CS 014: Introduction to Data Structures and Algorithms
Fall 2017
Lab 7

Objectives

In this lab, you will implement different algorithms for the selection problem and compare them both analytically and experimentally.

Deliverables

- (10%) attending the lab on Monday.
- (40%) The final source code.
- (5%) You must adhere to the following submission format. You need to submit a single file on iLearn named "CS014_lab7_<UCR Net IDs>.zip" where <UCR Net IDs> are the students UCR Net ID separated by underscores. The ZIP file should contain the source code in .cc and .hh files. The ZIP file should also contain a PDF report that contains the filled-in table, the log-log plots, and the answers to the questions below. On the cover page, your report should mention:
    - Your TA name.
    - Your lab section number.
    - You names and UCR Net IDs.
- (25%) The PDF report that contains the answers to all the questions below.
- (20%) The in-lab assignment on Monday 11/20.

Groups

- This lab will be done in groups of two to three members. Deliverables from groups of smaller or larger sizes will receive no credits except with an instructor approval prior to the beginning of the lab.

Due date

- The deliverables are due on Tuesday 11/21 by 11:59 PM Pacific Time. However, you are highly encouraged to deliver it during the lab on Monday 11/20 to save your time.

Problem definition

Given an unsorted array A of size n and an integer $k \in [0, n[$, you are required to return the kth smallest item in the list. That is, you need to return the value x such that there are exactly k items in the array A that are less than x. You can assume that the array has distinct elements.

The four algorithms that you are required to implement are briefly described below. If you have any questions about any of them, you can ask the TAs or the instructor during the office hours.

1. Select1: This algorithm scans the array k times. In each scan, it selects the minimum element in the array that has not been previously selected. That is, at the first iteration, it selects the smallest element. At the second iteration, it selects the second smallest element. At the i^{th} iteration, it selects the i^{th} smallest item. After running k iterations, it returns the k^{th} smallest element in the array.
2. Select2: This algorithm simply sorts the entire array and returns the k^{th} element in the sorted order. You are allowed to sort the array in place and return the k^{th} element afterward. For simplicity, you are allowed to use the STL sort function.
3. Select3: This algorithm uses a heap to keep track of the k smallest elements in the array. After scanning the entire array, it returns the k^{th} smallest element in the heap. You are *allowed* to use the three following STL heap functions, make_heap, push_heap, and pop_heap.
4. Select4: This algorithm runs a variation of the Quick Sort algorithm to select the k^{th} smallest element in the array. Initially, it selects the pivot using the median-of-three method. Then, it partitions the array around the pivot in the same way done in Quick Sort. After that, it recursively processes the side of the array that will contain the k^{th} smallest element. For example, if the pivot ends up at position 55 while the required value of k is 35, it recursively processes the subarray [0, 55]. Notice that you are NOT allowed to use recursion in implementing this algorithm.

Detailed steps

1. Create a workspace in Cloud9 for this lab and set the value 'https://github.com/aseldawy/CS014-Lab7.git' in the field 'Clone from Git or Mercurial URL' when you create a Cloud9 workspace. Instead, you can download the source code from iLearn and upload it to an empty workspace in Cloud9.
2. You will find placeholders for four select functions. Implement the four of them according to the description above.
3. Analyze the worst-case asymptotic running time for the first three algorithms. For the fourth algorithm, analyze its best-case asymptotic running time which is similar to the average case for this specific algorithm.
4. The main function has three parts.
    a. The first part runs a few tests to test the correctness of your code. Feel free to extend these tests to ensure a better coverage for the corner cases.
    b. The second part runs the four algorithms for different values of k while n is kept constant while measuring their running times.
    c. The third part runs the four algorithms for different values of n while k is kept constant. The running times are measured as well.
5. According to the asymptotic running times you analyzed above, how do you expect the four algorithms to behave while changing k and n? Make sure to answer this question before running the algorithms and collecting their running times. All attempts will be considered correct for this specific question.

6. After you are confident that your implementation is correct, run the program and collect the running times when k and n are changed. Write down all the numbers in the report in a tabular format.
7. Fill in the attached Excel Spreadsheet and watch the plots being updated as you enter the numbers. Copy both plots to your report.
8. Compare the actual running times to your analysis in question 4. Do they match? Explain the results of the plot based on the asymptotic running time analysis.