Comp 3170 - Analysis of Algorithms & Data Structures

Final

Shahin Kamalli
University of Manitoba - Winter 2018

First name (please write as legibly as possible within the boxes)

DONALD

Last name

DUCK

Student ID number

1 2 3 4 5 6

"the whole purpose of education is to turn mirrors into windows."    Sydney J. Harris

Do not open this booklet until instructed.

- You can bring one-page (double-sided) of written notes to the exam. Other than that, you are not allowed to use any printed/written material, laptops/cell-phones. Please turn off your cell phones and put them in your bags.
- Manage your time. We start the exam at 1:30 and end the exam at 4:30. You have 3 hours.
- There are 13 pages (including this cover page). Write your answers in the provided boxes. In the unlikely case that you need more space, use the blank pages; if you do so, indicate it on this page (first page of the exam).
- In the unlikely case that you find the exam too long/hard, do not panic. The marks will be scaled so that the highest mark gets the full mark.
- There are more important things in life than this exam. So, relax and smile (but still manage your time).
1. Short Answer (30 marks)

Provide your short answers in the provided boxes. There is no need to justify your answers.

1. True or False: $\log_3(n^4) \in O(\log n)$.  
   [True]

2. True or False: It is possible to augment an AVL tree on $n$ nodes so that rank() and select() operations can be performed in $O(\log n)$.  
   [True]

3. Assume $T(1) = 5$ and $T(n) = 9T(n/3) + n$. Give an expression for the run-time of $T(n)$ using $\Theta$ notation.  
   $\Theta(n^2)$

4. True or false: Height of a balanced binary search tree is $\Theta(\log n)$.  
   [True]

5. True or false: Quick-select runs in $\Theta(n)$ in the average case.  
   [True]

6. True or False: Insertion time (time to insert a new item) in Binomial heaps is asymptotically less than the insertion time in ordinary Binary Heaps.  
   [False]

7. True or False: The expected height of a skip list formed by $n^2$ items is $O(\log n)$.  
   [True]

8. True or False: Binomial trees are binary trees.  
   [False]

9. True or False: Amortized cost of operations in a union-find data structure (forest structure with union-weight and path compression) is $\Theta(1)$.  
   [False]

10. True or False: if a problem belongs to class NP, then it belongs to class EXP.  
    [True]

11. True or False: If a problem $A$ is 3Sum-Hard, then there is no algorithm that can solve $A$ in $O(n^3)$.  
    [False]

12. Consider the following pseudocode:

\[
\begin{align*}
foo(n) \\
1. & \quad i \leftarrow 1 \\
2. & \quad \text{while } i < \max\{n, 2018\} \text{ do} \\
3. & \quad \text{prod} \leftarrow i \\
4. & \quad \text{while } \text{prod} < n \text{ do} \\
5. & \quad \text{prod} \leftarrow \text{prod} \times 2 \\
6. & \quad i \leftarrow i + 3 \\
7. & \quad \text{return prod}
\end{align*}
\]

What is the worst-case running time of $foo(n)$? Express your answer using $\Theta$-notation in terms of $n$.

$\Theta(n \log n)$
13. True or False: $\alpha(n) \in o(1)$. **True**

14. True or False: If a problem is APX-hard, then it is NP-hard. **True**

15. True or False: If a polynomial time algorithm is discovered for an NP-hard problem, then there will be polynomial time algorithms for all NP-hard problems. **False**

2. Selection (12 marks)

Consider a variant of Median-of-Five algorithm for Selection in which, instead of partitioning the input into $n/5$ blocks of size 5, we partition the input into $n/7$ blocks of size 7. The algorithm finds the median of each group of length 7 and recursively finds the median of these medians to set the pivot for the next recursion.

a) Derive a recursive formula for the time complexity $T(n)$ of this algorithm. Briefly justify you answer.

Assume $z$ is the selected median-of-medians. So, half of blocks have their medians smaller than $z$; each of these blocks have 4 element smaller than their median (and hence smaller than $z$). So, in total, there are at least $\frac{n}{7} \times 4 = \frac{2n}{7}$ items smaller than $z$. Similarly, there are at least $2n/7$ items larger than $z$. Consequently, the size of recursion can be at most $5n/7$. Assume $T(1) = d$. For $n > 1$, we can write

$$T(n) \leq \frac{T(n/7)}{\text{finding median of medians}} + \frac{cn}{\text{selection}} + \frac{T(5n/7)}{\text{size of recursion}}$$

b) Try to solve the recursion by guessing that $T(n) \in O(n)$. Indicate whether we can state $T(n) \in O(n)$.

Let's guess $T(n) \in O(n)$ and use strong induction to prove it. We should prove there is a value $M$ s.t. $T(n) \leq Mn$ for all $n \geq 1$. For the base we have $T(1) = d \leq M$ as long as $M \geq d$. For any value of $n$ we can state:

$T(n) \leq T(n/7) + T(5n/7) + cn$ (from above recursion)

$\leq M \cdot n/7 + M \cdot 5n/7 + cn$ (induction hypothesis)

$= (6/7 \cdot M + c)n$

Note that $(6/7 \cdot M + c) \leq M$ for any value of $M \geq 7 \cdot c$, any value of $M \geq \max d, 7c$ works. So, we could prove that $T(n) \in O(n)$.
3. AVL Trees (12 marks)

1. Perform operation \textit{insert}(9) on AVL tree $T_1$:
   Draw the tree after each rotation performed.

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) {9}
  \node (2) at (-1,-1) {8}
  \node (3) at (-2,-2) {5}
  \node (4) at (-2,-2.5) {3}
  \draw (1) -- (2);
  \draw (2) -- (3);
  \draw (2) -- (4);
\end{tikzpicture}
\end{center}

(no need to show balance factors)

\textit{one double rotation}
\textit{fixes the tree}

2. Perform operation \textit{delete}(13) on AVL tree $T_2$:
   Draw the tree after each rotation performed.

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) {15}
  \node (2) at (-1,-1) {12}
  \node (3) at (-2,-2) {6}
  \node (4) at (-2,-2.5) {10}
  \draw (1) -- (2);
  \draw (2) -- (3);
  \draw (2) -- (4);
\end{tikzpicture}
\end{center}

(no need to show balance factors).

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) {10}
  \node (2) at (-1,-1) {11}
  \node (3) at (-2,-2) {9}
  \node (4) at (-2,-2.5) {6}
  \draw (1) -- (2);
  \draw (2) -- (3);
  \draw (2) -- (4);
\end{tikzpicture}
\end{center}

\textit{Again, a double rotation is enough.}
4. Skip Lists (10 marks)

Starting with an empty skip list, insert the six keys 40, 10, 50, 60, 45, 70.
Draw the final skip list. Assume we use the following coin tosses to determine the heights of towers.
(note, not every toss is necessarily used). Recall that for each item we flip coins until we see a Tail.


\[
\begin{align*}
\infty & \rightarrow 40 \rightarrow 10 \rightarrow 50 \rightarrow 60 \rightarrow 70 \rightarrow +\infty \\
& \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
& \infty \rightarrow 40 \rightarrow 45 \rightarrow 50 \rightarrow 60 \rightarrow +\infty \\
& \downarrow \quad \downarrow \quad \downarrow \\
& \infty \rightarrow 10 \\
\end{align*}
\]

You lose 2 marks if the top level (list of $\infty \rightarrow 10$) is missing.
5. Binomial Heaps (12 marks)

1. Consider the following binomial heap.
   Show the resulting heap when we apply the operation extract-max(). In case of merging trees, in case there were three binomial trees of the same order, merge the two 'older' trees (keep the new tree which is the product of previous merge). Show your work (intermediate steps).

![Diagram of binomial heap and operations]

- Lose 3 marks if the order of 8 and 9 is not correct.
- The same for 21, 22.
2. Consider the following binomial heap.
   Show the resulting heap when we apply the operation insert(14). Show your work (intermediate steps).
6. Amortized Analysis (14 marks)

Consider a variant of dynamic arrays in which when an array becomes full, instead of doubling the size of the array, we multiply the size of the array by a factor of 10, i.e., increase its size from $x$ to the $10x$. This way, array sizes will be 1, 10, 100, 1000, ....

1. Use aggregate-cost method to find an upper bound for the amortized cost of each operation.

(note that $1 + a + a^2 + ... + a^n = \frac{a^{n+1}-1}{a-1}$)

The total cost will be $m$ (for inserting new items) plus $\sum_{j=0}^{\log_{10}(m-1)} 10^j$ (for copying old items to new arrays). It will be at most $m + \frac{10^{\log_{10}(m-1)+1}-1}{9} = m + 10(m - 1)/9 - 1/9 = 19m/9 - 11/9$. So, the amortized cost per operation is roughly $19/9 \in \Theta(1)$. You will get the full mark as long as you show the amortized cost is a constant.

2. Use the potential-function method to find an upper bound for the amortized cost of each operation. You can use the potential function $\Phi(i) = \frac{10}{9}i - \frac{1}{9}a_i$ where $a_i$ is the size of the array.

For an inexpensive operation, the size of array does not change. The difference in potential will be $\frac{10}{9}i - \frac{10}{9}(i - 1) = 10/9$. The amortized cost will be $1 + 10/9 = 19/9$ (1 for inserting the item and 10/9 for difference in potential). For an expensive operation, the size of the array changes from $a_{i-1}$ to $10a_{i-1}$. The difference in potential will be

$$(10/9 \cdot i - \frac{1}{9} \cdot 10a_{i-1}) - (10/9 \cdot (i - 1) - \frac{1}{9}a_{i-1}) = -a_{i-1} + 10/9$$

The amortized cost will be $1 + a_{i-1}$ (for inserting/moving items) plus $-a_{i-1} + 10/9$ (difference in potential) which sums up to $19/9$

You will get the full mark as long as you prove a potential which gives a constant amortized costs.
7. Disjoint Sets (8 marks)

1. Consider a union-find structure based on union-by-rank and path-compression formed by $T_1$ and $T_2$ in the following figure.

   Draw the result after the following operations: union($e, v$), find($d$).

   Get full mark for showing the final tree.
   - 2 marks for each mistake.

2. Consider a similar structure formed by $T_3$ and $T_4$ in the following figure. Draw the result after the following operations: union($p, u$), find($h$).

   For the union operation, as both trees have the same rank, assume $b$ becomes the parent of the united tree.
8. Decision Trees (10 marks)

In the nuts-and-bolts problem, we are given $n$ bolts and $n$ nuts of different sizes, where each bolt exactly matches one nut. Our goal is to find the matching nut for each bolt. The nuts and bolts are too similar to compare directly (i.e., we cannot compare two nuts and we also cannot compare two bolts); however, we can test whether any nut is too big, too small, or the same size as any bolt.

Prove that in the worst case, $\Omega(n \log n)$ nut-bolt tests are required to correctly match up the nuts and bolts.

Use an arbitrary ordering of the nuts and label them accordingly from 1 to $n$. Now, bolts can appear in any order in the input, i.e., each permutation of the numbers 1, 2, ..., $n$ forms an input to the problem. So, there are $n!$ possible instances, i.e., we have a decision tree with $n!$ leaves.

Each internal node of the tree has degree 3 because the output for each comparison has 3 possibilities. So, our decision tree has out-degree of 3 for each internal node. So, we have a decision tree of degree 3 with $n!$ leaves. The height of the tree will be at least $\log_3 n! = \frac{\log n!}{\log 3} = \Omega(n \log n)$. 
9. 3-Coloring Reduction (12 marks)

In the class, we showed that the 3coloring problem, which asks whether a graph can be colored using three colors, is NP-hard.

Prove that 6coloring problem, which asks whether a graph can be colored using 6 colors is NP-hard as well.

Create $G'$ as a copy of $G$ with an additional vertices $A, B, C$ which are connected to all vertices of $G$. $A, B, C$ should also be connected to each other and form a triangle. Adding $3n$ edges to a graph can be done in $\Theta(3n)$ in most implementations. Regardless, this reduction takes polynomial time.

Note that there might be other valid reductions. I will read your answers carefully :).

Next, we show the reduction is valid, i.e., $G$ is colorable using 3 colors if and only if $G'$ is colorable in 6 colors. First assume $G$ is 3-colorable; we should show $G'$ is 6-colorable. This is true because we can use the same coloring as $G$ for $G'$ and add three new, distinct colors for the extra vertices $A, B, C$. We have been using 6 colors to create a valid coloring of $G'$. Hence $G'$ is 6-colorable.

Next assume $G'$ is 6-colorable. Since $A, B, C$ are connected to all other vertices; their colors are different from the color of all other vertices; hence, if we remove $A, B, C$ from the colored graph $G'$, the result will be a valid coloring of $G$ with 3 colors, i.e., $G$ is 3-colorable.
10. Bin Packing (18 marks)

In the class, we showed that the 2Bin packing problem, which asks whether a multiset of items with different sizes in range \((0, 1]\) can be placed into two bins of capacity 1 is NP-complete. Show that the following problem is NP-complete.

3Bin-Packing: given a multiset of \(n\) items with different sizes in range \((0, 1]\), decide whether the items can be placed into three bins of capacity 1.

For the hardness proof, use a reduction from 2Bin packing. Recall that, to prove NP-completeness, you should show the problem is in NP, provide a reduction, and prove that your reduction is valid.

First, note that the 3Bin-packing problem is in NP. A certificate to the problem is a packing of the multiset of items into bins; we can check in polynomial time (linear time) if the total sum of items in each bin is at most 1 and that there are four bins in the packing.

We provide a reduction from 2Bin-Packing to 3Bin-packing. Given an instance of 2Bin-packing, described by a multiset \(S\) of items, we form an instance of 3Bin-packing as multiset \(S \cup \{1\}\), i.e., we add an item of size 1 to the multiset. This can be done in polynomial (constant time).

Next, we show the answer to the 2Bin-packing instance is yes iff the answer to the 3Bin-packing instance is yes.

Assume the answer to the 2Bin-packing instance is yes, i.e., we can pack \(S\) into 2 bins of capacity 1. Then, the same packing can be used to pack items of \(S\) while a new bin is used for the new item of size 1, i.e., we can pack \(S \cup \{1\}\) using 3 bins. So, the answer to the 3Bin packing instance is also yes.

Next, assume the answer to the 3Bin-packing instance is yes, i.e., we can pack \(S \cup \{1\}\) into three bins. The item of size 1 is packed lonely in a bin (because it completely occupies the space of the bin). So, removing this item from the packing decreases the number of bins by 1, i.e., we can pack \(S\) into 2 bins. So, the answer to the 2Bin-packing instance is also yes.
11. Approximation Algorithms (12 marks)

In the class, we studied an approximation algorithm which achieved an approximation factor of 2.
To approach this question, you are expected to know how that algorithm works. (To refresh your mind, it worked by processing edges in an arbitrary order and marking them.)

1. Consider the above graph $G$. Trace the algorithm to find a vertex cover for the graph. You should process edges in the order $e_1, e_2, \ldots, e_{10}$. It suffices to show the final vertex cover.

First $e_1$ is processed: \{1, 4\} are added to the solution and $e_1, e_2, e_3$ are marked. The next unmarked edge is $e_3$: \{2, 3\} are added to the solution and $e_3, e_4, e_5, e_8, e_9$ are marked. Next unmarked edge is $e_7$: \{5, 6\} are added to the solution and $e_7, e_{10}$ are marked. There is no other unmarked edge, and the solution is \{1, 2, 3, 4, 5, 6\}.

2. A smart student claims that the algorithm outputs optimal solutions when the graph is a tree (i.e., it has no cycle). Prove or disprove her claim. To prove, you should show that the algorithm's output is optimal when the graph is a tree. To disprove, you should present a tree for which the output of the algorithm is not the minimum vertex cover.

The claim is clearly wrong. A simple tree formed by two connected vertices has a vertex cover of size 1 while the algorithm selects both vertices and gives a vertex cover of size 2.