1 Introduction and Objective

This tutorial marks the end of the first $\frac{1}{3}$ of CS2040C: Basic C++, basic analysis of algorithms (worst case time complexity only), various sorting algorithms, and various linear Data Structures (DSes).

This tutorial also marks the start of the next $\frac{1}{3}$ of CS2040C: Various non-linear DSes. We start by reviewing a bit of CS1231 topics of graphs and trees, discuss the Priority Queue (PQ) ADT with its Binary Heap implementation (use https://visualgo.net/en/heap to help you answer some questions in this tutorial).

Finally, we will have a 15% midterm test about the first $\frac{1}{3}$ of CS2040C + a little bit of Binary Heap data structure. Thus we will spend the remaining time of this tutorial time to discuss questions that have appeared in Steven’s past relevant midterm test and thus will not appear again verbatim in CS2040C midterm test this semester.

2 Tutorial 04 Questions

CS1231 Review (Focus on Trees)

Q1). Prove or disprove: “There is a unique path between any two distinct vertices of a Tree (a connected acyclic undirected graph)”.

Q2). Prove or disprove: “In a complete binary tree with $N$ vertices, the number of vertices with degree 2 (or more) is greater than $N/2$”.

Basic Binary Heap Stuffs

Q3). Quick check: Let’s review all 5 basic operations of Binary Heap (use the Exploration mode of http://visualgo.net/en/heap). During the tutorial session, the tutor will randomize the Binary
Heap structure, ask student to Insert(random-integer), perform ExtractMax() operations (or the first few steps of HeapSort()), and/or the \(O(N \log N)\) or the \(O(N)\) Create(from-a-random-array).

More About Binary Heap Data Structure

Q4). What is the minimum and maximum number of comparisons between Binary Heap elements required to construct a Binary (Max) Heap of arbitrary \(n\) elements using the \(O(n)\) Create(array)? Note that this question has been integrated in VisuAlgo Online Quiz, so it may appear in future Online Quizzes :) .

Figure 1: Now automated :)

Midtest Preparation

Q5). Two countries A and B are at war. You are the army general of country A and has a strong belief in the effectiveness of a tight battle line formation. You have a troop of \(N\) (\(1 < N < 1000000\)) soldiers (indexed from 0 to \(N-1\), initially all are alive of course) that you have put in one long and tight battle line: \(\{0, 1, 2, ..., N-1\}\), where soldier 0 has no one on his left and soldier \(N−1\) has no one on his right. The value \(N\) is given in the first line of the input. Then you send this troop to fight country B’s army.

Over the course of the war, there are \(G\) (\(0 < G \leq N\)) groups of soldiers who are killed, which is described in \(G\) casualty reports. The value \(G\) is given in the second line of the input. As your troop forms a tight battle line, every time a group of soldiers is killed, their indices happen to be contiguous. That is, a casualty report is simply a pair of two integers \((L, R)\) \((0 \leq L \leq R \leq N-1)\) that describes that your soldiers with index from \(\{L, L + 1, ..., R - 1, R\}\) are all killed simultaneously :(. As you have highlighted the importance of maintaining tight battle line, your soldiers with index \(L - 1\) (if exists, as \(L\) can be index 0) and \(R + 1\) (if exists, as \(R\) can be index \(N-1\)) must quickly ‘close the gap’ and march forward again. These \(G\) casualty reports are valid reports, i.e. once a soldier has perished, they will never be mentioned again in future casualty reports. The \(G\) casualty reports arrive to you chronologically.

Now your job is to design a C++ program that you can use to immediately order soldier \(L - 1\) (if exist, or report -1 otherwise) and \(R + 1\) (if exist, or report -1 otherwise) to close the gap to maintain tight battle line. For example, if given \(N = 10\) (soldiers) in the first line, \(G = 3\) (casualty reports) in the second line, and three pairs of casualty reports: \((4, 5), (0, 3), (7, 7)\) in the next three lines, your program must quickly outputs \((3, 6), (-1, 6),\) and \((6, 8)\), respectively in three lines, i.e.:
To help you understand the sample test case, here are the states of your troop for that test case:

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3 6</td>
</tr>
<tr>
<td>3</td>
<td>-1 6</td>
</tr>
<tr>
<td>4 5</td>
<td>→ 6 8</td>
</tr>
<tr>
<td>0 3</td>
<td></td>
</tr>
<tr>
<td>7 7</td>
<td></td>
</tr>
</tbody>
</table>

0123456789 // initial state, you have 10 soldiers, with 3 casualty reports
01236789   // after 1st report (soldiers 4 and 5 perish), 3 and 6 close gap
6789       // after 2nd report (soldiers 0, 1, 2, and 3 perish), 6 is the leftmost
689        // after 3rd report (soldiers 7 perish), 6 and 8 close gap

You can stay back and ask the tutor about any other questions that you may have to prepare you for the more challenging Midterm Test on Friday, 23 February 2018, 17.05-18.05.

**Problem Set 2**

We will end the tutorial with discussion of near-deadline PS2.