BLG335E, Analysis of Algorithms I, Fall 2014, Project 2

Handed out: 28 October 2014

Due: 19 November 2014, 11.00 P.M.

QUICK SORT AND PRIORITY QUEUES

Submit the homework through Ninova. You must submit your all program and header files. Put the code files for part 1 and 2 into separate directories named Q1 and Q2. You must also submit a softcopy of your report. This is an individual project. You must write your own code for each part. Do not copy code from the internet or from your friends.

1. (Quick Sort – 20 points)

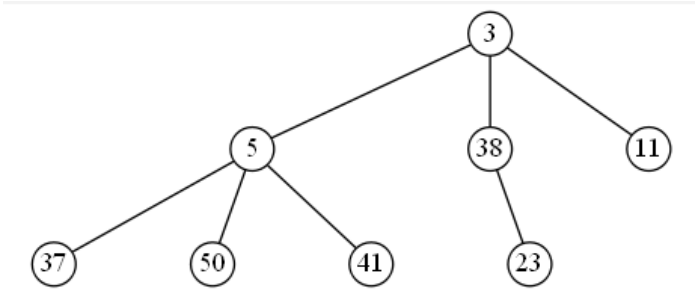
In this part, you are asked to analyze the average and worst case analysis of the Quick Sort algorithm. Based on the lecture notes, implement the Quick Sort algorithm in C++. For all parts, you may choose the pivot value as in the lecture slides (refer to the PARTITION method). Your program should write the sorted array to a file named sorted.txt.

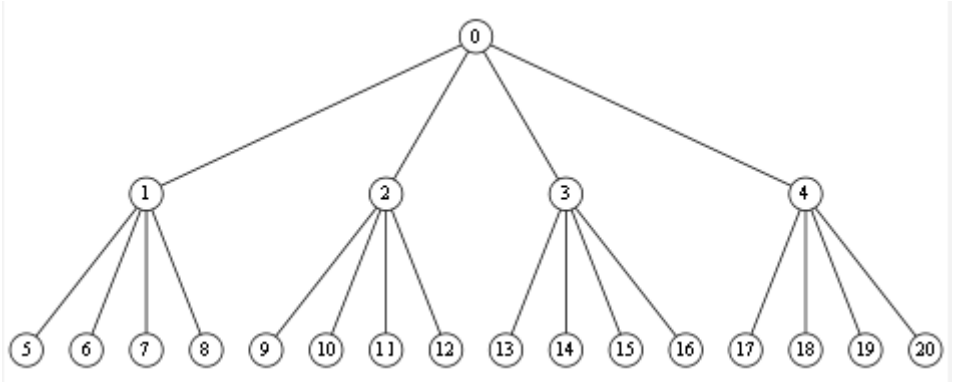
a. (10 points) Worst case running time of Quick Sort:
   i. Analyse the complexity for the worst case (give the recurrence for this case and solve it).
   ii. Report the worst case running time of Quick Sort algorithm with arrays you generated, of size N = 100, 1000, 10000, 100000, 1000000

b. (10 points) Average case running time of Quick Sort:
   i. Analyse the complexity for the average case (give the recurrence for this case and solve it).
   ii. Report the average running time (in clicks) of Quick Sort algorithm with randomly generated arrays of size N = 100, 1000, 10000, 100000, 1000000. For every N, generate 100 random arrays and report the average running time in clicks. Comment on the results that you got in both parts.

2. (Priority Queues – 80 points)

In this part you will build a priority queue (PQ) for a simulation using the d-ary heap data structure. Priority queue is an abstract data type (ADT). A d-ary heap is a generalization of a binary heap, where each node has d children instead of 2. Examples of d-ary heaps for d = 3 and 4, respectiely, are shown in the figure below respectively.





A university uses an online application system to admit students. In this online system, applicants must enter their exam scores for English and Math and also their GPAs. The final score for each applicant is computed as:

Final score = 0.25 * English score + 0.30 * Math score + 0.45 * GPA

The registrar's office reviews the remaining applications and may update the scores of applicants. Applicants with extra successes like degree in competitions, academic publications etc. can get extra scores between **1** and **10**. Applicants which are out of field can lose points between **1** and **10**. Registrar's office updates scores by adding or subtracting points between **1** and **10** to the final score and they take the N best scores from the applicant list and invite them to interview. Applicants may also withdraw their own application.

You need to implement a PQ using **d-ary heap** data structure that supports the following operations: **[30 pts.]**

- New applicants can be added to the heap **[INSERT]**
- Some applicants can be removed from the heap **[REMOVE]**
- Some applicants' scores can be updated (increase or decrease key) **[UPDATE_KEY]**
- Applicant that has the best score can be removed from the heap **[EXTRACT_MAX]**

The output must contain the following:

- Number of total applicants
- Number of updates
- Number of applicants who will be invited to the interview

- Total running time in milliseconds

You need to develop a simulation for this application system with the following features:

- Best **N** applicants will be removed after **M** operations, if N is bigger than the number of applicants all applicants will be removed.
- Each operation is an update with a probability of **P** and addition of new applicant with a probability of **1-P.**
- If an operation is update, it updates a random applicant with a random increment between **-10** and **10**.
- Otherwise it is a new application and its final score are read and calculated from the provided input file.
- For each simulation step, a random applicant withdraws his/her application with a probability of 0.2.
- Simulation stops after **M** add, update or remove operations.

a. Explain the implemented PQ operations and corresponding simulation features with their theoretical running times (consider the d value). Explain how you find child and parent indices of a node in a d-ary heap. **[20 pts.]**

b. A graph demonstrating the effect of **M** choice on the running time. You should run the simulation for different values of **M** values between 1000 and 100000 for constant **N**, **P, d = 3**. Comment on the results considering the theoretical running times. **[10 pts.]**

c. A graph demonstrating the effect of the **P** choice on the running time. You should run the simulation for different values of **P** {0.1, 0.2, …, 0.9} for constant **M**, **N, d = 3**. Comment on the results considering the theoretical running times. **[10 pts.]**

d. A graph demonstrating the effect of **d** choice on the running time. You should run the simulation for different values of **d (d = 3 and d = 4)** for constant **M, P, N**. **[10 points]**

Your program should run from the command line with the following format:

./studentID_AoA1_P2 d M N P

All your code must be written in C++ using object oriented approach and it should compile and run on Linux using g++. It shouldn't require any external libraries including STL.