Algorithms* Lab 4

Review

Topics covered this week:

- sorting lower bound, non-comparison based sorting (bucket sort, counting sort, radix sort)

In lab exercises: Collaboration level : 0

1. Argue that Quicksort is not stable by showing a small example.

2. Are the following sorting algorithms stable:

   (a) Bubblesort:

   (b) Insertions ort:

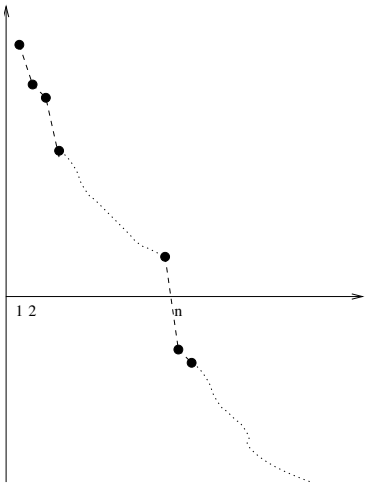
   (c) Selection sort:

   (d) Mergesort:

   (e) Heapsort:

*csci2200, Laura Toma, Bowdoin College

3. Consider a *monotonically decreasing* function $f : N \to Z$ (that is, a function defined on the natural numbers taking integer values, such that $f(i) > f(i+1)$). Assuming we can evaluate f at any i in constant time, we want to find $n = \min\{i \in N | f(i) \leq 0\}$ (that is, we want to find the value where f becomes negative).



We can obviously solve the problem in $O(n)$ time by evaluating $f(1), f(2), f(3), \ldots f(n)$. Describe an $O(\log n)$ algorithm.

(*Hint*: Evaluate f on $O(\log n)$ carefully chosen values between 1 and $2n$ - but remember that you do not know n initially).

Homework: Counting sort or Quicksort?

Use $\{Java$ or$C++\}$ to implement, test and compare Quicksort and CountingSort on randomly generated integers in a given range. Run your code on inputs of sizes $n = 1,000$, $n = 10,000$, $n = 100,000$, $n = 1M$, $n = 10M$, $n = 100M$ and in the range $K = 1,000$, $K = 100,000$, $K = 100M$ (each combination). Record the running times in each case in a table and write a brief report with your findings.

Comments, etc

- Ideally, both the size of the array and the range should be specified on the command line. Having them as constants in your code is another alternative, but less preferred.
- Write a single source file. The general outline is the following: declare and allocate an array of the required size; initialize the array with random integers in the given range; call a function to sort it using CountingSort; check that it's sorted; call another function to sort it using Quicksort; check that it's sorted; print the running times.
-

```
/*
File: sorting.cpp
Authors: Eric and Duncan
Date:
*/

//quicksort in place
//assume array is populated with data prior to this call
void quicksort(array a);

//countingsort
//sorted result is returned as a new array
//assume array is populated with data prior to this call
array countingsort(array a);


//return 1 if a is sorted (in non-decreasing order);
//return 0 otherwise
//assume array is allocated and populated with data prior to this call
int issorted(array a);


int main(char** argv, int argc) {

   //declare an array a

   //populate the array with random data

   //call CountingSort to sort a
```

```
    b = countingsort(a);

   //test that array is sorted
    if (issorted()) {
          printf("testing counting sort: passed\n");
    } else {
         printf("testing countingsort: failed\n");
    }

    //call quicksort
    quicksort(a);

    //test that array is sorted

}
```

- Time and print the time it takes to sort. This should not include the time to populate the input with random values, or the time to test that it's sorted.
- You may get errors when allocating such large amounts of memory. Its' worth experimenting with static vs dynamic allocation. Different compilers.languages may have different limits, so expect different results with Java vs C++.
- To compile from the command line, you can use

  ```
  [ltoma@lobster ~]$g++ sorting.c -o sorting
  ```

  In Java:

  ```
  [ltoma@lobster ~]$javac  sorting.java
  ```

Pair-programming honor code

For this lab you will work with one partner and do pair-programming. The honor code for pair-programming is the following:

1. You must work together, physically in the same place.
2. The overall lab must be a true joint effort, equally owned, created and understood by both partners.
3. Specifically splitting the lab into parts and working on them separately is not allowed and violates the honor code for the class.
4. If you start together as a team and are unable to complete the project together, then you will each inherit the shared code, and split up to work individually on the remainder of the project. You would then submit individually, with a note on what happened.
5. The two partners in a team obviously share everything, obviously. Across teams, collaboration is allowed at COLLABORATION LEVEL: 1.

What to turn in

1. Submit the code (details later).
2. In addition to the file, submit a brief report containing the table with the running times and a brief discussion of yoru results.
3. Bring hard copies of your code of your report, one per team, stapled.