DDA6050 Assignment 3

Zihan, Zhou 119010484@link.cuhk.edu.cn

If you have any questions about grading, please feel free to reach out to me via email at 119010484@link.cuhk.edu.cn.

1 Amortized Analysis (20 pts)

1. Assuming we possess two stacks capable of performing push and pop operations at a constant $O(1)$ cost per operation, construct a queue with push and get operations in such a way that the amortized time for each queue operation remains within $O(1)$. Also prove that the implemented queue indeed has the $O(1)$ amortized time per operation. (10 pts)

2. Suppose that we run a program over a sequence of n days. On the i-th day, if $\log_2(i)$ is an integer, then this program costs i units of computation resources to examine the outputs obtained so far. Otherwise, it only costs 1 unit of computation resource this day. Compute the amortized computation cost per day. (10 pts)

2 Element Selection Algorithm (20 pts)

Suppose we are given a sorted circular linked list of numbers, implemented as a pair of arrays, one storing the actual numbers and the other storing successor pointers. Specifically, we are given an array $X[1 \ldots n]$ of distinct real numbers and an array next $[1 \ldots n]$ be an array of indices with the following property:

- If $X[i]$ is the largest element of X, then $X[\text{next}[i]]$ is the smallest element of X
- Otherwise, $X[\text{next}[i]]$ is the smallest element of X that is larger than $X[i]$.

For example:

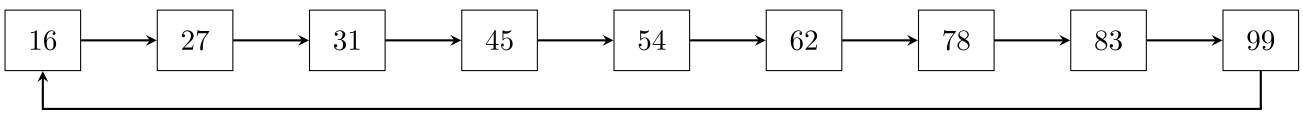
i	1	2	3	4	5	6	7	8	9
$X[i]$	83	54	16	31	45	99	78	62	27
next $[i]$	6	8	9	5	2	3	1	7	4

Start at any index and follow the next pointers to see the sorted order.

For example, starting at index 5 (chosen arbitrarily):

- $X[5] = 45$
- $\text{next}[5] = 2$, so go to $X[2] = 54$
- $\text{next}[2] = 8$, so go to $X[8] = 62$
- $\text{next}[8] = 7$, so go to $X[7] = 78$

- next[7] $= 1$, so go to $X[1] = 83$
- next[1] $= 6$, so go to $X[6] = 99$
- next[6] $= 3$, so go to $X[3] = 16$ (wraps around to smallest)
- next[3] $= 9$, so go to $X[9] = 27$
- next[9] $= 4$, so go to $X[4] = 31$
- next[4] $= 5$, back to $X[5] = 45$ (cycle completes)



Assume n is sufficiently large.

1. Describe and analyze a randomized algorithm that determines whether a given number x appears in the array X in $\mathcal{O}(\sqrt{n})$ expected time without modifying the input arrays. (10 pts) **Hints:** Randomly sample $\mathcal{O}(\sqrt{n})$ distinct indices uniformly and choose one of them, and continue to search in $\mathcal{O}(\sqrt{n})$ time complexity.

2. Analyze the probability of success and show that the algorithm succeeds with high probability (i.e., at least 99%). (10 pts)

3 Shelf Scheduling Algorithm (50 pts)

Suppose we are given a collection of n jobs to execute on a machine containing a row of p identical processors. The parallel scheduling problem asks us to schedule these jobs on these processors, given two arrays $T[1 \dots n]$ and $P[1 \dots n]$ as input, subject to the following constraints:

- When the i-th job is executed, it occupies a contiguous interval of $P[i]$ processors for exactly $T[i]$ seconds.
- No processor works on more than one job at a time.

A valid schedule specifies a non-negative starting time and an interval of processors for each job that meets these constraints. Our goal is to compute a valid schedule with the smallest possible makespan, which is the earliest time when all jobs are complete. You may assume that n is a power of 2.

1. Prove that the parallel scheduling problem is NP-hard. (15 pts) **Hints:** You only need to consider the case where $n = 2$.

2. Describe a polynomial-time algorithm that computes a 3-approximation of the minimum makespan of a given set of jobs. That is, if the minimum makespan is M, your algorithm should compute a schedule with a make-span at most $3M$. (15 pts) **Hints:**

   - Since p is a power of 2 , you can write $p = 2^k$ for some integer k. This means there are only $\log p$ different "scales" of processor requirements that matter.

- Visualize each job as a rectangle: width= $P[i]$ (processors needed), height= $T[i]$ (time duration).
- Think about packing these rectangles into shelves: Place jobs left-to-right within each shelf (horizontal), stack shelves bottom-to-top (vertical), and process job categories from large-to-small (big rectangles first).

3. Let M^* be the optimal makespan. Briefly show that $M^* \geq \max_{i=1}^{n} T[i]$ (longest job lower bound) and $M^* \geq \frac{\sum_{i=1}^{n} T[i] \cdot P[i]}{p}$ (total work lower bound) (5 pts).

4. Prove that your proposed algorithm is a 3-approximation. (15 pts) **Hints:** Decompose the schedule:

$$\text{ALG} = A + B$$

   where:

   - A: total height of **non-last shelves**
   - B: total height of **last shelves**

   Bounds:

   - $A \leq 2 \cdot \text{OPT}$ (non-last shelves are $\geq$ half-full)
   - $B \leq 1 \cdot \text{OPT}$ (last shelves contain $\leq$ one long job per level)

4 Randomized Algorithm. (10 pts)

Suppose we have n servers and m tasks. Each task is independently and uniformly randomly assigned to a server among n of them. In this question, you do not need to rigorously consider if a value is integer. Suppose that $m = 2n \log n$.

1. We focus on the first server. Show that the probability of it receiving at least $2e \cdot \log n$ tasks is no larger than $1/n^2$. (5 pts)

2. Show that when n is large enough, then with high probability, no server receives at least $2e \log n$ tasks. (5 pts)