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

Examination March 12, 2025

Name:_____

No books, notes, scratch paper, or calculators. Use pen or pencil, any color. Use the rest of this page and the backs of the pages for scratch paper. If you need more scratch paper, it will be provided. If you want anything on extra pages to be graded, staple those pages to your test and write, "Please grade this page."

The entire test is 235 points.

1. Fill in the blanks.

   (a) [5 points] A binary tree with 8 nodes cannot have height less than **4**. (Exact answer.)

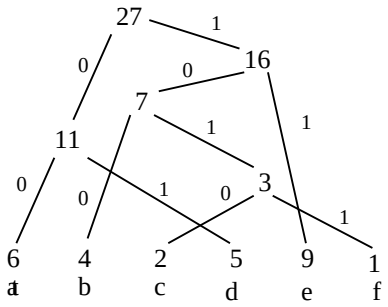
   (b) [10 points] **quicksort** and **mergesort** are divide-and-conquer sorting algorithms.

   (c) [5 points] **binary search** is a divide-and-conquer searching algorithms.

   (d) [5 points] The asymptotic expected height of a treap of n nodes is $\Theta(logn)$.

2. [10 points] Construct an optimal prefix-free binary code for the following weighted alphabet.

a	6	00
b	4	100
c	2	1010
d	5	01
e	9	11
f	1	1011



3. [10 points] What does the following code compute? What is the loop invariant?

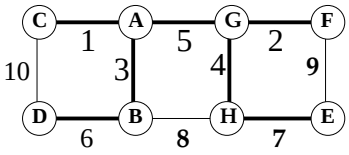
```
float mystery(float x, int n) // input condition: n > 0
 {
  assert(n > 0);
  float z = 1.0;
  float y = x*x;
  int m = 2*n;
  while(m > 0)
   {
    if(m%2) z = y*z;
    y = y*y;
    m = m/2;
   }
  return z;
 }
```

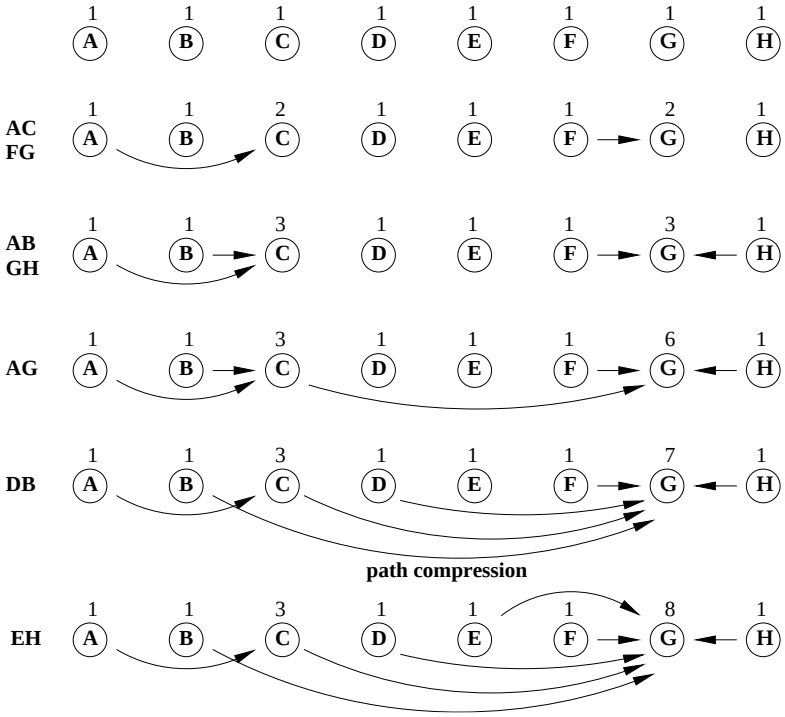
It computes x^{4n}. The loop invariant is $x^{4n} = y^m * z$

4. [20 points] Sort the following array using heapsort. Add extra rows if needed.

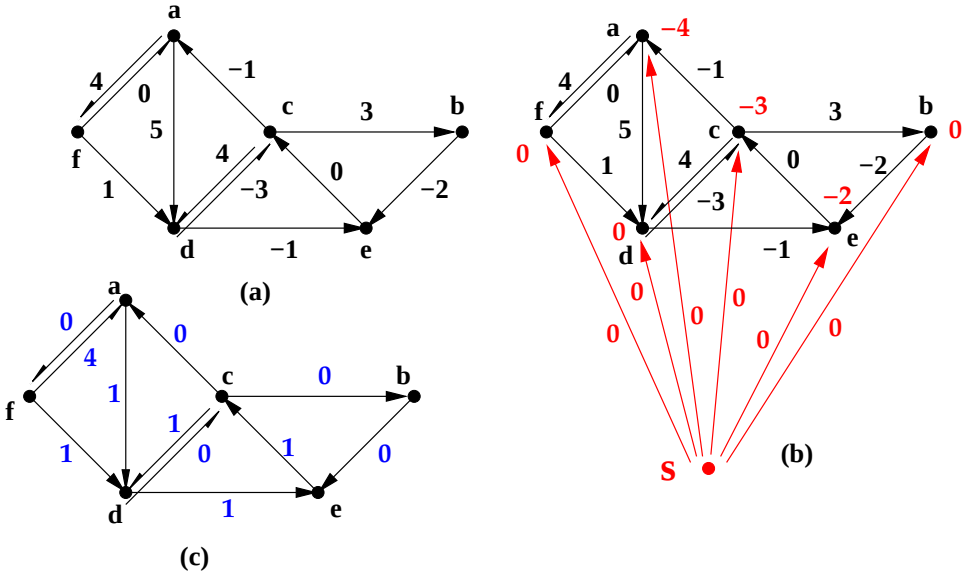
1	2	3	4	5	6	7	8
A	N	H	Z	D	V	L	Q
A	N	V	Z	D	H	L	Q
A	Z	V	N	D	H	L	Q
A	Z	V	Q	D	H	L	N
Z	A	V	Q	D	H	L	N
Z	Q	V	A	D	H	L	N
Z	Q	V	N	D	H	L	A
A	Q	V	N	D	H	L	**Z**
V	Q	A	N	D	H	L	**Z**
V	Q	L	N	D	H	A	**Z**
A	Q	L	N	D	H	**V**	**Z**
Q	A	L	N	D	H	**V**	**Z**
Q	N	L	A	D	H	**V**	**Z**
H	N	L	A	D	**Q**	**V**	**Z**
N	H	L	A	D	**Q**	**V**	**Z**
D	H	L	A	**N**	**Q**	**V**	**Z**
L	H	D	A	**N**	**Q**	**V**	**Z**
A	H	D	**L**	**N**	**Q**	**V**	**Z**
H	A	D	**L**	**N**	**Q**	**V**	**Z**
D	A	**H**	**L**	**N**	**Q**	**V**	**Z**
A	**D**	**H**	**L**	**N**	**Q**	**V**	**Z**
A	**D**	**H**	**L**	**N**	**Q**	**V**	**Z**

5. [20 points] Walk through Kruskal's algorithm to find the minimum spanning tree of the weighted graph shown below. Show the evolution of the union/find structure. Whenever there is choice between two edges of equal weight, choose the edge which has the alphabetically largest vertex. Whenever there is a union of two trees of equal weight, choose the alphabetically larger root to be the root of the combined tree. Indicate path compression when it occurs.





6. [20 points] 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.



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

(a) [10 points] for(int $i = n$; $i > 2$; $i = \sqrt{i}$)

$\Theta(\log \log n)$

(b) [10 points] for(int $i = 0$; $i < n$; $i++$)
for(int $j = i$; $j < n$; $j = j^2$)

The true answer is that the code will not halt. That was unintentional. If I had initialized $i = 2$ instead of $i = 0$, the answer would have been $\Theta(n)$.

(c) [10 points] for(int $i = 1$; $i < n$; $i++$)
for(int $j = i$; $j < n$; $j = 2 * j$)

$\Theta(n)$

(d) [10 points] for(int $i = n$; $i > 1$; $i--$)
for(int $j = 1$; $j < i$; $j = 2 * j$)

$\Theta(n \log n)$

8. [20 points] Write pseudocode for the Bellman-Ford algorithm. Be sure to include the shortcut.

Let $e_j = (s_j, t_j)$ of weight w_j be the j^{th} arc, for $0 \leq j < m$.
To simplify the code, we use only the primary shortcut, with the Boolean variable finished.

V(0) = 0
For all i from 0 to $n - 1$
  V(i) = ∞
finished = false
For all ℓ from 1 to n, provided not finished
  finished = true
  For all j from 0 to $m - 1$
    temp = V(s_j) + w_j
    If temp < V(t_j)
      V(t_j) = temp
      back(t_j) = s_j
      finished = false
If not finished
  "There is a negative cycle."

If this were a C++ program, the 5^{th} line would be:
for(int $\ell = 1$; $\ell <= n$ and not finished; ℓ++)

9. [20 points] 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
  $V(i, j) = W(i, j)$
  $\text{back}(i, j) = i$
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
        $\text{back}(i, k)$ =$\text{back}(j, k)$

10. [10 points] What is the asymptotic complexity of the function **george**(n), in terms of n?

```
int george(int n)
 {
  // input condition: n >= 0
  if(n < 1) return 1;
  else return george(n-1)+george(n-1);
 }
```

11. [10 points] What is the asymptotic complexity of the function **martha**(n) in terms of n?

```
int martha(int n)
 {
  // input condition: n >= 1
  if(n == 1) return 0;
  else return n + martha(n/2);
 }
```

martha(n) = $\Theta(n)$

12. [10 points]

What is the asymptotic time complexity of above code which computes **martha**(n), in terms of n?

Hint: This is **not** the same question as the previous one!

The time to compute martha(n) is $\Theta(\log n)$.

13. [20 points] Write a C++ function which solves the simplest coinrow problem we have discussed, namely, given a sequence of positive numbers, find the subsequence of maximum total, subject to the condition that the subsequence may not contain any two consecutive terms of the sequence.

//

```
#include<cstdio>
#include<iomanip>
#include<cassert>
#include<string>
#include<cmath>
#include<iostream>
#include<sstream>
#include<stdio.h>
#include<stdlib.h>
 using namespace std;

const int N = 20; // number of coins
const int sentinel = -1;
int coin[N];
int A[N]; // A[i] = maximum sum of legal subsequence ending at coin[i];
int B[N];  // B[i] = backpointer

void getcoins()
 {
  for(int i = 0; i < N; i++)
   cin >> coin[i];
 }

void computeA()
 {
  A[0] = coin[0];
  B[0] = sentinel;
  A[1] = coin[1];
  B[1] = sentinel;
  A[2] = coin[0]+coin[2];
  B[2] = 0;
  for(int i = 3; i < N; i++)
   {
    int a1 = coin[i] + A[i-3];
    int a2 = coin[i] + A[i-2];
    //cout << "i = " << i << " a1 = " << a1 << " a2 = " << a2 << endl;
    if(a1 > a2)
     {
      A[i] = a1;
```

```
      B[i] = i-3;
     }
    else
     {
      A[i] = a2;
      B[i] = i-2;
     }
   }
 }

void writecoins()
 {
  for(int i = 0; i < N-1; i++)
   cout << coin[i] << ",";
  cout << coin[N-1] << "." << endl;
 }

void writebest(int i)
 {
  if(i > sentinel)
   {
    writebest(B[i]);
    cout << coin[i] << " + ";
   }
 }

void writebest()
 {
  if (A[N-2] > A[N-1])
   {
    writebest(B[N-2]);
    cout << coin[N-2] << " = " << A[N-2] << endl;
   }
  else
   {
    writebest(B[N-1]);
    cout << coin[N-1] << " = " << A[N-1] << endl;
   }
 }

int main()
 {
  getcoins();
  writecoins();
```

```
    computeA();
    writebest();
    return 1;
   }
//
```