University of Nevada, Las Vegas Computer Science 477/677 Spring 2021

Answers to Practice for Final Examination: Part II

This portion of the practice final is 230 points.

1. Give the asymptotic time complexity, in terms of \( n \), of each of these code fragments. (10 points each)

(a) for(int \( i = 1; i < n; i=2*i \))
    for(int \( j = 1; j < i; j++ \))
    \( \Theta(n) \)

(b) for(int \( i = 1; i < n; i=2*i \))
    for(int \( j = i; j < n; j++ \))
    \( \Theta(n \log n) \)

(c) for(int \( i = 1; i < n; i++ \))
    for(int \( j = i; j > 0; j = j/2 \))
    \( \Theta(n \log n) \)

(d) for(int \( i = 1; i < n; i++ \))
    for(int \( j = n; j > i; j = j/2 \))
    \( \Theta(n) \)

(e) for(int \( i = 1; i < n*n; i++ \))
    \( \Theta(n^2) \)

(f) for(int \( i = 1; i*i < n; i++ \))
    \( \Theta(\sqrt{n}) \)

(g) for(int \( i = 1; i < n; i++ \))
    for(int \( j = 1; j < n; j = j+i \))
    \( \Theta(n) \)

(h) This problem requires two answers. Its time complexity is not \( \Theta \) of any of the usual functions we deal with. Instead, it’s \( \Omega \) of some function of \( n \) and \( O \) of some other function of \( n \). Give both.
    for(int \( i = 2; i < n; i=i*i \))
    for(int \( j = 1; j < i; j++ \))
    \( O(n) \) and \( \Omega(\sqrt{n}) \)

2. Give asymptotic solutions to the following recurrences.

(a) \( F(n) = F(n/2) + F(n/3) + n \) \( F(n) = \Theta(n) \)

(b) \( G(n) = G(n/4) + 2G(n/16) + \sqrt{n} \) \( G(n) = \Theta(\sqrt{n \log n}) \)

(c) \( H(n) = H(n - \log n) + \log n \) \( H(n) = \Theta(n) \)
3. [10 points] Draw an acyclic directed graph of 6 vertices and 15 arcs.

4. [10 points] Draw a directed graph with exactly two strong components, each of which has 4 vertices. The graph must have a “source” vertex s from which every vertex is reachable.

5. [10 points] Draw a planar graph with 5 vertices and 10 edges.

Impossible. For a planar graph, if \( n > 2 \), \( m \leq 3n = 6 \)
6. (a) [20 points] Write pseudocode for the Floyd-Warshall algorithm. Let the vertices of a weighted directed graph be 1, 2, \ldots, n. Let \( W(i,j) \) be the weight of the arc \((i,j)\). If there is no arc, we let \( W(i,j) = \infty \). Let \( V[i,j] \) be the minimum weight of any path from \( i \) to \( j \) found so far, and \( \text{back}[j] \) the backpointer of that path.

**The Floyd Warshall Algorithm**

```java
for(int i = 1, i <= n, i++)
    V[i,i] = 0;
for(int i = 1, i <= n, i++)
    for(int j = 1; j <= n; j++)
    {
        V[i,j] = W(i,j); // could be infinity
        back[j] = i;
    }
for(int j = 1, j <= n, j++)
    for(int i = 1, i <= n, i++)
        for(int k = 1, k <= n, k++)
        {
            temp = V[i,j] + V[j,k];
            if(temp < V[i,k])
            {
                V[i,k] = temp;
                back[k] = j;
            }
        }
```
(b) [20 points] Write pseudocode for the Bellman-Ford algorithm. Be sure to incorporate the shortcut.

Let $0, 1, \ldots, n-1$ and $e_0, e_1, \ldots, e_{m-1}$.

**The Bellman Ford Algorithm**

```java
for(int i = 1; i <= n; i++)
    V[i] = \infty;
V[0] = 0;
bool done = false;
while(not done)
    { done = true;
      for(int k = 0; k < m; k++)
         { temp = V[x_k] + W_k;
           if(temp < V[y_k])
              { V[y_k] = temp;
                back[y_k] = x_k;
                done = false;
              }
         }
    }
```

7. [20 points] If you need to solve the all-pairs problem for a weighted graph with $n$ nodes and $m$ edges, which algorithm would you use?

The Floyd-Warshall algorithm is easier to use, since its code is quite simple. However, if $m \log n$ is substantially less than $n^2$, the time savings may make Johnson’s algorithm a better choice. Both algorithms have $\Theta(n^2)$ space complexity.

8. [20 points] XXX Write the Polish and reverse Polish expressions equivalent to $a \ast (- (b - c) \ast d)$.

**Polish:** $\ast a \ast \sim b \ast c d$

**Reverse Polish:** $abc \sim d \ast \ast$

9. [20 points] Prove that there is no comparison-based algorithm for sorting six items that never uses more than nine comparisons.

$2^9 = 512 < 720 = 6!$
10. [20 points]

I made a mistake writing this code in Part I of the practice final. Here is the correct version.

```c
int product(int a, int b)
{
    assert(b >= 0);
    int c = a;
    int d = b;
    int total = 0;
    while(d > 0)
    {
        if(d%2) total = total + c;
        c = 2*c;
        d = d/2;
    }
    return total;
}
```

(a) What does this function do?

It returns the product, \(a \times b\).

(b) What is the loop invariant of the while loop?

\(a \times b = c \times d + \text{total}\).