CSCB63 Assignment 3 - Summer 2021

125 Marks

Due Date: August 4th 11:59 pm on MarkUs

Note: Make sure you complete this assignment on your own without help from any resources other than class notes and textbooks. Read the guidelines on Plagiarism. https://utsc.calendar.utoronto.ca/4-academic-integrity

Submission Instructions:

MarkUs Instance: https://markus.utsc.utoronto.ca/cscb63s21/

Handwritten solutions **will not be accepted**.
Only Diagrams can be hand drawn and attach as a picture in the final document.
Late Assignment **will be** graded with 10% penalty for each late day up to 3 days (August 7, 11:59pm).
Submit following files: Assignment3.pdf

Q. 1 (5 marks)

In class we looked at a dynamic array that expanded by doubling every time it is full. We now consider a variation that expands by 1.5 instead. In other words, if the current dimension of the array is n then when full, an append operation would create a new array of size $\lceil 1.5n \rceil$ and move all the elements over appending the new element in the first available location. Use the potential method to prove that the amortized complexity of this variation is still $O(1)$.

You should define your function $\varphi(h)$ showing that is satisfies the requirements of a potential function and then calculate the amortized cost for append when an expansion does not occur and when it does occur (as we did in class).

Q. 2 (20 marks)

Design a data structure that stores integers in first in first out manner. It supports the operations of adding elements from one end, deleting elements from other and finds minimum element, each in $O(1)$ amortized time. In other words, any sequence of m operations should take time $O(m)$.

You may assume that, in any execution, all the items that get added to this data structure are distinct.

(a) [5 marks] Describe your data structure. Hint: you may use two data structure for maintaining different properties.
(b) [5 marks] Describe in words or in pseudo-code, all three operations.
(c) [15 marks] Analyze the time complexity including both the worst-case cost for each operation, and the amortized cost of any sequence of m operations.
For amortized complexity, use potential method and aggregate method. Explain your answer in detail.

Q. 3 (10 marks)

Consider the following data structure for representing a set. The elements of the set are stored in a singly linked list of sorted arrays, where the number of elements in each array is a power of 2 and the sizes of all the arrays in the list are different. Each element in the set occurs in exactly one array. The arrays in the linked list are kept in order of increasing size.

To perform SEARCH, perform binary search separately on each array in the list until either the desired element is found, or all arrays have been considered.

To insert a new element x into the set (given the precondition that x is not already in the set),

```
create a new array of size 1 containing x
insert this new array at the beginning of the linked list
while the linked list contains 2 arrays of the same size
        merge the 2 arrays into one (sorted) array of twice the size
```

(a) Draw the data structure that results after inserting the following sequence of elements into an initially empty set:

2, 63, 1, 32, 77, 7

(b) What is the worst-case time, to within a constant factor, for performing SEARCH when the set has size n? Justify your answer.
HINT: Using the following notation may help you express your answer:
"Let I_n denote the set of bit positions in the binary representation of n that contain the value 1.".

(c) What is the worst-case time, to within a constant factor, for performing INSERT when the set has size n? Justify your answer.

(d) Use the aggregate method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(log\, n)$.

(e) Use the accounting method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(log\, n)$.

Q. 4 (5 marks)

Suppose you want to find connected components using disjoint sets on undirected graph $G = \{V, E\}$, where $V = \{l, m, n, o, p, q, r, s, t, u, v\}$ and the edges in the E are processed in the order $(o,t), (q,v), (r,t), (m,r), (l,s), (t,u), (o,v), (m,u), (o,q), (r,u), (l,p)$. List the vertices in each connected component after each iteration of find-set and union. Show all the steps.

Q. 5 (15 marks)

Disprove following claim with an example:

The height of n node Fibonacci heap is always $O(log\, n)$. Use following argument to disprove the claim.

a) [10 points] Show that for any positive number of nodes n, a sequence of Fibonacci heap operations that create a Fibonacci heap that consists of one tree that is a linear chain of n nodes.

What is that sequence of operations?

Draw an example heap with the sequence of operations to demonstrate your argument.

b) [5 points] Write pseudo code for the process that will form a linear chain of n nodes.

Q. 6 (10 marks)

Recall tree implementation of disjoint sets. Suppose we want to add another operation $getset(x)$, which is given a node x and returns all the members of x's set in any order.

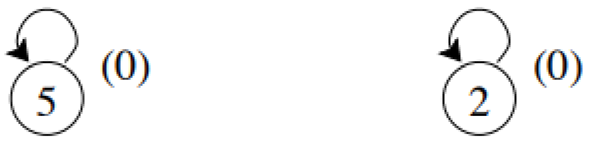
Show, how you can add a single attribute to each node in a disjoint set tree implementation so that $getset(x)$ takes time linear in the number of members of x's set and the asymptotic running time of the other operations remain unchanged. Explain your answer

Assume that returning members of each set are in $O(1)$ time.

Q. 7 (6 marks)

Let L be a circularly linked list implementation of the disjoint set ADT with extra pointers to the representative of the list and using union-by-weight. A circularly linked list is simply a list with a pointer from the tail to the head. Let T be a tree implementation of the disjoint set ADT using union-by-rank and path compression.

(a) Draw the forest T that results from executing the following operations on a tree implementation of the disjoint set ADT using union-by-rank and path compression. Assume that the forest is initially empty. Write the value stored inside each node, and beside each node, indicate its rank (the first picture is completed for you). For each `Union(x, y)` operation, when there is a choice, assume that the new representative is the representative of the set that contained x. Draw the forest after performing operations `MakeSet(5), MakeSet(2)`:



- Draw the forest after performing operations `Union(5, 2), MakeSet(3), MakeSet(4), Union(3, 4), Union(5, 3), MakeSet(1), MakeSet(8), Union(1, 8), FindSet(8).`
- Draw the forest after performing operation `Union(4, 1).`

(b) Consider a sequence of `MakeSet, FindSet`, and `Union` operations $P = P_1, P_2,, \ldots, P_n$, and suppose we perform P on each of L and T (both of them initially empty). For each `Union(x, y)` operation, when there is a choice, assume that the new representative is the representative of the set that contained x. For one operation P_i, let L_i and T_i be the number of nodes accessed when executing P_i on L and T, respectively. Let TL and TT be the total number of nodes accessed when executing P on L and T, respectively, i.e.,

$TL = \sum_{i=1}^{n} L_i$ and $TT = \sum_{i=1}^{n} T_i$

Is it possible to have $TT < TL$? Justify your answer.

(c) Is it possible to have $TL < TT$? Justify your answer.

Q. 8 (5 marks)

Suppose you put multiple orders on an online shopping website using several credit cards. The website often records transactions on your account in order of the times of transaction using all credit cards. Assume you would like to receive the statement based on transaction ordering for individual credit cards. Essentially, the problem is to convert time-of-transaction ordering to transactions based on credit card number ordering.

Out of all the sorting algorithms we have studied throughout this course, which sorting algorithm would you use? Give a valid argument in terms of complexity of chosen algorithm.

Q. 9 (5 marks)

Let positive integer n be given. Consider the following randomized algorithm, what is the expected value of the final value of t? Show your steps and/or justification.

```
c := 1
t := 0
while c ≤ n:
        t := t + 1
        if random(1, 2^c) = 1:
             c := c + 1
```

Q. 10 (9 marks)

Consider a binary search tree with *seven distinct* elements; the tree is perfectly balanced (that is, of height 2), and rooted at *root*. In this question, we will consider two slightly different methods for searching the tree for a key k. For each method, we will be interested in the *expected* time to search for k, when k is chosen at random from amongst the keys in the tree.

You may if it's helpful, for Parts (a) and (b) of this question, assume (without loss of generality) that the 7 keys in the tree are 1, 2, 3, 4, 5, 6, 7, where 4 is at the root, 2 and 6 are at depth 1, and 1,3,5,7 are at depth 2.

(a) Consider the following search method SEARCH1(root, k):

```
SEARCH1 (r,k)/*Return a pointer to a node with key k in subtree rooted
               at r.*/
if k = key(r) then
      return r
elsif k > key(r) then
      return SEARCH1(rchild(r),k)
else
      return SEARCH1(lchild(r),k)
```

Counting each "=" or ">" test as a comparison, what is the *expected* number of comparisons to do a search, where the expectation is over the random choice of k from amongst the keys in the tree, with all of them being equally likely?

Briefly explain your reasoning and show your work.

(b) Next consider the search method SEARCH2(root, k):

```
SEARCH2 (r,k)/*Return a pointer to a node with key k in subtree rooted
                at r.*/
if k > key(r) then
    return SEARCH2(rchild(r),k)
elsif k = key(r) then
    return r
else
    return SEARCH2(lchild(r),k)
```

Again, suppose that each time we do a search, the element to be sought is chosen at random from amongst the seven elements in the tree. Compute the expected number of comparisons using `SEARCH2`.

(c) What if, instead of a perfectly balanced tree of height 2, we started with a perfectly balanced tree of much larger height, say 10, and considered the same two experiments above. Which of the two procedures, `SEARCH1` or `SEARCH2`, would yield the smaller expected number of comparisons? Justify your answer.

Q. 11 (10 Marks)

Consider the abstract datatype SEQ whose objects are sequences of elements, and which supports two operations:

- PREPEND(x, S), which inserts element x at the beginning of the sequence S and
- ACCESS(S, i), which returns the i'th element in the sequence.

Suppose that we represent S by a singly linked list. Then PREPEND(x, S) takes 1 step and ACCESS(S, i) takes i steps, provided S has at least i elements.

Suppose that S initially has exactly one element. A sequence of n operations is performed. Each operation in the sequence is (independently) chosen to be a PREPEND with probability p and an ACCESS with probability $1 - p$. For each ACCESS operation, the value of the parameter i is chosen uniformly from $[1..|S|]$.

(a) Consider when $n = 2$. List the set of all possible sequences of length two and the corresponding probability that each one occurs.
(b) Derive the expected length of the linked list just before the k'th operation is performed.
HINT: You may find it useful to use indicator random variables.
(c) Derive the expected number of steps taken to perform the k'th operation.
(d) Derive the expected number of steps taken to perform all n operations.

Q. 12 (15 marks)

Suppose you are given a multi-set S of n integers and an index k such that $1 \leq k \leq n$, suggest a variation of quicksort algorithm with average case complexity of $O(n)$ to find out the kth smallest element.

(a) Give the pseudocode for the algorithm with function $Smallest(k, S)$ that finds and returns the kth smallest integer. You may add any helper function if required. If you are using other helper function, give pseudocode for that function as well.

(b) What will be the worst-case scenario? What will be the complexity in the worst-case? Explain your answer.

(c) Argue the average case complexity is $O(n)$.

Q. 13 (10 marks)

Suppose you are given an array A of n integers,

(a) Suggest an algorithm to determine whether there is a number in A that appears more than $n/2$ times.

(b) Argue that the average case complexity of the algorithm is $O(n)$.