Assignment 2 CS395-S13 Due: Mon Feb 11, 12:30pm

Name: _____ Points: 225

Remember no late papers and so always turn something in for partial credit. You can turn in paper copy before class on the due date or as a pdf through the website any time up to the due date.

1. (30 pts) Assume you are running an algorithm and it takes a minute to run. For each of the orders of execution below, how long will the program take if you double the size of the input? Show how you got your answer.

   a) n

   b) n^2

   c) $1000n^2$

   d) n^3

   e) $\sqrt{n}$

2. (20 pts) In the previous problem the size of the input was doubled. That made the time computation easy for orders of execution that are constant powers without consideration of what n actually is. For the various order of execution measures in this problem, it is not so easy. Assume that it takes 1 minute to do a problem of size $n = 10$. How long does it take to do a problem of size $n = 20$ and $n = 40$? For problem (a) give your answer in years assuming 365 days in a year.

   a) 2^n

   b) $\log_2(n)$

   c) $\ln(n)$

   d) $n \log_2(n)$

3. (30 pts) Recently, I was looking for all possible distinct cases of a combinatorial configuration. So I thought I would enumerate them. This was modeled as a search space of $(2^n)!$ cases (that reads as two to the n **factorial**). Assume I wrote my algorithm to do 1,000,000 cases a second.

a) How big an n can I test considering time as a resource? (Hint: pick a time after which you think the algorithm has consumed too much time and hence become intractable. Explain using your choice.)

b) Suppose I wanted to store 10 bytes worth of information about each case for handy reference. How much memory is consumed in each case that is tractable in terms of time (Hint: give me a number for each tractable case and explain)?

c) I removed all symmetries in the problem and saved a **vast amount of time and space** by making the algorithm $O\left(\frac{(2^n-1)!}{n!}\right)$. Now, with this savings, what size problems do you think are tractable in the space resource and why?

4. (20 pts) If your program is $O(2^n)$ and can process 1 element in a 1 minute, 2 elements 2 minutes, 3 elements in 4 minutes, 4 elements in 8 minutes, etc. How many elements can you completely process in:

   a) a day?

   b) a month of 30 days?

   c) a year of 365 days?

   d) before you retire in 50 years?

   e) before Justin Bieber's death of old age 100 years from now?

5. (30 pts) Show from the definition of O and Ω that:

   a) $n(n+1)/2 \in O(n^3)$

   b) $n(n+1)/2 \in O(n^2)$

   c) $n(n+1)/2 \in \Omega(n)$

6. (35 pts) Compute the sums:

   a) $\sum_{i=1}^{n} 1$

   b) $\sum_{i=1}^{n} i$

   c) $\sum_{i=1}^{n} 3i(i+1)$

   d) $\sum_{i=1}^{n} \sum_{j=i+1}^{n} 1$

   e) $\sum_{i=1}^{n} \sum_{j=1}^{i} 1$

   f) $\sum_{i=1}^{n} \sum_{j=1}^{n} 1$

   g) $\sum_{i=1}^{n} \sum_{j=1}^{n} ij$

7. (30 pts) Given this algorithm:

```
def alpha(x, n) :
    prod = 1
    while n>0 :
       prod *= x
       n -= 1
    return prod
```

   a) What does this algorithm do?
   b) What is the measure of input size?
   c) What is the key operation for measure of resource consumption (time)?
   d) What is the order of execution for this algorithm? Explain.

8. (30 pts) Given this algorithm:

```
def beta(x, n) :
    prod = 1
    while n>0 :
        if n&1 == 1 :
            prod *= x
        n >>= 1
        x *= x
    return prod
```

   a) What does this algorithm do?
   b) What is the measure of input size?
   c) What is the key operation for measure of resource consumption (time)?
   d) What is the order of execution for this algorithm? Explain.