Homework 2
Com S 311
Due: Feb 4, 11:59PM

Late Submission Due: Feb 5, 11:59PM (25% penalty)

There are 4 problems each problem is worth 60 points. When asked to write an algorithm, please give the pseudo code. Do not write Java/C code.

1. This problem asks you to compute worst-case run times of given algorithms.

   (a) Consider the following algorithm

   ```
   r = 0;
   for i in the range [1, n] {
      for j in the range [i, n]
          for k in the range [1, j-i]
              r++;

    print (r);
   }
   ```

   Derive the run time of the above algorithm (as a function of n). You must formally *derive* the run-times (merely stating run times or high level explanation of run time do not suffice).

   (b) Consider the following algorithm.

   ```
   for (i = n; i >=1; i = i/2)
     for j in range [1, i]
        Constant Number of Operations
   ```

   Derive the run time of the above algorithm (as a function of n). You must formally *derive* the run-times (merely stating run times or high level explanation of run time do not suffice).

2. This problem on sorting algorithms.

   (a) Consider the following insertion sort algorithm.

```
Input: Array a of size n, consisting of integers.

for i in range [1, n-1] {
   j = i;
   while (j > 0 AND a[j-1] > a[j]) {
       swap(a[j], a[j-1]);
       j = j-1;
   }

}
```

   Derive the worst-case and best-case run-time complexities of the above algorithm. What kind of inputs lead to wort-case complexity? What kind of inputs lead to best-case complexity? You must formally *derive* the run-times (merely stating run times or high level explanation of run time do not suffice).

   (b) Implement Selection Sort, Bubble Sort, and Insertion Sort algorithms in Java. Calculate and compare physical runtimes of your sorting programs on arrays of various as follows.

   - Create a sorted array of size n, run all three algorithms and report run times for $n = 3000, 30000$ and 300000.
   - Create a reverse sorted array of size n, run all three algorithms and report run times for $n = 3000, 30000$ and 300000.
   - Create a random array of size n, run all three algorithms and report run times for $n = 3000, 30000$ and 300000.

   You may use `System.currentTimeMillis` to calculate the run times. Please **Do not submit** the Java code. Just report run times.

3. This problem is on determining whether one function is Big-O of the other. Provide a proof for your answers. If you are showing that if $f \in O(g)$, then you must show that there is a constant $c > 1$ such that for every n, $f(n) \le cg(n)$. Your proof must state the value of the constant c. If you are showing that $f \notin O(g)$, then you must show the for every constant $c > 1$ it is not the case that $f(n) \le cg(n)$. In your proofs, you may use the fact that for every $k, a > 0$, $n^k \in O(n^{k+a})$, and $\log^k n \in O(n^a)$.

   - Is $48n^4 - 46n^2 + 25n + 31 \in O(n^4)$?
   - Is $n^{\log n} \in O(2^{\sqrt{n}})$?
   - Is $2^{2^{n+1}} \in O(2^{2^n})$?
   - Is $n^3(5 + \sqrt{n}) \in O(n^3)$.

4. For a positive integer k ($2 \le k \le 9$) a *base k-number* is a sequence of digits $d_1d_2 \cdots d_n$ such that $0 \le d_i \le k - 1$ for every i ($1 \le i \le n$). Given a base k-number $d_1d_2 \cdots d_n$, its decimal equivalent is

$$d_n + d_{n-1}k + d_{n-2}k^2 + \cdots + d_1k^{n-1}$$

a) Give an algorithm that gets k and a base k-number $d_1 d_2 \cdots d_n$ as inputs, and outputs its decimal equivalent. Analyze the worst-case (asymptotic) run time of your algorithm. Your grade depends on the run time of the algorithm. Note that *computing exponent* operation is not a primitive operation (and thus is can not be done in constant/$O(1)$ time). Note that run time should be expressed as function over the *size of the input.* Size of $d_1 d_2 \cdots d_n$ is n. You may assume that $d_1 d_2 \cdots d_n$ is represented as an array, with d_1 being the first element of the array.

b) Give an algorithm that gets a decimal number m and and integer $(2 \leq k \leq 9)$ as input, and outputs m's base k-equivalent number. Analyze the worst-case (asymptotic) run time of your algorithm. Your grade depends on the run time of the algorithm. Note that run time should be expressed as function over the *size of the input.* Size of an integer m is number of bits needed to represent m.

GUIDE LINES:

- Please write your recitation number, time and TA name.
- It is important to know whether your really know! For each problem, if you write the statement "I do not know how to solve this problem" (and nothing else), you will receive 20% credit for that problem. If you do write a solution, then your grade could be anywhere between 0% to 100%. To receive this 20% credit, you must explicitly state that you do not know how to solve the problem.
- You must work on the homework problems on your own. You should write the final solutions alone, without consulting any one. Your writing should demonstrate that you understand the proofs completely.
- When proofs are required, you should make them both clear and rigorous. Do not hand waive.
- If you hand writing is not legible, then your homework will not be graded.
- Any concerns about grading should be made within one week of returning the homework.
- **Please submit your HW via Canvas. If you type your solutions, then please submit pdf version. If you hand-write your solutions, then please scan your solutions and submit a pdf version. Please make sure that the quality of the scan is good, and your hand writing is legible. Name your file must be *YourNetID-HW2.pdf*. For example, if your net id is `potterh`, then the file name should be *potterh-HW2.pdf*. HW's submitted in incorrect format (non pdf, incorrect file name etc) will incur a penalty of 20%**