Quiz 1

Quiz 1 Grades

0.10 11.20 21.30 31.40 41.50 51.60 61.70 71.80
0 7 14 21 28
Problem 0. Name. [1 point] Write your name on every page of this exam booklet! Don’t forget the cover.

Possibly useful facts for elsewhere in the quiz

1. **Markov Inequality:** For any nonnegative random variable $X$, we have
   \[ \Pr \{ X \geq \lambda \} \leq \frac{\mathbb{E} [X]}{\lambda}. \]

2. **Chernoff Bounds:** Let $X_1, \ldots, X_n$ be $n$ independent Boolean random variables. Suppose that for $i = 1, 2, \ldots, n$, we have $\Pr \{ X_i = 1 \} = \delta_i$ for $0 \leq \delta_i \leq 1$. Let $X = \sum_{i=1}^{n} X_i$ and
   \[ \delta = \left( \frac{1}{n} \right) \cdot \sum_{i=1}^{n} \delta_i. \]
   Then, for any $\delta \leq \gamma \leq 1$,
   \[ \Pr \{ X \geq \gamma n \} \leq e^{-2(\gamma - \delta)^2 n}. \]
   Note that this is a generalization of the Chernoff bound we saw in class to the case of not necessarily identically distributed random variables.

3. **Harmonic series:**
   \[ \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1).\]
Problem 1. True or False. [21 points] (7 parts)

Circle T or F for each of the following statements to indicate whether the statement is true or false, respectively. You need not justify your answers, but since wrong answers will be penalized, do not guess unless you are reasonably sure.

(a) T F The recurrence \( T(n) = 2T(\sqrt{n}) + \lg n \) has solution \( T(n) = \Theta(\lg^2 n) \).

Solution: False: Substitute \( m = \lg n \). Then, \( T(m) = 2T(m/2) + m = \Theta(m \lg m) \). So, \( T(n) = \Theta(\lg n \lg \lg n) \).

(b) T F The numbers 1, 2, \ldots, 10 can be placed into a tree data structure such that the tree satisfies both the min-heap property and the binary-search-tree property at the same time.

Solution: True: Consider a tree that has only its rightmost branch, containing all the elements in monotonically increasing order.

(c) T F The following collection \( \mathcal{H} = \{h_1, h_2, h_3\} \) of hash functions is universal, where each hash function maps the universe \( U = \{A, B, C, D\} \) of keys into the range \( \{0, 1, 2\} \) according to the following table:

<table>
<thead>
<tr>
<th></th>
<th>( h_1(x) )</th>
<th>( h_2(x) )</th>
<th>( h_3(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution: True: By verifying that any two rows do not have more than one element in common.
(d) **T F** Consider a sequence of \( n \) **INSERT** operations, \( 2n \) **DECREASE-KEY** operations, and \( \sqrt{n} \) **EXTRACT-MIN** operations on an initially empty Fibonacci heap. The total running time of all these operations is \( \Theta(n \lg n) \) in the worst case.

**Solution:** False: The time by this sequence of operations is \( O(n + 2n + \sqrt{n} \lg n) = O(n) \).

(e) **T F** In the analysis of the disjoint-set data structure presented during lecture, once a node other than the root or a child of the root is block-charged, it will never again be path-charged.

**Solution:** True: Follows from the definitions of block-charges and path-charges.

(f) **T F** A van-Emde-Boas data structure can support **FIND-MIN**, **FIND-MAX**, **SUCCESSOR**, and **PREDECESSOR** operations over the set \( \{1, 2, \ldots, 2\sqrt{n}\} \) in \( O(\log n) \) time.

**Solution:** True: vEB trees can perform the above operations in \( O(\log \log (2\sqrt{n})) \) or \( O(\log n) \) time.

(g) **T F** Let \( G = (V, E) \) be a connected undirected graph with edge-weight function \( w : E \rightarrow \{1, 2, \ldots, 10 \mid |E|\} \). Then a minimum spanning tree of \( G \) can be constructed in \( O(E \lg \lg V) \) time.

**Solution:** True: Use vEB data structure as heap.
Problem 2. S3L3CT. [12 points]
Professor John Von Meanmann is implementing an algorithm to find the $i$th smallest of a set $S$ of $n$ distinct elements. The professor improvises on the traditional worst-case linear-time algorithm and organizes elements into groups of 3 instead of groups of 5, resulting in the following algorithm, called S3L3CT:

1. If $n = 1$, return the only element in the array.
2. Divide the $n$ elements of the input array into $\lfloor n/3 \rfloor$ groups of 3 elements each, with 0 to 2 elements left over.
3. Find the median of each of the $\lceil n/3 \rceil$ groups by rote.
4. Use S3L3CT recursively to find the median $x$ of the $\lceil n/3 \rceil$ medians found in Step 3.
5. Partition the input array around $x$. Let $k = \text{RANK}(x)$.
6. If $i = k$, then return $x$. Otherwise, use S3L3CT recursively to find the $i$th smallest element on the low side if $i < k$, or the $(i - k)$th smallest element on the high side if $i > k$.

State the recurrence for the running time $T(n)$ of S3L3CT running on an input of $n$ elements, and provide a tight asymptotic upper bound on its solution in terms of $n$. In order to simplify the math, assume that any set on which S3L3CT operates contains a multiple of 3 elements.

Solution: $x$ is at least as large as $n/3$ elements (for half of the triplets, 2 out of 3 elements) and at least as small as $n/3$ elements (for the other half of the triplets, 2 out of 3 elements). So the recursion in Step 5 is, in the worst case, on an array of size $2n/3$. The recursion in Step 3 is always on $n/3$ elements. So we get the recurrence $T(n) = T(2n/3) + T(n/3) + \Theta(n)$.

We now use a recursion tree to estimate a solution to the above recurrence. The depth of the tree is dominated by the term $T(2n/3)$. Thus, a good estimate for the depth of the recursion tree is $O(\log_{3/2} n)$, which is $O(\lg n)$. And from the recurrence, it is clear that we do $O(n)$ at each level. Thus, an initial guess would be that the recurrence is bounded by $O(n \lg n)$. We shall now use the substitution method to verify our guess. Make an inductive hypothesis that $T(m) \leq dm \lg m$, for all $m \in [1, n)$. We will, now, prove the hypothesis for $n$.

$$T(n) = d\frac{2n}{3} \lg \frac{2n}{3} + \frac{dn}{3} \lg n/3 + \Theta(n)$$
$$= 2dn/3(\lg 2 + \lg n - \lg 3) + nd/3(\lg n - \lg 3) + \Theta(n)$$
$$= dn/3(3\lg n + 2 \lg 2 - \lg 3) + \Theta(n)$$
$$= dn \lg n + dn/3(2 - 2 \lg 3) + \Theta(n)$$
$$\leq dn \lg n + 2dn/3(1 - \lg 3) + kn \quad \text{(from the definition of } \Theta(n)\text{)}$$
$$= dn \lg n + n(2d/3(1 - \lg 3) + k)$$
$$= dn \lg n - n(2d/3(\lg 3 - 1) - k)$$
$$< dn \lg n, \quad \text{for all } d > 3k/2(\lg 3 - 1)$$

Thus, we have established that there is an absolute constant $d$ such that for sufficiently large $n$, $T(n) < dn \lg n$. Thus, $T(n) = O(n \lg n)$. 

Problem 3. SWAT Team. [12 points]

Two swatsmen with fly swatters are located at arbitrary positions along a long corridor with many leaky windows. One at a time, houseflies appear at various locations along the corridor, and a swatsman goes to the location of the fly and swats it dead. The cost of a given strategy is the total distance traveled by the swatsmen. Argue that the greedy strategy of the closest swatsman going to the location of the fly is not $\alpha$-competitive for any finite $\alpha$.

(Hint: Consider the case that flies only appear at three locations $A$, $B$, and $C$, where $B$ falls between $A$ and $C$ and the distance from $A$ to $B$ is much smaller than the distance from $B$ to $C$, as shown below:

```
A ---- B ---- C
```

Consider the sequence $(C, A, B, A, B, A, A, B, A, A, ...)$ of fly arrivals.)

Solution: Suppose on way of contradiction that there are $\alpha$ and $\beta$ such that $Greedy \leq \alpha \cdot OPT + \beta$.

Let $L$ be the distance between $A$ and $C$, and $\epsilon L$ be the distance between $A$ and $B$, where $0 < \epsilon < 1/2$. Let $T$ be the length of the sequence of flight arrivals. Take $T > (3 \alpha / \epsilon) + (\beta / \epsilon L)$.

On the sequence of fly arrivals defined above:

- OPT sends one swatsman to $C$ for the first fly, and then, for the rest of the flies, send one swatsman to $A$ and one swatsman to $B$. The total cost is at most $3L$.

- The greedy algorithm, after sending one swatsman to $C$ and the other to $A$, lets the second swatsman take care of all flies in $B$ and $A$. The cost is at least $\epsilon LT$.

We chose the parameters so $\epsilon LT > \alpha \cdot 3L + \beta$, which contradicts the assumption.
Problem 4. FIFO = 2 × LIFO. [12 points]

A FIFO queue $Q$ supporting the operations ENQUEUE and DEQUEUE can be implemented using two stacks $S_1$ and $S_2$, each of which supports the operations PUSH, POP, and a test whether the stack is empty.

$\text{ENQUEUE}(Q, x)$:
1. $\text{PUSH}(S_1, x)$

$\text{DEQUEUE}(Q)$:
1. if $S_1 = \emptyset$ and $S_2 = \emptyset$
2. \textbf{error} “queue underflow”
3. if $S_2 = \emptyset$
4. while $S_1 \neq \emptyset$
5. \hspace{1em} $x = \text{POP}(S_1)$
6. \hspace{1em} $\text{PUSH}(S_2, x)$
7. \hspace{1em} \textbf{return} $\text{POP}(S_2)$

Define a potential function $\Phi(Q) = c |S_1|$ for an appropriate constant $c > 0$, where $|S_1|$ is the number of items in $S_1$. Argue using a potential-function argument that each ENQUEUE and DEQUEUE operation takes $O(1)$ amortized time.

\textbf{Solution}:

Take the potential function $\Phi$ to be $3 |S_1|$. Clearly, $\Phi$ is always non-negative and initially, $\Phi_0 = 0$. The amortized cost of each operation is its true cost plus the change in the potential. Let us evaluate it for each of the operations separately:

- \textbf{ENQUEUE}: The change in potential is $+3$. The true cost is 1. Thus, the amortized cost is at most 4.
- \textbf{DEQUEUE}: There are two cases:
  - If $S_2 \neq \emptyset$, the change in potential is 0, and the true cost is 3 (the operations in steps 1,3,7). Thus, the amortized cost is 3.
  - If $S_2 = \emptyset$, the change in potential is $-3 |S_1|$, and the true cost is at most $3 |S_1| + 4$. Thus, the amortized cost is at most 4.

In all cases, the amortized cost is at most 4.
Problem 5. Big Edges. [10 points]

Let $G = (V, E)$ be a connected undirected graph with distinct edge weights $w : E \to \mathbb{R}$, and let $c$ be a cycle in $G$. Consider the edge $e$ on $c$ with the largest weight, that is, $w(e) \geq w(e')$ for all $e' \in c$. Prove that $e$ does not belong to the minimum spanning tree of $G$.

**Solution:** Suppose for the sake of contradiction that $e = \{u, v\}$ is in a minimum spanning tree $T$ of $G$. If we remove $e$ from $T$, we divide it into two trees. This corresponds to a cut $(C, V - C)$ in $G$ such that $C$ is spanned by one of the trees and $V - C$ is spanned by the other. Since $e$ is in a cycle and it crosses this cut, there must exist another edge in the cycle, $e'$, that crosses this cut. (Any cycle must cross a cut an even number of times. To see why, follow the edges of the cycle.)

Since edge $e'$ crosses the cut, adding it in will connect the two trees thus forming a spanning tree: $T' = T - \{e\} \cup \{e'\}$.

Since all edge weights are distinct and $e$ is the edge with the largest weight on the cycle, necessarily $w(e') < w(e)$. The weight of $T'$ is therefore strictly lower than the weight of $T$, and therefore $T$ cannot be a minimum spanning tree.
Problem 6. Minimum Madness. [12 points] (4 parts)

Consider the following program to find the minimum value in an array $A$ of $n$ distinct elements.

\[
\begin{align*}
\text{MINIMUM}(A, n): \quad & \\
1 & \text{min} = \infty \quad /\!/ \text{Set min to be a large value.} \\
2 & \textbf{for } i = 1 \text{ to } n \\
3 & \quad \textbf{if } \text{min} > A[i] \\
4 & \quad \text{min} = A[i]
\end{align*}
\]

Assume that $A$’s elements are randomly permuted before invoking $\text{MINIMUM}(A, n)$ and that all permutations are equally likely. Let $X_i$ be the indicator random variable associated with the event that the variable $\text{min}$ is changed in line 4 during the $i$th iteration of the \textbf{for} loop, and let $Y = \sum_{i=1}^{n} X_i$ be the random variable denoting the total number of times $\text{min}$ is so updated.

(a) Argue that the probability that $A[i]$ is smaller than all the elements in $A[1..i-1]$ is $1/i$.

\textbf{Solution:} Notice that the probability that $A[i]$ is smaller than all the elements $A[1..i-1]$ is exactly equal to the probability that we find the minimum element of $A[1..i]$ at position $i$ (here we use that all the elements are distinct). Since $A$ is randomly permuted and every permutation is equally likely, each one of positions $1..i$ is equally likely to hold the minimum element, and the probability it lands at position $i$ is exactly $1/i$.

(b) Show that $E[X_i] = 1/i$.

\textbf{Solution:}

\[
\begin{align*}
E[X_i] &= 0 \cdot \Pr \{X_i = 0\} + 1 \cdot \Pr \{X_i = 1\} \\
&= \Pr \{X_i = 1\} \\
&= 1/i
\end{align*}
\]
(c) Prove that $E[Y] = \Theta(\lg n)$.

**Solution:** By linearity of expectation and the sum of the harmonic series,

\[
E[Y] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1) = \lg n \cdot \lg e + O(1) = \Theta(\lg n)
\]

(d) Prove that for sufficiently large $n$, it holds that $\Pr\{Y \geq 5E[Y]\} \leq \frac{1}{n^4}$.

**Solution:** We use the following *multiplicative* version of Chernoff bound: For independent Boolean random variables $X_1, \ldots, X_n$, let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then, for any $0 < \delta \leq 2e - 1$,

\[
\Pr\{X > (1 + \delta)\mu\} < e^{-\mu\delta^2/4}.
\]

For our problem, since $\mu > \ln n$, and so

\[
\Pr\{Y > 5E[Y]\} < e^{-4\ln n} = \frac{1}{n^4}.
\]

The statement of the Chernoff bound given in the beginning of the quiz is not strong enough to prove the required bound. We gave full credit to anyone who used it correctly.