University of Nevada, Las Vegas Computer Science 477/677 Fall 2024

Answers to Examination September 25, 2024

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 examination is 265 points.

1. [5 points] For any sorting algorithm where all branch points in the code are comparisons, the worst case number of comparisons while sorting n items is $\Omega(n \log n)$. Alternative answer: $\Omega(\log(n!))$.

2. [5 points] The items of a priority queue represent **unfulfilled obligations**

3. [15 points] Name three kinds of priority queues: **stack queue heap**

4. The worst case asymptotic time coplexity of sorting an array of length n is

   (a) [5 points] $O(n^2)$ using bubblesort.

   (b) [5 points] $O(n^2)$ using selection sort.

   (c) [5 points] $O(n^2)$ using insertion sort.

   (d) [5 points] $O(n \log n)$ using mergesort.

   (e) [5 points] $O(n \log n)$ using polyphase mergesort.

   (f) [5 points] $O(n^2)$ using quicksort.

   (g) [5 points] $O(n \log n)$ using heapsort.

   (h) [5 points] $O(n^2)$ using treesort with an ordinary binary search tree.

5. Give an asymptotic value of $F(n)$ for each recurrence.

   (a) [10 points] $F(n) = F(n - \sqrt{n}) + n$

   $F(n) - F(n - \sqrt{n}) = n$

   $$\frac{F(n) - F(n - \sqrt{n})}{\sqrt{n}} = \frac{n}{\sqrt{n}} = \sqrt{n} = n^{\frac{1}{2}}$$

   $F'(n) = n^{\frac{1}{2}}$

   $F(n) = \Theta(n^{\frac{3}{2}})$

   (b) [10 points] $F(n) = 2F(n/2) + n$

   $A = 2, B = 2, C = 1,\ B^C = A$

   $F(n) = \Theta(n \log n)$

(c) [10 points] $F(n) = 2F(n/3) + n^2$

$A = 2, B = 3, C = 2,\ B^C = 9 > A$

$F(n) = \Theta(n^C) = \Theta(n^2)$

(d) [10 points] $F(n) = 4F(n/2) + 1$

$A = 4, B = 2, C = 1,\ B^C = 2 < A$

$F(n) = \Theta(n^{\log_B A}) = \Theta(n^2)$

(e) [10 points] $F(n) = 1 + F(\log n)$

$F(n) = \Theta(\log^*(n))$

6. Write the asymptotic time complexity, in terms of n, of each of these code fragments.

(a) [10 points]

```
for(int i = 0; i < n; i++)
 for(int j = 0; j < i; j++)
  cout << "Hello world" << end;
```

$\int_0^n \int_0^x dydx = \int_0^n y\Big|_0^x dx = \int_0^n \frac{x}{2}dx = \frac{x^2}{4}\Big|_0^n = \frac{n^2}{4} = \Theta(n^2)$

(b) [10 points]

```
for(int i = 0; i < n; i++)
 for(int j = 1; j < i; j = 2*j)
  cout << "Hello world" << end;
```

Two ways to work this. You can start with the shortcut that the time complexity of the inner loop is $\Theta(\log i)$. Since the inner loop does not iterate at all if $i = 0$, we can replace 0 by 1 in the outer loop. Since we are using calculus, we replace log with ln, and we have

$\int_1^n \ln x dx = \Big|x \ln x - x + 1\Big|_1^n = n\ln n - n - 1\ln 1 + 1 = \Theta(n \log n)$

(c) [10 points]

```
for(int i = 0; i < n; i++)
 for(int j = i; j < n; j = 2*j)
  cout << "Hello world" << end;
```

This is a bit trickier than problem 6b As in that problem, we can replace 0 by 1 in the outer loop. We can then substitute $k = \log j$, and we get

```
for(int i = 1; i < n; i++)
 for(int k = log i; k < log n; k++)
  cout << "Hello world" << end;
```

Replace the integers i and k by real variables x and y, and we have

$$\int_1^n \int_{\ln x}^{\ln n} dydx = \int_1^n \big|y\big|_{y=\ln x}^{y=\ln n} dx = \int_1^n (\ln n - \ln x)dx$$

$$= \big|x \ln n - x\ln x + x\big|_1^n = n\ln n - n\ln n + n - 1\ln n + 1\ln 1 - 1 = \Theta(n)$$

(d) [10 points]

```
for(int i = 2; i < n; i = i*i)
 cout << "Hello world" << end;
```

Substitute $j = \log i$ and $m = \log n$ and we obtain

```
for(int j = 1; j < m; j = 2*j)
 cout << "Hello world" << end;
```

Substitute $k = \log j$ and $\ell = \log m$ and we obtain

```
for(int k = 0; k < \ell; k++
 cout << "Hello world" << end;
```

The asymptotic complexity is $\Theta(\ell) = \Theta(\log m) = \Theta(\log\log n)$.

(e) [10 points]

```
for(int i = 0; i < n; i++)
 for(int j = 0; j*j < i; j++)
  cout << "Hello world" << end;
```

$j^2 < i$ is equivalent to $j < \sqrt{i}$. Thus we have $\int_0^n \int_0^{\sqrt{x}} dydx = \int_0^n \sqrt{x}dx = \frac{x^{\frac{3}{2}}}{3/2} = \Theta(x^{\frac{3}{2}})$

(f) [10 points]

```
for(int i = n; i > 2; i = sqrt(i))
 cout << "Hello world" << end;
```

(sqrt returns the square root of its parameter.) Let $j = \log i$ and $\ell = \log j$. Let $m = \log n$ and $p = \log m$. Then we have

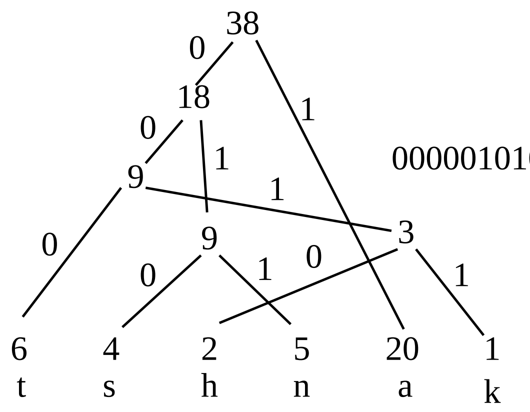
```
for(int j = m; j > 1; j = j/2)
 cout << "Hello world" << end;


for(int l = p; l > 0; l++)
 cout << "Hello world" << end;
```

The answer is $\Theta(p) = \Theta(\log\log n)$.

7. [10 points] Find an optimal prefix code for the indicated weighted alphabet, then encode "thanks" using your prefix code.

t	6	000
s	4	010
h	2	0010
n	5	011
a	20	1
k	1	0011



thanks

000001010110011010

8. [20 points] What follows is a C++ implementation of queue of integer. Finish the subprogram dequeue.

```
int const N = 20;   // N must be at least as large as the number of times
                    // enqueue is executed during the program.

struct queue
 {
  int item[N];
  postn front; // a position in the array, which is an integer
  postn rear; // a position in the array, which is an integer
 };

void init(queue&Q)
 {
  Q.front = 0;
  Q.rear = 0;
 }

bool empty(queue Q)
 {
  return Q.front == Q.rear;
 }

void enqueue(queue&Q, int newitem)
 {
  assert(Q.rear < N);
  Q.item[Q.rear] = newitem;
  Q.rear++;
 }

int dequeue(queue&Q)
 {
  assert(not empty(Q));
  int rslt = Q.item[Q.front];
  Q.front++;
  return rslt;
 }
```

9. [20 points] The following code returns the quotient of its two parameters, namely dividend/divisor, truncated to an integer. Find the loop invariant (LI) of the loop.

```
int quotient(int dividend, int divisor)
 // input condition: dividend >= 0 and divisor is positive
 {
  int m = dividend;
  int rslt = 0;
  //LI holds here
  while(m >= divisor)
   {
    //LI holds here
    m = m - divisor;
    //LI may NOT hold here
    rslt = rslt + 1;
    //LI holds here
   }
  //LI holds here
  return rslt;
 }
```

`LI: dividend = divisor * rslt + m`

10. [20 points] Walk through the steps of heapsort, sorting an array whose items are X,B,L,F,U,N in that order. For ease of grading, use the matrix below. You might not need all the rows. You do not need to write an explanatory phrase at the end of each row.

1	2	3	4	5	6
X	B	L	F	U	N
X	B	N	F	U	L
X	U	N	F	B	L
L	U	N	F	B	**X**
U	L	N	F	B	**X**
B	L	N	F	**U**	**X**
N	L	B	F	**U**	**X**
F	L	B	**N**	**U**	**X**
L	F	B	**N**	**U**	**X**
B	F	**L**	**N**	**U**	**X**
F	B	**L**	**N**	**U**	**X**
B	**F**	**L**	**N**	**U**	**X**
B	**F**	**L**	**N**	**U**	**X**

11. [20 points] What follows is a C++ implementation of binary tree with integer data.

```
struct treenode;
typedef treenode*tree;
struct treenode
{
 int item;
 tree left = NULL;
 tree right = NULL;
}
```

Finish the C++ code, using recursion, which prints out the items of a binary tree in preorder.

```
void preorderwrite(tree root)
 {
  if(root)
   {
    cout << root->item << endl;
    preorderwrite(root->left);
    preorderwrite(root->right);
   }
 }
```