Lebanese University
Faculty of Science II
Fanar

Year: 2020 - 2021
Course: I2206 (E)
Data Structures
Dr. Ralph El Khoury

LAB Assignments – Exam

This contain 6 pages. Total nb of points 100 pts

Remarks

1. **Team of work**: You can form a group of at most 2 persons to solve and submit your project.
2. **Project Submission:** solve the following 6 problems (the problem 7 is an extra problem, if you solve it you get extra grades) and submit a soft copy.

   a. Soft copy:

   1. A small pdf report containing:
      a. The answers of questions that does not require coding.
      b. The necessary complexities of your functions when it is asked.
      c. The necessary input and output for each problem.
   2. Each coding problem should have its own **cpp** source code file containing your program.

   The submission should be only via Microsoft Teams by uploading your project in the Assignment category with all the necessary files as a **zip** file. Only one zip file. Just in case of a problem, you can send it as a private message in Teams. And in case the first two methods are not working you can email them to my email: ralph.khoury@ul.edu.lb.

   b. The hard copy actually is not necessary, but you can prepare a hard copy for the report and a CD for the coding sticked with the report, because the faculty department may ask for them.

3. **Presentation**: No need for ppt slides to present your work, but we may fix a date to ask you some questions about your project, for all or some of you.
4. **Submission Date**:
   a. Due date is 24/06/2021, it means, submit before 24/06/2021.
   b. If submitted in [24/06/2021, 27/06/2021] you will have a penalty of 20%.
   c. No submission is accepted after 27/06/2021.
5. **Contact:** You can contact me via Teams if there are any question. I will let you know via Teams if there are any news about the project. I will not answer questions about the project after 10/06/2021.

Problem 1: Vector

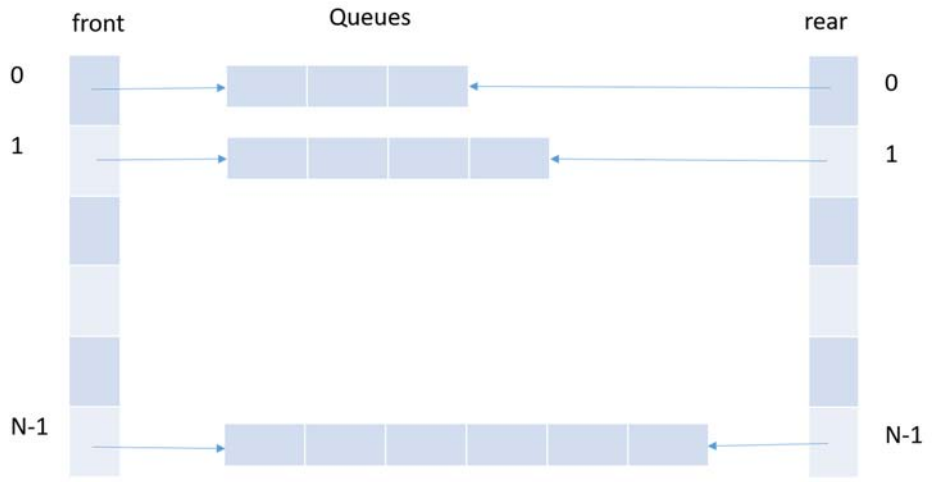
We aim to implement a dynamic ADT list based on arrays called Vector similar to the C++ vector STL. The main characteristic of a vector is that we can add many elements to the vector without worrying about the maximum limit size of the array since it is incremented implicitly when needed to store all the needed elements. In the following table, we gave a brief description about some of vector STL operations.

- Write a C++ program that implements Vector with similar operations given in the below table respecting the complexities in it and respecting the main characteristic described above.

STL vector operations	Description	Complexity
vector<int> v;	Make an empty integer vector	O(1)
vector<int> v(n);	Make a vector with n elements.	O(n)
vector<int> v(n, value);	Make a vector with n elements, initialized to value.	O(n)
v.assign(n,value)	It assigns new value to the vector elements by replacing old ones.	O(n)
v.resize(n)	Resizes the container so that it contains 'n' elements.	O(n)
v.size();	Return current number of elements.	O(1)
v.capacity()	Returns the size of the storage space currently allocated to the vector expressed as number of elements.	O(1)
v.empty();	Return true if vector is empty.	O(1)
v.front();	Return the first element.	O(1)
v.back();	Return the last element.	O(1)
v.push_back(value);	Push the elements into a vector from the back.	O(1) (average)
v.insert(position, value);	Insert value at the position indexed by position.	O(n)
v.pop_back();	Remove value from end.	O(1)
v.erase(position);	Erase value indexed by position.	O(n)

Problem 2: Queue

We aim to implement a quality of service system in a network router so that it can differentiate the type of traffic traversing it, and it can give privilege to data belonging to a given traffic compared to other type of data. So we will create N queues based on *linked list* each. You have an array of pointers to the front of queues and an array of pointers to the rear of the queues, as shown in the figure. Each node of the queues contains the *type* and *data*, and the necessary pointers for connecting with other nodes. Queue qi contains the data (and type) of traffic of type i, such that type i has a higher priority than j, if i<j. It means when the data arrives to the system, they are stored in the necessary queue according to its type. The data leaves the system according to their priority, from the higher priority to the lower one; priority i before j, if i<j. It means, q0 dequeue all its data before q1, then q1 dequeue all its data before q2, etc. qj does not dequeue its data before qi, if i<j.



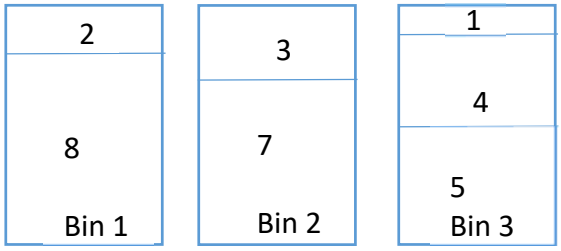
To simplify the task, the data arriving to the system are all 0. You should generate a random number ni for each qi that corresponds to the number of data arriving to the queue qi with type i. So you receive n_1, n_2, …, n_{N-1} data then you serve (output from the system) n random data (starting from the queue with high priority, it means lower type i). Repeat this reception and service many times and calculate the size of each queue and then the average size of each queue.

- Write a C++ program that solves this problem. What should be the relation between ni for all i and n so that the system will not overload?

Example: N=3; q1, q2 and q3 initially empty

Sample Input				**Sample Output**		
n1	**n2**	**n3**	**n**	**q1.size()**	**q2.size()**	**q3.size()**
3	4	5	6	0	1	5
2	3	1	8	0	0	4
4	5	3	5	0	4	7
				Average		
				0	1.67	5.33

Problem 3: Stack



Consider you have n items of sizes s1, s2, … sn. All sizes satisfy 0<si<=10. Pack these items in the fewest number of bins, given that each bin has a capacity of 10. We have an unlimited number of bins. We propose to use an approximate method to fill these items as follow: sort the items in decreasing order and then scan the bins in order and place the new item in the tightest spot among all bins instead of placing it in the first spot found. Thus, a new bin is created only when the results of previous placements have left no other alternative. Each bin is represented by a stack. You can use C++ stack STL. It is possible to use an array of stacks.

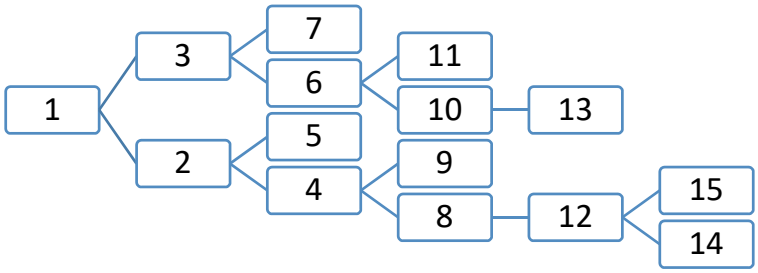
Example: consider the following 7 items: 2, 5, 4, 7, 1, 3, 8. After sorting in descending order: 8, 7, 5, 4, 3, 2, 1. We insert 8 in bin 1, then 7 in bin 2, 5 in bin 3; 4 in bin 3; 3 in bin 2; 2 in bin 1; 1 in bin 3.

- Write a C++ program that solves this problem, given n items with random size, then output the number of bins used and the content of these bins. Give the complexity of your method.

Problem 4: Tree

We need to create a genealogy tree, so we can determine the relationship between persons: cousin or descendant. Each person has at most 2 children. You are given as input the parents and their children and a sequence of requests to determine the relationship between two persons. The format of the output is:

1. A and B descendant-x
2. C and D cousin-m-n-p



The first format means that between A and B there is a relationship of descendant with distance x (i.e. the number of edges between A and B), if x=1, it means that A is parent to B, or B parent of A. If x=2, A is grand-parent of B, or B is the grand-parent of A, etc. The second format C is cousin to D, where C has a distance of m from the first common ancestor of D, where D has a distance of n from the first common ancestor with C, where the first common ancestor is p.

A sample input:

Number of persons: N = 15

Parent	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Child1	2	4	6	8	-	10	-	12	-	13	-	14	-	-	-
Child2	3	5	7	9	-	11	-	-	-	-	-	15	-	-	-

Requests:

Person 1	1	2	4	12	8	12	13
Person 2	2	12	6	7	5	9	6

A sample output:

1 and 2 are descendant-1
2 and 12 are descendant-3
4 and 6 cousin-2-2-1
12 and 7 cousin-4-2-1
8 and 5 cousin-2-1-2
12 and 9 cousin-2-1-4
13 and 6 descendant-2

- Write a C++ program to solve this problem, given a random input from a text file as shown in the sample input, including requests, then give the output as shown in the sample output. Give the complexity of your program.

Problem 5: Sorting

1. We have an array with n elements containing 0, 1 and 2. Write a C++ program that sorts the array in linear complexity.
2. You are given the ages of n persons in a country, where ages are integers in the interval [1,26]. Write a C++ program to sort the ages with a best possible complexity. What is your complexity?
3. Consider you have a list of n string of characters, for example: cab, bcd, axz, xwy, mpo, dcv.
    a. Consider that the length of all strings are same and equal to m. In this example, the length of each string is 3. Write a C++ program that modifies the mergesort or quicksort to sort this list. What is its complexity?
    b. Can you find a sorting method to sort an array of strings (with length m each) with better complexity? If yes, write a C++ program to implement it. What is its complexity?

Problem 6: Heap

Consider a minimum priority queue PQ implemented using binary min-heap tree. This PQ is used for a printer to print tasks (documents). A printer receives many tasks to be printed. Each task has a priority, an ID and a duration (number of unit time needed to be printed). Consider you receive n tasks into the printer. Tasks with low value of priority are printed before tasks with high value of priority. You wish to calculate the necessary time for your task, having a given ID, to be totally printed.

- Write a C++ program to implement the solution with n random tasks as input. What is the complexity of your program? Draw the binary min-heap for the example below, and redraw it when you remove the first task.

Example, you receive n = 6 tasks with a form (priority, ID, duration): (2,100,3), (5,40,4), (1,20,2), (4,30,5), (5,200,4), (3,50,4). Your ID is: 30. The necessary time to print your task is: 14.

Problem 7: Hashing – Extra

We aim to store in a hashing table a list of words, it means a small dictionary, and verify if a given text or paragraph has all its words written well according to this small dictionary. Your program takes as input a text file containing the list of words to add to the hashing table, and in the output you should display the words that does not correspond exactly to the words in the dictionary.

- Write a C++ program that implements a hashing table and solve the problem. What is the complexity of your program?