CSCB63 Assignment 3 - Winter 2021
Due NOON, Sunday Mar. 21, 2021

Only a subset of the questions will be graded however you are responsible for all the material on this assignment.

1. 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 $\mathcal{O}(1)$.

   You should define your function $\phi(h)$ 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).

2. 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:
   $$2, 63, 1, 32, 77, 8$$

   (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 $O(\log n)$.

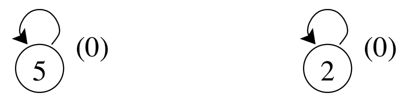
   (e) Use the accounting method to prove that the amortized insertion time in a sequence of n insertions, starting with an initially empty set, is $O(\log n)$.

3. 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 $\textsc{Union}(x, y)$ 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 $\textsc{MakeSet}(5)$, $\textsc{MakeSet}(2)$:



- Draw the forest after performing operations $\textsc{Union}(5, 2)$, $\textsc{MakeSet}(3)$, $\textsc{MakeSet}(4)$, $\textsc{Union}(3, 4)$, $\textsc{Union}(5, 3)$, $\textsc{MakeSet}(1)$, $\textsc{MakeSet}(8)$, $\textsc{Union}(1, 8)$, $\textsc{FindSet}(8)$.
- Draw the forest after performing operation $\textsc{Union}(4, 1)$.

(b) Consider a sequence of $\textsc{MakeSet}$, $\textsc{FindSet}$, and $\textsc{Union}$ 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 $\textsc{Union}(x, y)$ 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.