Homework #3

(Due: Dec 4)

Task 1. [80 Points] Searching in Circular Linked Lists Stored in Arrays

You are given n distinct real numbers in an array $A[1:n]$ and a permutation of the first n natural numbers in another array $Next[1:n]$. The permutation in $Next[1:n]$ is carefully created to ensure that if, for any $i \in [1,n]$, $A[i]$ is the largest number in A then $A[Next[i]]$ is the smallest, otherwise $A[Next[i]]$ is the smallest number in A with value larger than $A[i]$. Figure 1 shows an example.

(*a*) [**40 Points**] Given any number $s \in \mathbb{R}$, you are required to search for s in A. Design a randomized algorithm that returns FALSE if the number does not exist in A, otheriwse returns TRUE along with the index $i \in [1,n]$ such that $A[i] = s$. Your algorithm must run in $\mathcal{O}(\sqrt{n}\log n)$ time w.h.p. in n. You cannot preprocess the given data and cannot use any extra space.

	1	2	3	4	5	6	7	8
A:	25	7	20	5	11	15	42	38

	1	2	3	4	5	6	7	8
Next:	8	5	1	2	6	3	4	7

Figure 1: [Task 1] All numbers in array A are distinct and linked in sorted order using a circular linked list formed by the pointers stored in the $Next$ array. If $A[i]$ is the largest number in A then $A[Next[i]]$ is the smallest, otherwise $A[Next[i]]$ is the next larger number in the sorted order.

Now consider n distinct real numbers that were originally stored in an $\sqrt{n} \times \sqrt{n}$ array A' such that the numbers in every row were sorted in increasing order from left to right and those in every column were sorted in increasing order from top to bottom. However, you do not have access to that array. Instead, you are given another array $A[1:\sqrt{n}, 1:\sqrt{n}]$ where the same n numbers are stored, but possibly in a different order. However, with every entry $p = A[i,j]$ you are given four pointers (each with a row index and a column index): $Left[i,j]$, $Right[i,j]$, $Up[i,j]$, and $Down[i,j]$. The entry $q = A[Left[i,j].row, Left[i,j].col]$ is the left neighbor of p in A' provided p is not the first entry in its row in A', otherwise q is the last entry in that row. The entry $r = A[Right[i,j].row, Right[i,j].col]$ is the right neighbor of p in A' provided p is not the last entry in its row in A', otherwise r is the first entry in that row. Similarly, for $Up[i,j]$ and $Down[i,j]$. Figure 2 shows an example.

(*b*) [**40 Points**] Given a 2D array $A[1:\sqrt{n}, 1:\sqrt{n}]$ as described above with four pointers per cell, and any number $s \in \mathbb{R}$, you are required to search for s in A. Design an efficient

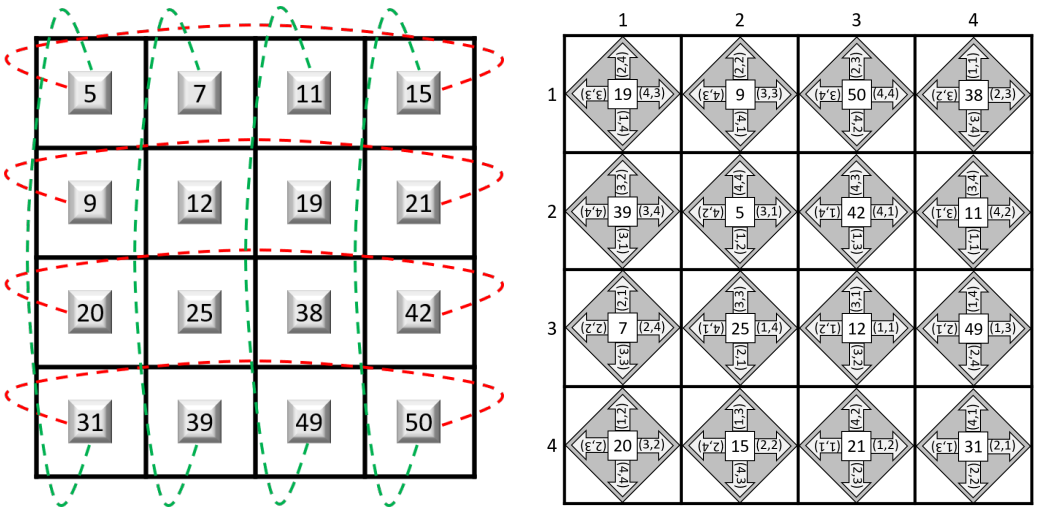

Figure 2: [Task 1] The numbers in every row of the 4×4 array on the left are sorted in increasing order from left to right and those on every row are sorted from top to bottom. All number in the array are distinct. The 4×4 array on the right stores the same numbers but not necessarily in the same order. However, the rowwise sorted order of the numbers in the left array are also maintained in the right array using the left and right pointers. The columnwise sorted order can be retrieved by following the up and down pointers.

randomized algorithm for the purpose and prove a high probability (upper) bound on its running time. As in part (a), you are not allowed to preprocess the given data and you cannot use any extra space.

Task 2. [100 Points] Recursive Randomized Min-Cut

Consider the randomized min-cut algorithm we saw in the class that returns a min-cut with probability $\geq 1 - \frac{1}{e}$. Given a connected undirected multigraph with n vertices, the strategy is to run the following algorithm $\frac{n^2}{2}$ times and return the smallest cut identified by those runs. Each run uses an algorithm that starts with the original n-vertex graph and performs a sequence of $n-2$ edge contractions. Each contraction is performed on an edge chosen uniformly at random from the current set of edges. A contraction step contracts the two endpoints of the given edge into a single vertex and removes all edges between them, but retains all other edges (and thus leading to a multigraph). After $n-2$ contraction steps only 2 vertices remain, and all edges between those two vertices are returned as a potential min-cut.

(a) [**10 Points**] Argue that each contraction step can be implemented to run in $\mathcal{O}(n)$ time, and thus the randomized min-cut algorithm described above takes $\mathcal{O}\left(n^4 \log n\right)$ time to return a min-cut with high probability.

Now consider the recursive algorithm given in Figure 3.

RECURSIVE-RANDOMIZED-MIN-CUT(G, α)
(Input is an undirected multigraph G with n vertices, and an integer constant $\alpha > 0$. Output is a cut of G.)

1. ***if*** $n \leq \alpha^3$ ***then***
2. $\quad C \leftarrow$ a min-cut of G found using brute force (exhaustive) search
3. ***else***
4. $\quad$ ***for*** $i \leftarrow 1$ ***to*** α ***do***
5. $\quad\quad G' \leftarrow$ multigraph obtained by applying $n - \left\lceil \frac{n}{\sqrt{\alpha}} \right\rceil$ random contraction steps on G
6. $\quad\quad C' \leftarrow$ RECURSIVE-RANDOMIZED-MIN-CUT(G', α)
7. $\quad\quad$ ***if*** $i = 1$ ***or*** $|C'| < |C|$ ***then*** $C \leftarrow C'$
8. ***return*** C

Figure 3: [Task 2] A recursive randomized min-cut algorithm.

(b) [**10 Points**] Let $T(n)$ be the running time of the algorithm on a multigraph with n vertices. Write a recurrence relation describing $T(n)$ and solve it.

(c) [**10 Points**] Let $P(n)$ be the probability that the algorithm returns a min-cut when run on a multigraph with n vertices. Argue that

$$P(n) \geq 1 - \left(1 - \frac{1}{\alpha} P\left(\frac{n}{\sqrt{\alpha}} + 1\right)\right)^{\alpha}.$$

(d) [**5 Points**] Let $p(r)$ be the probability that the algorithm returns a min-cut from recursion level $r \geq 0$, where $r = 0$ at the base case (in line 2). Argue that $p(0) = 1$, and use part (c) to show that for $r > 0$,

$$p(r) \geq 1 - \left(1 - \frac{1}{\alpha} p\left(r - 1\right)\right)^{\alpha}.$$

(e) [**30 Points**] Let $q(r) = \frac{\alpha}{p(r)}$. Show that $q(r) \leq \left(\frac{\alpha(\alpha-1)^{\alpha}}{\alpha^{\alpha} - (\alpha-1)^{\alpha}}\right) r + \alpha$.

(f) [**10 Points**] Let $r(n)$ be the number of recursion levels to descend to reach the base case starting from an input of size n. Write a recurrence relation for $r(n)$ and solve it to show that $r(n) \leq \log_{\left(\frac{\alpha^2}{1+\alpha\sqrt{\alpha}}\right)} \left(\frac{n}{\alpha^3}\right)$.

(g) [**10 Points**] Use parts (c)–(f) to show that $P(n) \geq \frac{1}{\left(\frac{\alpha(\alpha-1)^{\alpha}}{\alpha^{\alpha} - (\alpha-1)^{\alpha}}\right) \log_{\left(\frac{\alpha^2}{1+\alpha\sqrt{\alpha}}\right)} \left(\frac{n}{\alpha^3}\right) + 1}$.

(h) [**15 Points**] Based on the result from part (g), how would you use the algorithm in Figure 3 to obtain a min-cut of an n-node multigraph w.h.p. in n? What is the running time of the resulting algorithm?