CS 4310 - Algorithms - Spring 2022

HomeWork1

Given: January 20, 2022
Due: February 3, 2022

General Comments: *Show all your work, otherwise no partial credit. No credit without proper justifications. State all your assumptions. No Algorithm is complete without its time and space complexity. When presenting an algorithm, first indicate if it is similar to a well-known algorithm, second describe intuitively how the algorithm works (which may be supported by examples), third give its pseudo code and finally analyze its time and space complexity. Always describe general idea of the algorithm before giving its pseudo-code. Do not reinvent the wheel, i.e., if a well-known algorithm can be modified to solve a problem efficiently, use that solution and clearly indicate the changes required. Do not unnecessarily complicate a solution, i.e., if a simple but efficient solution exists then we should use it. Finally, if you just write pseudo-code of a well-known algorithm without indicating how it applies or modified to the problem at hand, no credit will be given even if its correct.*

1. (20pts) For the code snippets below, find time and space complexities. First derive the time complexity using summation notation, then find the close form and finally express it using tight asymptotic (big-Oh, big-Theta, big-Omega) notation. Assume constant $c_1>0$ overhead per iteration for a loop. **Don't bury / hide lower order terms or all constants** into one until the last step of expressing your solution using tight asymptotic notation. First, list all the basic instructions you are counting and then do the frequency counting.

```
a.  acc=10 ;
 for(i=5; i ≤ n/2; i++)
      for(j=2*n; j ≥ 1; j--)
           acc -= j + acc++ ;
```

```
b.  for(i=2; i≤n; i+=2);
```

```
c.  for(i = 1; i ≤ n; i *= 3)
        x = y^2;
```

```
d.  y=0;
    for (i=0; i<n; i++){
       y -= 2;
       if (n%3 == 0)
          for (j=0; j≤2m; j++) x++;
       else
          for (j=0; j>m; j++) x--;
      }
```

```
e.  for(i=1; i≤n; i++)
    {
       for(j=1; j≤ i; j++)
      {
          for(k=n; k>j; k--)
             x=x+5; y /= 3;
          }
      x = x+y-i;
       }
```

2. (20pts) This question concerns the asymptotic relations between functions; you can assume that all logarithmic functions are in base 2. It's a repeat from general comments above, but **you must justify your answers**, otherwise no credit. Sort the following functions in an asymptotically non-decreasing order of growth using big-oh and big-theta notations; in other words, provide a ranking of these functions such that g1 = O(g2), g2 = O(g3), g3 = Θ(g4), g4 = O(g5), . . .

   (sqrt(4))^logn, 2^(sqrt(n)), (100)*n log(2n), n^2, (n * 2^(n/2)), (2/n)*n!, 8^logn, log(n^1500000), (log n)^2, 2^100000, 2^(4^n)

3. (20pts) Suppose you are given a new hardware device that can merge d > 2 different sorted lists of total size n into a single sorted list in O(n) time, independent of the value of d. Such a device could, for example, be based on a hardware streaming system or could be based on a network protocol. Show that you can use this device to sort items efficiently by answering the following. (3.a) Suppose that you are given d sorted lists where list *i* has length n_i with $m = \Sigma\, n_i$. Design and analyze an efficient algorithm to sort the m items using the hardware device, i.e., your algorithm uses the hardware device as a method call mergeDSortedLists(L[][], d, m) to sort the m items, where L is a d X m two-dimensional array containing a total of m items and L[i] contains a sorted list of length n_i. Function mergeDSortedLists(…) returns a 1-dimensional sorted array of length m. What is the input size of this problem? What is the output-size of this problem? What are the time and space complexities of your algorithm? (3.b) Suppose now that you are given a list of n items (unordered). Show that you can use the hardware device to sort n elements in O(n log n / log d) time. Notice that, if d is $\Theta(\sqrt{n})$, then you can use this device to sort in linear time. Obviously, your algorithm will use method calls mergeDSortedLists(…) judiciously. What is the input size of this problem? What is the output-size of this problem? What are the time and space complexities of your algorithm?

4. (20pts) **(4.a)** Algorithm A uses 2n log n + 5n operations, while algorithm B uses n^2 operations. Determine the value n_0 such that A is better than B for $n \geq n_0$. **(4.b)** Are there values of n such that B is better than A? If yes, list the ranges. If not, justify why not. **(4.c)** Determine the value n_0 such that A is better than B for $n \geq n_0$, when B uses n^3 operations.

5. (20pts) Solve the following recurrence relation

   T(n) = 2*T(n/2) + 4*n^2. if n>=2,

   = 8 otherwise.

First find a closed form using expansion (aka substitution) method and then express your solution using tight asymptotic notation. These kind of recurrence relations arise in solutions when manipulating matrices.

General Instructions on submitting your homeworks.

- For assignments, submit a SINGLE zipped file of your source codes, scripts (to run your program if any) and a brief report along with a copy of a couple of sample executions of your solution to the class's Elearning. No need to say, but you should be using good conventions and programming practices in developing your programs [just in case you forgot, refresh them from some of the coding conventions etc links provided on the TopicsCovered page.]
- Use <hw#cs4310_yourlastname_mmddyy.{zip,ppt,doc,tex}> as the naming convention for your zipped, ppt, MS-Word, or LaTex files when submitting on Elearning. Replace '#' with the appropriate homework number.
- There will be significant point penalties for not following the naming convention above, good coding practices, submitting a different format of archive file or if your program does not run. Make sure it is a .zip and NOT another format (no .rar, .tar, .tar.gz, etc)
- Beginning of the first file should clearly identify you, the class, submission date, and the main goals of the homework / programming assignment. Also **add credit to others if you had to resort to looking up the solution elsewhere**. Finally add one of the sentences: " I give permission to the instructor to share my solution(s) with the class." or " I do NOT give permission to the instructor to share my solution(s) with the class."

- Attach Plagiarism-free declaration.

Any student may be asked to show and discuss his or her solution in class, so be ready with your presentation.