University of Nevada, Las Vegas Las Vegas Computer Science 477/677 Fall 2019

Answers to Assignment 9 Due Wednesday December 4, 2019

1. Solve each recurrence, expressing the answers using O, Ω, or Θ, whichever is most appropriate.

   (a) $F(n) = 4F(n/2) + n)$
   $F(n) = \Theta(n^2)$

   (b) $F(n) = F(n/2) + \log n$ (Hint: use substitution.)
   $F(n) = \Theta(\log^2 n)$

   (c) $F(n) = F(n-2) + \log n$ (Hint: do not be misled by irrelevancies.)
   $F(n) = \Theta(n \log n)$

   (d) $F(n) = F(n - \sqrt{n}) + n$ (Hint: divide by sides by something.)
   $F(n) = \Theta(n^{3/2})$

   (e) $F(n) = 3(F(n/3) + F(2n/3)) + n^2$
   $F(n) = \Theta(n^2)$

   (f) $F(n) = F(n/2) + F(n/3) + F(n/6) + 1$
   $F(n) = \Theta(n)$

2. Explain how to find the median of n items, deterministically, in $O(\log n)$ time using n processors. Can you do it with asymptotically fewer processors, but still in $O(\log n)$ time?

3. Consider a union/find problem where there are n items, and the total number of `find` operations is n and the total number of `union` operations is also n. Assume that you use path compression.

   (a) Is the time complexity $O(n)$? (Hint: No.)

   (b) What is the time complexity?
   $O(n\alpha(n))$, where α is the inverse Ackermann function.

4. $2n$ items are placed into an open hash table of size n, using a pseudo-random hash function.

   (a) What is the average number of items in a bucket? (Hint: 2.)
   2

   (b) Approximately how many buckets will have no items?
   n/e^2

   (c) Approximately how many buckets will have exactly one item?
   $2n/e^2$

   (d) We say that a two items *collide* if they have the same hash value. Approximately how many other items does a given item x collide with?
   2

   More generally, if each of n items is assigned one of nm labels, the average number of items assigned to each label is m, and the expected proportion of labels which are assigned to exactly k items is $\frac{m^k}{e^m k!}$

5. You are given an acyclic directed graph $G = (V, E)$. Let $n = |V|$ and $m = |E|$.

   (a) Write an algorithm which finds a topological ordering of V.

   On page 90 of our textbook, it is stated that the post number obtained during DFS search is a topological order. This algorithm takes $O(n + m)$ time.

   (b) Write an algorithm which finds the longest path in G.

   Let Pred[v] be the set of predecessors of v, namely all u in V such that (u,v) in E.

```
for all v in V in topological order
 if (Pred[v] is empty) L(v) = 0;
 else
  {
   back[v] = that u in Pred[v] with maximum L[v];
   L(v) = 1 + L[back[v]];
  }
 Pick t in V such that L[t] is maximum.
 Follow back pointers from t to find the maximum length path.
```

   This algorithm takes $O(n + m)$ time.

   (c) Write an algorithm which finds the transitive closure of G.

   (d) Write an algorithm which finds the transitive reduction of G.

   These problems are both a lot harder than you might think. Both can be solved $O(nm)$ time using algorithms explained on the Wikipedia page. We let Succ[v] be the set of all nodes u such that (v,u) is in E. The following algorithm computes the transitive closure of G = (V,E).

```
for all v in V in reverse topological order
Use DFS to visit all nodes reachable from v.
For each node u reachable from v, insert the edge (v,u) into E.
```

   There are $O(n)$ iterations of the outer loop. For each of those, the DFS search takes $O(m)$ time. Thus the time is $O(nm)$.

   The following code computes the transitive reduction of G.

```
for all v in V // in any order, however, reverse topological order seems best
 {
  Let Succ[v) = {u[1], u[2] , ...}
  for all u[i] in U
   {
    Use DFS to visit all nodes reachable from u[i].
    if (u[j] is reachable from u[i] for some j != i)
     delete (v,u[j]) from E
   }
 }
```

   There are $O(n)$ iterations of the outer loop. For each of those, the DFS search takes $O(m)$ time. Thus the time is $O(nm)$.

6. You can only type 80 characters on a line. You are given a sequence of words, $w_1, w_2, \ldots w_n$ of various lengths, which do not fit into one line. You want to construct a paragraph, where each line is as long as possible without exceeding 80 characters. The last line can have any length. No word has length greater than 80, and there must be a space between any two consecutive words on a line. Design an algorithm for this problem. (There is a linear, that is, $O(n)$, time algorithm.)

   We compute the Boolean array value break[ ]

```
numonline = length[1];
for(int i = 2; i<=n; i++)
  {
   if(length[i]+numonline) <= 80)
    {newline[i] = false;
     numonline += length[i]+1;
   else
    {newline[i] = true; // a new line begins with word[i]
     numonline = length[i];}
  }
 newline[1] = true;
```