University of Nevada, Las Vegas Computer Science 477/677 Fall 2025

Study for Examination March 12, 2025

1. Fill in the blanks.

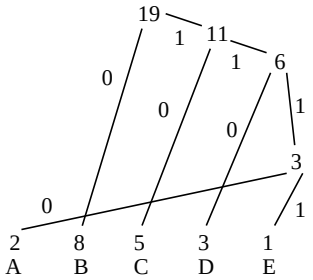
   (a) A binary tree of height 4 cannot have more than **16** leaves. (Exact answer.)

   (b) **quicksort** and **mergesort** are divide-and-conquer sorting algorithms.

   (c) For **Dijkstra's** algorithm, no arc may have **negative** weight.

2. Construct an optimal prefix-free binary code for the following weighted alphabet.

A	2	1110
B	8	0
C	5	10
D	3	110
E	1	1111



3. Sort the following array using heapsort. Add extra rows if needed.

A	M	G	X	E	W
A	M	W	X	E	G
A	X	W	M	E	G
X	A	W	M	E	G
X	M	W	A	E	G
G	M	W	A	E	**X**
W	M	G	A	E	**X**
E	M	G	A	**W**	**X**
M	E	G	A	**W**	**X**
A	E	G	**M**	**W**	**X**
G	E	A	**M**	**W**	**X**
A	E	**G**	**M**	**W**	**X**
E	A	**G**	**M**	**W**	**X**
A	**E**	**G**	**M**	**W**	**X**
A	**E**	**G**	**M**	**W**	**X**

4. What is the loop invariant of the following code computes x^{2n}. What is the loop invariant?

```
float power2(float x, int n) // input condition: n > 0
 {
  assert(n > 0);
  float z = 1.0;
  float y = x*x;
  int m = n;
  while(m > 0)
   {
    if(m%2) z = y*z;
    y = y*y;
    m = m/2;
   }
  return z;
 }
```

$x^{2n} = y^m * z$

5. Give the asymptotic complexity of each code fragment in terms of n.

   (a) for(int $i = 2$; $i < n$; $i = i^2$)

   $\Theta(\log \log n)$

   (b) for(int $i = 0$; $i < n$; $i++$)
   for(int $j = i$; $j > 1$; $j = \sqrt{j}$)

   $\Theta(n \log \log n)$

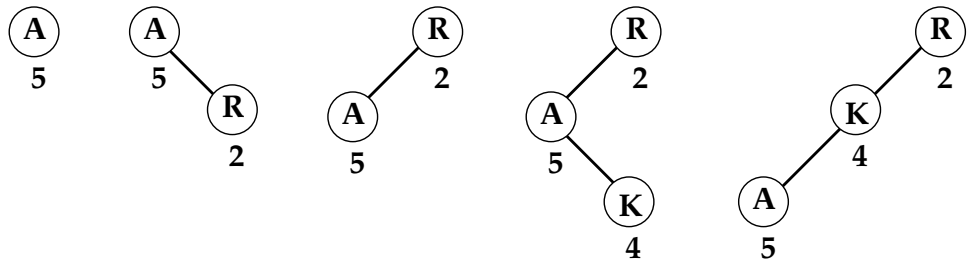
   (c) for(int $i = n$; $i > 1$; $i--$)
   for(int $j = n$; $j > i$; $j = j/2$)

   $\Theta(n)$

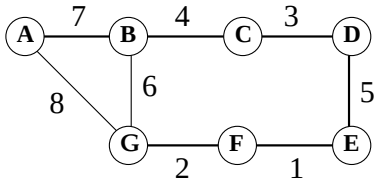
   (d) for(int $i = n$; $i > 1$; $i--$)
   for(int $j = i$; $j > 1$; $j = j/2$)

   $\Theta(n \log n)$

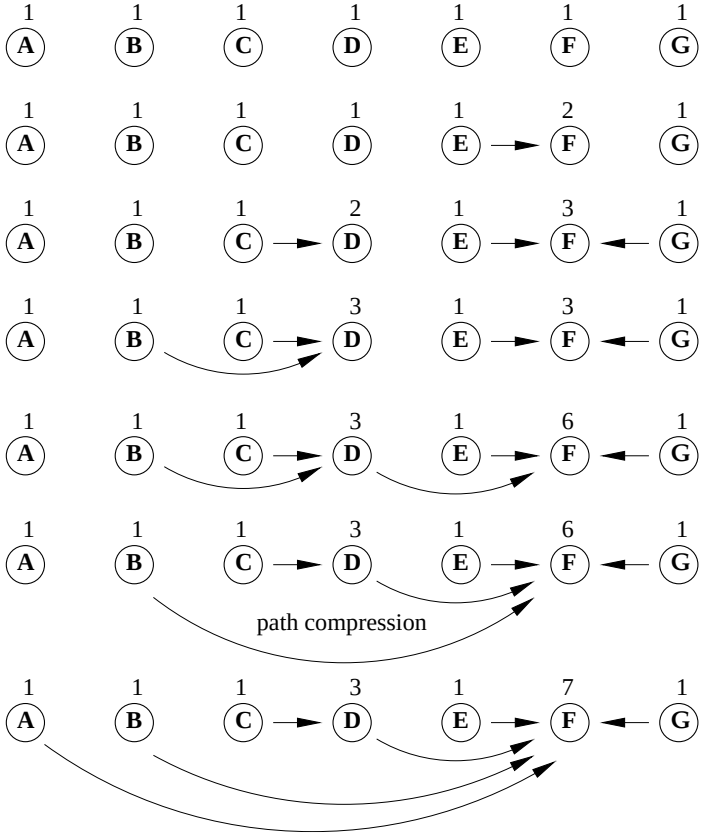
6. Starting with an empty treap, insert the items A, R, and K, in that order, where priorities are given in the table below. Use minheap order. Show the steps.



7. Use Kruskal's algorithm to construct a minimnal spanning tree of the weighted graph shown below. Use union/find, and show the evolution of the disjoint set structure. Don't forget path compression.



EF
FG
CD
BC
DE
BG
AB
AG



8. Define *Good numbers* $\{G_n\}$ recursively as follows:
   $G(1) = 1$
   $G(2) = 3$
   $G(n) = (G(n-2) + G(n-1))\%13$ for all $n \geq 3$

   Write a C++ function which computes and prints $G(n)$ for any given positive integer n, using dynamic programming, not recursion. Here's a start:

```
const int N = 20; // or whatever we need
int G[N];

void computeG(int n)
 {
  // input condition: n >= 1
  G[1] = 1;
  G[2] = 3;
  for(int i = 3; i < N)
   G[i] = (G[i-2]+G[i-1])%13;
  cout << "G(" << n << ") = " << G[n] << endl;
 }
```

9. Write pseudocode for the Bellman-Ford algorithm.

Let $e_j = (s_j, t_j)$ of weight w_j be the j^{th} arc.

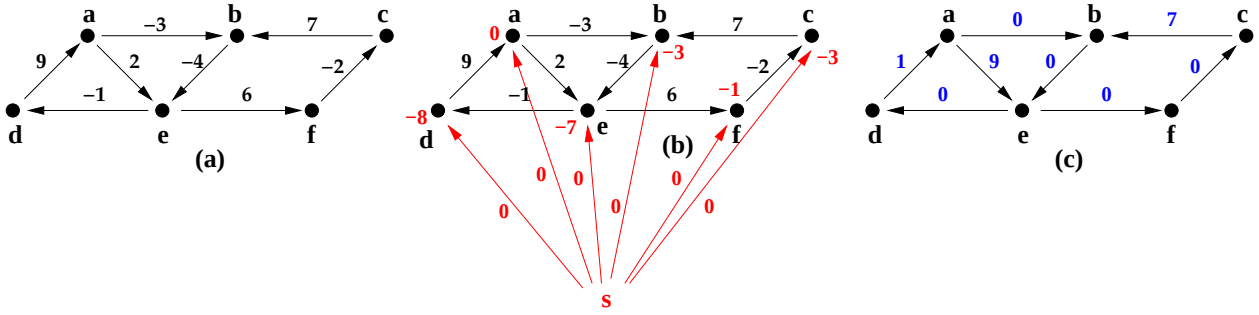
For all i from 1 to n
  V(i) $= \infty$
V(0) $= 0$
finished = false
For all j from 1 to m, provided not finished
  finished = true
  temp $=$ V$(s_j) + w_j$
  If temp $<$ V(t_j)
    V(t_j) = temp
    back$(t_j) = s_j$
    finished = false

10. Write pseudocode for the Floyd-Warshall algorithm.

Let W(i, j) be the weight of the arc from i to j. If there is no such arc, let W$(i, j) = \infty$.

For all i and j
  V(i, j) = W(i, j)
  back$(i, j) = i$
For all j
  For all i
    For all k
      temp $=$ V$(i, j)+$ V(j, k)
      If temp $<$ V(i, k)
        V(i, k) = temp
        back(i, k) =back(j, k)

11. In the figure below, (a) shows a weighted directed graph. In (c), replace each edge weight using the techniques of Johnson's algorithm. Use (a) and (b) for your work. Do not complete Johnson's algorithm.



12. What is the asymptotic complexity, in terms of n, of the following recursive C++ functions?

(a)
```
int george(int n)
{
 // input condition: n >= 0
 if(n < 1) return 1;
 else return george(n/2)+george(n/2)+n;
}
```
$\Theta(n \log n)$

(b)
```
int martha(int n)
{
 // input condition: n >= 0
 if(n < 1) return 1;
 else return martha(n-1)+marth(n-1);
}
```
$\Theta(2^n)$

(c)
```
int alexander(int n)
{
 // input condition: n >= 1
 if(n == 1) return 0;
 else return 1 + alexander(n/2);
}
```
$\Theta(\log n)$