Romil V. Shah

20008692

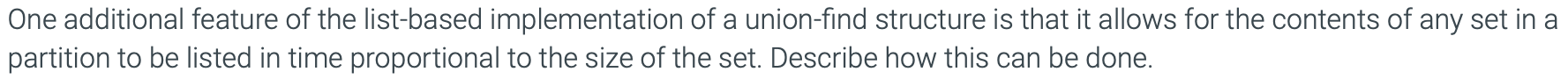
Assignment 4

CS/CPE 600

Prof. Reza Peyrovian

Submission Date: 10/09/2022

Q1. No. 7.5.5



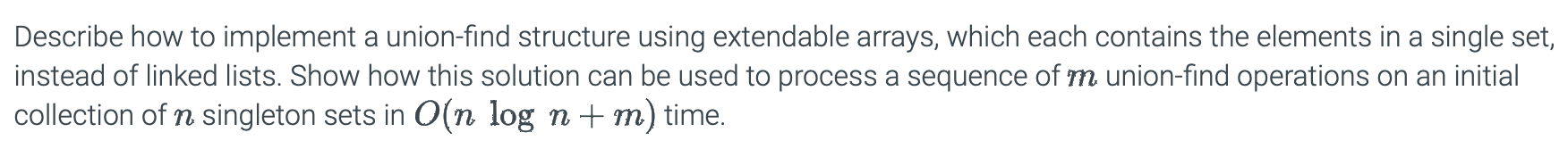
Ans.

Representing the list-based implementation using linked list will allow the contents of any set in a partition to be listed in time proportional to the size of the set.

Let us consider a list for set A, that contains the head node storing the pointer to the first and last nodes of linked list containing pointers to all the elements of A. Each node of a linked list for A stores a pointer to an element belonging to that set and pointer to the head node for A.

Constructing a set using linked list will be equal to the insertion operation for n element in linked list that is O(n) time.

Q2. No. 7.5.9.



Ans.

Given the elements in a single set, say {1}, {2}, {3} , …... , {n}, perform UNION operation on the set by merging smaller set into larger set which helps in reducing the amortized cost as the walking for all the elements occur from 1 to n in an effective way.

Consider an element x in the sequence of m union-find operation. Each time x’s element pointer changes the size of the set containing it gets double.

The size of the set containing x after n union will be 2n. Each node can have its update pointer changed at most log n times.

Thus, for all the n nodes of union-find data structure it will take O(n log n). Amortized time per union is O(log n).

In union-find structure, find and make set operation will take O(1) time.

In worst case and sequence of m, union operation will take O(n log n + m) time.

Q3. No. 7.5.21

Text

Description automatically generated

Ans.

Using Union-Find data structure, we can detect when a player wins and how many bonus points it will earn.

1. Firstly, we name all the nodes in the n \* n grid from 0 to n2 -1 and make a singleton set for the n nodes.

Example: {1}, {2}, {3}, ………{n}.

1. Placing black color and white color stone from top to bottom starting from both corners and left to right respectively.
2. Now using union operation to connect each new cell with the adjacent cell if they have same color (black or white), Starting from two end points (left-right or top- bottom)
3. Use find operation to check whether the added cell belongs to same color cell. Also check any set containing any cell on the left boundary is same as the set containing any cell on the right boundary (similarly for top-bottom).
4. If the condition holds this is the winning move, else continue with the union-find. Similarly, conditions can be checked from top to bottom.

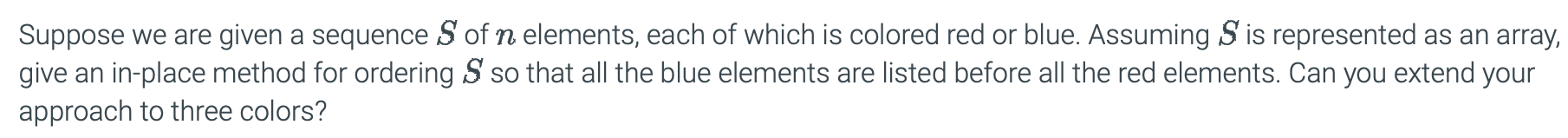
We complete the chain from one end to other (left to right or top to bottom)

After completing the chain, count the number of gold cells (k’) in the final set and that will be the bonus points for winning candidate.

Here, make set operation will take O(n)time while union operation for n elements will take O(n log n) time.

So the run time will be O(n log n) time.

Q4. No. 8.5.12



Ans.

We use an in-place (Quick-sort) method for ordering S so that all the blue elements are listed before all the red elements.

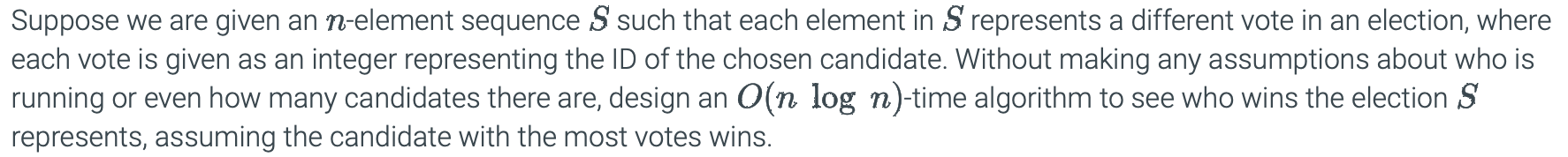
1. Take an integer ‘1’ which scan elements rightward and ‘0’ which scans elements leftward. If l is at the blue element than increment the value for l until it reached to red element.
2. If 0 is at the red element than decrement the value for 0 until it reached to blue element.
3. When l reached the red element and 0 reached the blue element, then swap the elements S[l] and S[0].
4. Recursively call this procedure until l and 0 reached to same position.

The above approach can be extended to 3 colors, by implementing the approach twice, we can solve it for three colors.

Firstly, we can solve it for one color by moving one of the colors to the left of the array and swapping other two colors at the right of the array. Now again applying the same approach to the other two colors which were moved to the right.

Starting point for this will be the ending point of the sorted color l and r be at the right most index of the array and then apply the similar approach as mentioned above.

Q5. No. 8.5.22



Ans.

We are given an n-element sequence S such that each element in S represent a different vote in election, where each vote is given as an integer.

Algorithm to check who wins the election:

a) We sort the elements in sequence S using quicksort algorithm.

b) Traverse the list and keep track of the number of votes for individual candidates. The person who receives maximum votes say ‘maxVotes’ and keep track of the count and check with other candidates in the list.

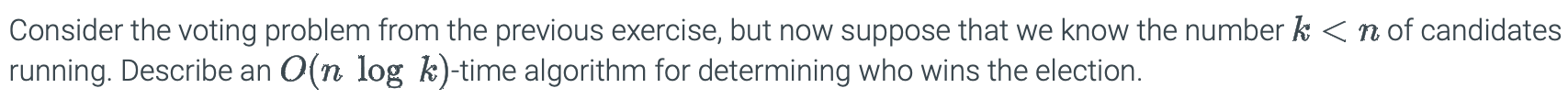
c) If the number of votes for any other individual is greater than ‘maxVotes’ then change the maximum votes with the current votes.

d) After that traverse the list, the maximum number of votes given to a particular person will wins the election.

Here, sorting of all the elements will take O(n log n) time. Traversing all the elements of sequence S will take O(n) time.

So, the algorithm will run in O(n log n) time.

Q6. No. 8.5.23



Ans.

Taking in account the previous exercise and algorithm,

a) To implement the algorithm, we use balanced search tree to store the counts of vote to respective candidate to their ID, here each tree node will represent the ID and stores the count of votes received by particular ID.

b) Initially the count for votes is 0, as we begin traversing the elements in S sequence, the count of votes will be incremented for a particular ID.

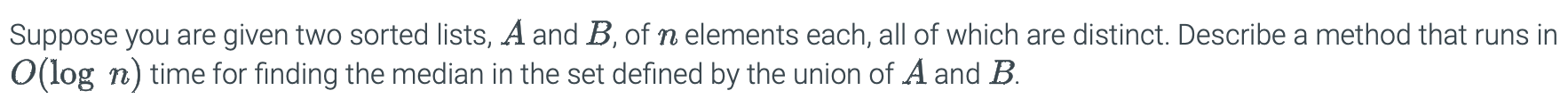
c) We go through all the nodes in the tree and find candidate ID which contains maximum number of votes.

The search and update operation in the tree for k elements will take O(log k) time.

Traversing the sequence of n elements in sequence S will take O(n) time.

So, the algorithm will run in O(n log k) time.

Q7. No. 9.5.17



Ans.

We are given 2 sorted lists, A and B of n elements each, all of which are distinct.

Method to calculate Median in the set defined by Union of A and B:

a) We start by calculating the medians medA and medB of sorted list A and B respectively.

b) If medA and medB are equal, then return either medA or medB.

c) Else if medA is greater than medB then median is present in one of the two subarrays defined below:

• From first element of A to medA (A[0] …...A[n/2]) or

• From medB to last element of B (B[n/2] to B[n-1])

d) Else if medA is smaller than medB them median is present in one of the two subarrays defined below:

• From medA to last element of list A (A[n/2] …...A[n-1]) or

• From first element of list B to medB (B[0] …...B[n/2)

e) Recursively call above process until the size of both the subarrays becomes 2.

f) Using formula to calculate the median when size of the lists become 2 is:

Median = (max(A[0], B[0]) + min (A[1], B[1])) / 2

The above algorithm will run in O(log n) time.

Example:

A = {3, 7, 13, 21, 34}

B = {4, 8, 15, 25, 35}

medA = 13

medB = 15

medA < medB (apply step d)

[13, 21, 34] and [4, 8, 15]

In above two medA and medB is 21 and 8 respectively

Now medA > medB (apply step c)

Subarray becomes [13, 21] and [8, 15]

Now, we use the formula to calculate median which we have given in Step F:

Median = (max(13, 8) + min(21, 15)) / 2

= (13 + 15) /2

= 14

Q8. No. 9.5.24

Text

Description automatically generated

Ans.

For determining if there is a student number that appears more than n/2 times in A, we must use Randomized Quick select algorithm based on prune and search paradigm.

Student number will work as the index and the value of vote stored in a particular index.

First, count the votes for a particular candidate and then apply prune and search.

Prune: In an array A containing n votes, select a random element x and partition A into three sequences.

Lt: elements that are less than x

Eq: elements that are equal to x

Gt: elements that are greater than x

Using this algorithm, depending on count of votes, candidates getting less votes will be in Lt side of x and candidates getting more votes will be on Gt side of x.

Search: Now, we recursively keep calling the quick select on Gt, which will give the result in form of name of the candidate who got majority of votes.

Prune will take O(n) to partition all the elements into three Lt, Eq, Gt. The second step will also take O(n) as it keeps calling quick select algorithm recursively.

So, the time complexity will be O(n) time.