University of Nevada, Las Vegas Computer Science 477/677 Fall 2026

Answerst to Second Examination October 29, 2026

1. Fill in the blanks.

   (i) [5 points] **Dijkstra's** algorithm does not permit any arc to have negative weight.

   (ii) [5 points] The expected height of a treap with n nodes is $\Theta(\log n)$.

   (iii) [5 points] The maximum possible number of edges of a planar graph with 5 nodes is **9**.

   (iv) [5 points] The worst case time complexity of Johnson's algorithm on a weighted directed graph with n vertices and m arcs is $\Theta(nm \log n)$.

2. For each of the code fragments below, find the asymptotic time complexity in terms of n.

   (i) [5 points]

   ```
   for(float x = n; x > 1; x = x/2)
   ```

   $\Theta(\log n)$

   (ii) [5 points]

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

   $\Theta(\sqrt{n})$

   (iii) [5 points]

   ```
   for(float x = n; x > 2.0; x = sqrt(x))
   ```

   $\Theta(\log \log n)$

   (iv) [5 points]

   ```
    for(int i = 1; i < n; i=2*i)
   ```

   $\Theta(\log n)$

   (v) [5 points]

   ```
   for(int i = n; i > 1; i--)
    for(int j = i; j > 1; j = j/2)
   ```

   $\Theta(n \log n)$

   (vi) [5 points]

   ```
   for(int i = n; i > 1; i--)
    for(int j = n; j > i; j = j/2)
   ```

   $\Theta(n)$

(vii) [5 points]

```
for(int i = 2; i < n; i=i*i)
```

$\Theta(\log \log n)$

3. [30 points] What follows is part of a C++ implementation of a linked list implementation of stack of integer. Fill in the missing code.

```
struct staknode;
typedef staknode*stak;
struct staknode
 {
  int itm;
  stak lnk;
 };

void push(stak&top, int n)
 {
  stak tmp = new staknode;
  tmp->itm = n;
  tmp->lnk = top;
  top = tmp;
 }

bool empty(stak top);
 {
  return top == nullptr;
 }

int pop(stak&top)
 {
  assert(not empty(top));
  int tmp = top->itm;
  top = top->lnk;
  return tmp;
 }
```

4. Give asymptotic solutions to the following recurrences.

(i) [5 points] $F(n) = 2F(n/2) + n$
$F(n) = \Theta(n \log n)$

(ii) [5 points] $F(n) = F(n/2) + F(n/3) + n$;
$F(n) = \Theta(n)$

(iii) [5 points] $F(n) = 4F(2n/5) + F(3n/5) + n^2$
$F(n) = \Theta(n^2 \log n)$

(iv) [5 points] $F(n) = F(n - \sqrt{n}) + 1$
$F(n) = \Theta(\sqrt{n})$

(v) [5 points] $F(n) = 3F(n/9) + 1$
$F(n) = \Theta(\sqrt{n})$

(vi) [5 points] $F(n) = 9F(n/3) + n^2$
$F(n) = \Theta(n^2 \log n)$

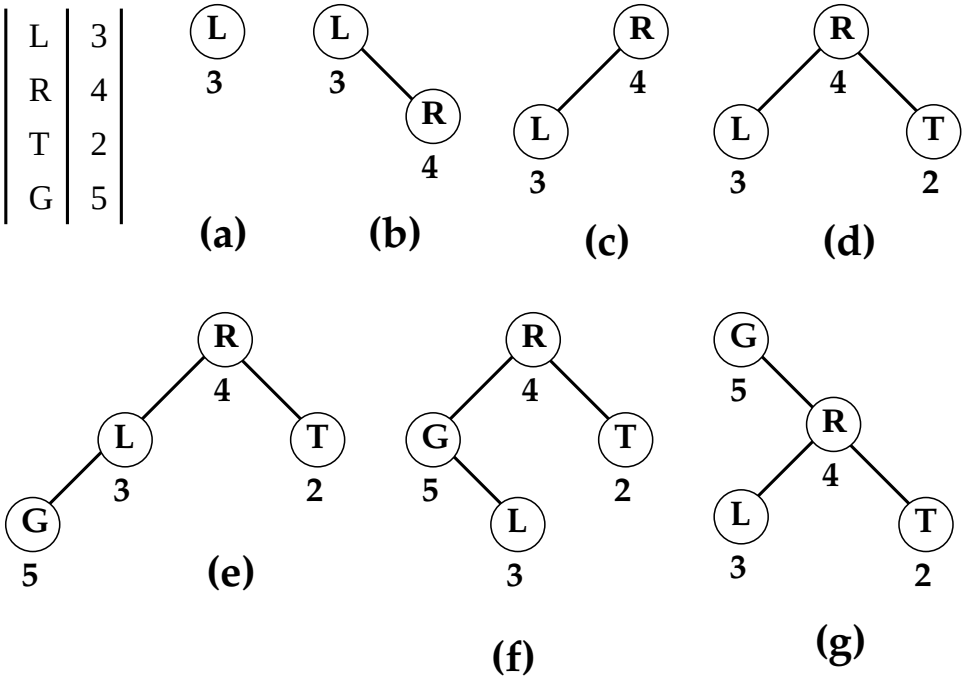
(vii) [10 points] $F(n) = F(\sqrt{n}) + \log n$
$F(n) = \Theta(\log n)$

5. [20 points] Write pseudocode for the Bellman Ford algorithm. Shortcut and back pointers.

6. [20 points] Write pseudocode for the Floyd Warshall algorithm. Backpointers.

7. [20 points] Insert the items L, R, T, G into a treap, in that order, using the priorities given in the table below, and using max-heap order. Show each a figure after each insertion or rotation.

L	3
R	4
T	2
G	5



8. [20 points] The following C++ code computes values of a function.

```
int G(int n)
 {
  if(n <= 2) return n;
  else return G(n/2)+G(n/3)+G(n/6)+n*n;
 }
```

(i) [10 points] What is the asymptotic value of $G(n)$? $G(n) = G(n/2) + G(n/3) + G(n/6) + n^2$
By Akra Bazzi, $G(n) = \Theta(n^2)$

(ii) [10 points] What is the asymptotic time complexity of the computation of $G(n)$ using the recursive code above? Assume that each arithmetic operation takes constant time.

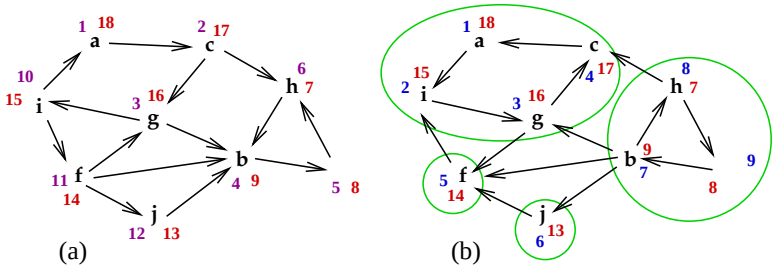
$T(n) = T(n/2) + T(n/3) + T(n/6) + 1$. By Akra-Bazzi, $T(n) = \Theta(n)$

(iii) [10 points] Write C++ code for a computation of $G(n)$ using dynamic programming, but not memoization. Assume that you are given the value of $n \geq 0$, as well as an array `int G(N)`, where N $> n$.

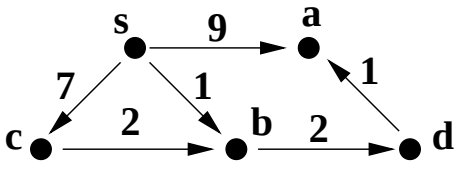
For i = 1 to n
If(n ¡= 2) G(n) = n
Else G(n) = G(n/2)+G(n/3) + G(n/6) + n*n

(iv) [10 points] What is the asymptotic time complexity of the computation of $G(n)$ using memoization? Assume that each arithmetic operation and each fetch or store takes constant time. The memos are for $G(i)$ for $i = n/(2^i * 3^i)$, provided the denominator is $O(n)$. There are $\Theta(\log^2 n)$ such values.

9. [20 points] Walk through the computation of the strong components of the graph shown in (a). Use the other figures as needed.



10. [20 points] Walk through Dijkstra's algorithm for the weighted directed graph shown below.



	s	a	b	c	d
V	0	~~9~~ 4	1	7	3
back		~~s~~ d	s	s	b

11. [20 points] Write C++ code which implements a function which computes x^n, where x and n are both positive. You can use any algorithm for this code, not necessarily one that I've shown you, but the time complexity of your code should be $O(\log n)$. I've helped you by writing the first three lines.

```
float power(float x, int n) // computes x to the power of n
 {
  assert(x > 0 and n > 0);
  if (n == 1) return x;
  else if (n%2) return x*power(x,n-1)
  else
   {
    float tmp = power(x,n/2);
    return tmp*tmp;
```

```
  }
}
```