University of Nevada, Las Vegas Computer Science 477/677 Spring 2020

Practice Examination for April 30, 2020

Updated `Sat Apr 25 12:12:28 PDT 2020`

The entire practice examination is 340 points.

The current closure order extends to April 30.

1. True or False. [5 points each] T = true, F = false, and O = open, meaning that the answer is not known to science at this time.

   (a) ________ Computers are so fast today that complexity theory is only of theoretical, but not practical, interest.

   (b) ________ If a problem can be worked in $O(n)$ time by a single processor, then it can be worked in polylogarithmic time, that is, $O(\log^k n)$ time for some constant k, if polynomially many processors are used.

   (c) ________ The asymptotic space complexity of a program cannot excced its asymptotic time complexity.

2. (10 points each) Find the asymptotic final value of `kount` for each of these code fragments in terms of n. In each case, the answer is $\Theta(n)$, $\Theta(n^2)$, $\Theta(n \log n)$, $\Theta(\log^* n)$, $\Theta(\log\log n)$, $\Theta(\sqrt{n})$,

   (a)
   ```
   int kount = 0;
   for(int i = 0; i < n; i++)
    for(int j = i; j > 0; j--)
     kount++;
   ```

   (b)
   ```
   int kount = 0;
   for(int i = 1; i < n*n; i = 2*i)
    kount++;
   ```

   (c)
   ```
   int kount = 0;
   for(int i = 0; i*i < n; i++)
    kount++;
   ```

   (d)
   ```
   int kount = 0;
   for(int i = 1; i < n; i = 2*i)
    for(int j = 0; j < i; j++)
     kount++;
   ```

(e)
```
int kount = 0;
for(int i = 1; i < n; i = 2*i)
 for(int j = i; j < n; j++)
  kount++;
```

(f)
```
int kount = 0;
for(int i = n; i > 0; i = log(i))
 kount++;
```

(g)
```
int kount = 0;
for(int i = 0; i < n*n; i = i+2*sqrt(i)+1)
 kount++;
```

Hint: Use the substitution $\mathtt{i} = \mathtt{j}^2$

(h) Deleted.

(i) Deleted.

(j)
```
int kount = 0;
for(int i = 2; i < n; i = i*i)
 kount++;
```

(k)
```
int kount = 0;
for(int i = 1; i < n; i++)
 for(int j = i; j < n; j=2*j)
  kount++;
```

3. (10 points each) Find the asymptotic complexity of $F(n)$ for each recurrence, expressed using Θ if possible, Ω or O otherwise.

For these problems use the master theorem.

(a) $F(n) \leq F(n/2) + n$

(b) $F(n) = 2F(n/2) + n$

(c) $F(n) = 4F(n/2) + n$

(d) $F(n) \geq F(n/2) + 1$

(e) $F(n) = 2F(n/4) + \sqrt{n}$

For these problems, use the anti-derivative method.

(f) $F(n) = F(n-1) + n$

(g) $F(n) = F(n-2) + n^2$

(h) $F(n) = F(n - \sqrt{n}) + n$

For these problems, use the generalized master theorem.

(i) ________________

$F(n) = F(n/3) + F(n/4) + F(n/5) + n$

(j) ________________

$F(n) = 2F(n/4) + F(n/2) + n$

(k) ________________

$F(n) = F(3n/5) + F(4n/5) + n$

(l) ________________

$F(n) = F(3n/5) + F(4n/5) + n^2$

(m) ________________

$F(n) = 2F(2n/3) + F(n/3) + 1$

(n) ________________

$F(n) \leq F(n/5) + F(7n/10) + n$

4. [20 points] Find an optimal prefix-free code for the alphabet $\{A, B, C, D, E, F, G\}$ with the following frequency distribution.

A	12
B	6
C	8
D	10
E	30
F	4
G	5

5. [20 points] Consider a array of n numbers. The sum of those numbers can be computed in logarithmic time by using n processors working in parallel.

   Suppose that the numbers in the array are:

   1, 2, 9, 0, 5, 7, 2, 8, 6, 3, 4, 1, 5, 9, 5, 6.

   Walk through the parallel algorithm which finds the sum using n processors. At each level, show the intermediate results. Your diagram should clearly indicate each time two numbers are combined into one number.

6. [20 points] Let A be an array of n numbers. Consider the problem of finding the maximum sum of any contiguous subarray. For example, if the items of A are -3, 2, 4, -5, 3, 2, -1, 4, the contiguous array with the maximum sum is 2, 4 -5, 3, 2, -1, 4; If the items of A are -5, 3, -2, 4, 6, -8, 1, -3, 5 then the answer is 3, -2, 4, 6. There are at four three known algorithms for this problem:

   (a) An exhaustive algorithm which takes $O(n^3)$ time.

   (b) A slightly more intelligent algorithm which takes $O(n^2)$ time.

   (c) A rather clever divide and conquer algorithm, which takes $O(n \log n)$ time.

(d) A sophisticated dynamic programming algorithm which takes $O(n)$ time.

Describe an algorithm for this problem, the fastest one you can find.

7. [20 points] The *distance* between two vertices x, y of a connected unweighted graph is defined to be the minimum number of edges of a path from x to y. The *diameter* of such a graph is defined to be the maximum distance between any two vertices.

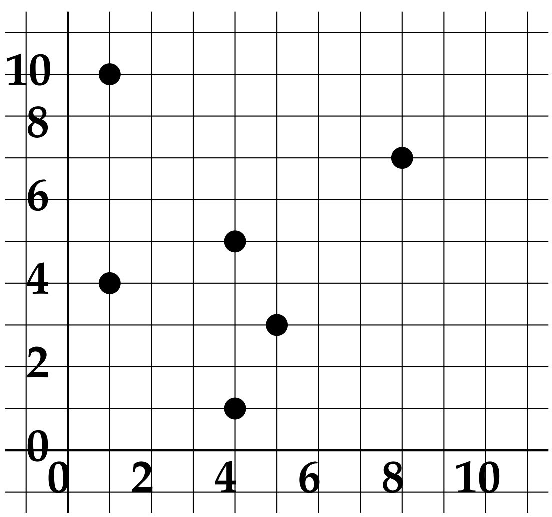
   Suppose you are given a connected undirected graph G with n vertices and m edges, where n is one billion and m is approximately $10n$, and no vertex has degree more than 100. (Think of the internet.) Your job is to find the diameter of G.

   (a) How long would that take if you use the Floyd-Warshall Algorithm?

   (b) Describe the algorithm you would recommend.

   (c) Can your computation be efficiently parallelized if you hava a parallel machine with a billion processors?

8. Use Graham scan to find the convex hull of the set of dots in the figure below. Use the point (1,4) as the pivot.



9. Fill in the blanks. [5 points each blank]

   (a) The items in a priority queue represent ____________________ ____________________

   (b) If a hash table has n places and there are n data items, What is the approximate percentange of places that will hold more than one item? __________ (Within 1 percentage point.)