you should

   - Describe your algorithm clearly in English.
   - Give pseudocode.
   - Argue correctness; you may provide a formal proof or a convincing argument.
   - Provide, with an explanation, the best (smallest) upper bound that you can for the running time. All bounds should be **worst-case** bounds, unless explicitly stated otherwise.

   You are also encouraged to analyze the space required by your algorithm. We will not remove marks if you don't, unless the problem explicitly asks you to analyze space complexity.

2. For every randomized algorithm that *you* design, you must either argue that it always terminates with the correct answer or analyze its **success probability**, that is, the probability it terminates with the correct answer. You must argue whether its running time is deterministic or a random variable and give the best upper bound that you can for it (if the running time is a random variable, you should upper bound its *expectation*). You are also encouraged to analyze the space required by your algorithm.

   - If a randomized algorithm is provided to you, follow carefully the instructions in the problem statement to analyze it.

3. **You should not use any external resources for this homework.** Failure to follow this instruction may have a negative impact on your performance in exams and interviews. For the same reason, **you should avoid collaborating** with your classmates, at least until you have thought through the problems on your own for a reasonably long time.

4. You should submit this assignment as a **pdf** file to Gradescope. Other file formats will not be graded, and will automatically receive a score of 0.

5. I recommend you type your solutions using LaTeX. For every assignment, you will earn 5 extra credit points if you type your solutions using LaTeX or other software that prints equations and algorithms neatly. If you do not type your solutions, make sure that your handwriting is very clear and that your scan is high quality.

6. You should write up your solutions **entirely on your own**. Collaboration is limited to discussion of ideas only. You should adhere to the department's academic honesty policy (see the course syllabus). Similarity between your solutions and solutions of your classmates or solutions posted online will result in receiving a 0 in this assignment, and possibly further disciplinary actions. There will be no exception to this policy and it may be applied retro-actively if we have reasons to re-evaluate this homework.

**Homework Problems**

1. (30 points) Consider a directed graph constructed by the following discrete-time process. [1] You start with a single node $v_1$. At every time step $t$ with $1 < t \leq n$, a single node $v_t$ is added to the graph, together with a single outgoing edge; the destination of the edge is a node chosen uniformly at random among the existing nodes in the graph.

   Let $G$ be the graph at the end of time step $n$.

   (a) (15 pts) What is the expected number of edges entering node $v_j$ in $G$? Give an exact formula in terms of $n$ and $j$, as well as an asymptotic expression using $\Theta$ notation.

   (b) (15 pts) What is the expected number of nodes with no incoming edges in $G$?

2. (25 pts) Consider the following online auction system. There are $n$ bidding agents; agent $i$ has an integer bid $b_i > 0$. All bids are distinct. The bidding agents appear in an order chosen uniformly at random. Each agent proposes its bid $b_i$ in turn, and at all times the system maintains a variable $b_{max}$ equal to the highest bid seen so far.

   How many times do you expect to update $b_{max}$ when this process is executed?

3. (30 points) In this problem, you will analyze a randomized algorithm that finds an item *close enough* to the median item of a set $S = \{a_1, \ldots, a_n\}$ of $n$ distinct numbers. Specifically, the algorithm finds an item $a_i$ such that at least $n/4$ items are smaller than $a_i$ and at least $n/4$ items are greater than $a_i$.

**Algorithm 1**

`Randomized Approximate Median` $(S)$

1: Select an item $a_i \in S$ uniformly at random
2: $rank = 1$
3: **for** $j = 1$ to $n$ **do**
4: **if** $a_j < a_i$ **then** $rank = rank + 1$
5: **end if**
6: **end for**
7: **if** $\lfloor n/4 \rfloor < rank \leq \lfloor 3n/4 \rfloor$ **then return** $a_i$
8: **else return error**
9: **end if**

[1] Ideas along these lines have been used for building models to explain statistics of the Internet graph and growth of peer-to-peer networks.

(a) (5 points) What kind of randomized algorithm is `Randomized Approximate Median`$(S)$ and why?

(b) (3 points) What is the running time of this algorithm?

(c) (8 points) What is the **success probability** of this algorithm?

(d) (14 points) Show how you can **amplify** (improve) the **success** probability of `Randomized Approximate Median`$(S)$ to over 99% by running several independent executions of the algorithm. You should analyze the success probability and the running time of the resulting algorithm.

**RECOMMENDED EXERCISES (do NOT return, they will not be graded)**

1. Problem 24-3, part a. only, from your textbook (p. 679).

2. Problem 7-2 in the textbook.

3. Problem 7-4 in the textbook.
   (If necessary, read pages 232-233 to refresh your memory on the definition of a stack.)

4. Given an *undirected unweighted* graph $G = (V, E)$, we define a **cut** $(S, V - S)$ to be a bipartition of the vertices. The **size** of a cut $(S, V - S)$ is defined as the number of *crossing* edges, that is, edges with one endpoint in $S$ and the other in $V - S$; we will henceforth denote the set of these edges by $F$. A **global min-cut** is a cut with minimum size.

   The **global min-cut** is a natural *robustness* parameter of the graph: its size represents the smallest number of edges whose deletion disconnects the graph.

   Your task is to analyze the following randomized algorithm for computing the global min-cut. (This elegant and clever randomized algorithm is due to David R. Karger.)

   (a) (4 pts) Let $(S^*, V - S^*)$ be a global min-cut in $G$ of size $k$. Let $F$ be the set of *crossing* edges of $(S^*, V - S^*)$. Compute the probability that an edge in $F$ is contracted in the first iteration of the algorithm.

   (b) (8 pts) Derive a lower bound on the total number of edges in $G$, based on the fact that a global min-cut has size $k$.

   (c) (4 pts) Derive an upper bound for the probability that a crossing edge in $F$ is contracted in the first iteration of the algorithm using parts (a) and (b) above.

**Algorithm 2**

`GlobalMinCut`$(G = (V, E))$

1: // For each node $v$, maintain the set $C_v$ of nodes that have been **contracted** into $v$
2: Initialize $C_v = \{v\}$ for every $v$
3: **if** $G$ has two nodes $x$ and $y$ **then**
4: return the cut $(C_x, C_y)$
5: **else**
6: Choose an edge $e = (x, y)$ uniformly at random
7: // **contract** edge $e$
8: Replace $e$ by a single new node $v_{xy}$
9: Remove all edges between $x$ and $y$ from $G$
10: For any edge that has exactly one endpoint in $\{x, y\}$, update that endpoint to be $v_{xy}$
11: Set $C_{v_{xy}} = C_x \cup C_y$
12: Let $G'$ be the resulting graph // note that $G'$ might be a **multigraph**
13: `GlobalMinCut`$(G')$
14: **end if**

(d) (17 pts) Derive a lower bound for the probability that the algorithm successfully returns the cut $(S^*, V - S^*)$ upon termination(using your answer in part (c). We will call this probability the **success probability** of the algorithm, and denote it by $p_s$.

*Hint: The algorithm will successfully return the cut $(S^*, V - S^*)$ if no crossing edge in $F$ is contracted in any recursive call before the last one. Let $\varepsilon_i$ be the event that, in call $i$, the contracted edge is not an edge in $F$. First express the event that the algorithm successfully returns $(S^*, V - S^*)$ upon termination in terms of the events $\varepsilon_i$. Then lower bound $p_s$ in terms of the probabilities of the $\varepsilon_i$'s.*

(e) (2 pts) Derive an upper bound for the probability that the algorithm fails to return $(S^*, V - S^*)$ upon termination. We will call this probability the **failure probability** of the algorithm, and denote it by $p_f$.

(f) (15 pts) Show how you can **amplify** the **success probability** $p_s$ to at least $1 - 1/n$. (Equivalently, show how to *reduce* the **failure probability** $p_f$ to at most $1/n$ by **running it repeatedly**.) Analyze the running time of the new algorithm. *Hint: You may want to use the inequality $1 + x \leq e^x$.*

(g) (10 pts) Give an upper bound on the number of global min-cuts in $G$.

*Hint: Observe that what you computed in part (d) is the probability that the algorithm successfully returns a specific global min-cut $(S^*, V - S^*)$. Let $M$ be the number of global min-cuts in $G$, and let $\mathcal{E}_i$ be the event that Karger's algorithm returns the global min-cut $(S_i, V - S_i)$ for $1 \leq i \leq M$. Now let $\mathcal{E}$ be the event that Karger's algorithm returns* **any** *of the global min-cuts $(S_i, V - S_i)$'s. First, express $\mathcal{E}$ in terms of the $\mathcal{E}_i$'s, and derive a lower bound for $\Pr[\mathcal{E}]$. Then use this lower bound to deduce an upper bound for $M$.*