University of Nevada, Las Vegas Computer Science 477/677 Spring 2023

Answers to Final Examination May 8, 2023

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 390 points.

1. In each blank, write Θ if correct, otherwise write O or Ω, whichever is correct. [5 points each]

   (i) $n^2 = O(n^3)$

   (ii) $\log(n^2) = \Theta(\log(n^3))$

   (iii) $\log(n!) = \Theta(n \log n)$

   (iv) $\log_2 n = \Theta(\log_4 n)$

   (v) $n^{0.000000000001} = \Omega(\log n)$

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

2. True or False. Write "T" or "F." If the answer is not known to science at this time, write "O" for "Open." [5 points each]

   (i) **T** The subproblems of a dynamic program form a directed acyclic graph.

   (ii) **F** Kruskal's algorithm uses dynamic programming.

   (iii) **T** A hash function should appear to be random, but cannot actually be random.

3. Solve each recurrence, expressing the answer as an asymptotic function of n. Use O, Ω, or Θ, whichever is most appropriate. Although I haven't in previous tests, this time I will take a point off for expressing the answer without using asymptotic notation, or for using the wrong one. [5 points each]

   (i) $F(n) \geq 3F(n/9) + 1$

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

   (ii) $F(n) = F(3n/5) + 4F(2n/5) + n^2$

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

   (iii) $F(n) = 2F(n/2) + n$

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

   (iv) $F(n) = 4F(n/2) + n$

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

(v) $F(n) \geq F(n/2) + 2F(n/4) + n$

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

(vi) $F(n) \leq 4F(n/2) + n^2$

$F(n) = \Omega(n^2 \log n)$

(vii) $F(n) \leq F(\sqrt{n}) + 1$

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

(viii) $F(n) = 2F(n/4) + \sqrt{n}$

$F(n) = \Theta(\sqrt{n} \log n)$

4. Give the asymptotic complexity, in terms of n, for each of these code fragments. [5 points each]

(i)
```
for(int i = 0; i < n; i++)
  for(int j = n; j > i; j = j/2)
```

$\Theta(n)$

(ii)
```
for(int i = 0; i < n; i++)
  for(int j = i; j > 0; j = j/2)
```

$\Theta(n \log n)$

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

$\Theta(\log \log n)$

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

$\Theta(\sqrt{n})$

5. Fill in the blanks.

(i) [5 points] **Heapsort** is a fast implementation of selection sort.

(ii) [5 points] **Treesort** is a fast implementation of insertion sort.

(iii) [5 points] The recurrence $F(n) = F(n/5) + F(7n/10) + n$ is used to compute the time complexity of **median find** or **selection**.

(iv) [5 points] A **perfect** hash function fills the hash table exactly with no collisions.

(v) [5 points] **Huffman's** algorithm finds an optimal binary code for a weighted alphabet such that the code for one symbol is never a prefix of the code for another symbol.

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

(vii) [5 points] If G is a weighted digraph, it is impossible to solve any shortest path problem on G if G has a **negative weight cycle**, usually just called a **negative cycle**.

(viii) [5 points] The height of a binary tree with 45 nodes is at least **5**. (You must give the exact answer. No partial credit.)

(ix) [5 points] The following is pseudo-code for what algorithm?

selection sort

```
int x[n];
input values of x;
 for(int i = n-1; i > 0; i--)
  for(int j = 0; j < i; j++)
   if(x[i] < x[j]) swap(x[i],x[j]);
```

(x) [5 points] **Dijkstra's** algorithm does not allow the weight of any arc to be negative.

(xi) [5 points] The asymptotic time complexity of Johnson's algorithm on a weighted directed graph of n vertices and m arcs is $O(nm \log n)$. (Or $O(nm \log m)$, which means the same thing.) (Your answer should use O notation.)

(xii) [5 points] The time complexity of every comparison-based sorting algorithm is $\Omega(n \log n)$ (Your answer should use Ω notation.)

(xiii) [10 points] The prefix expression $*\,a + \sim b * -c\ d \sim e$ is equivalent to the infix expression $a * (-b + (c - d) * -e)$ and the postfix expression $a\ b \sim c\ d - e \sim * + *$

(xiv) [5 points] A **perfect** hash function fills the hash table exactly with no collisions.

(xv) [5 points] In **open hashing** there can be any number of items at a given index of the hash table.

(xvi) [5 points] In **closed** hashing, (or **open addressing**) if the position at $h(x)$ is already occupied for some data item x, a **probe** sequence is used to find an unoccupied position in the hash table.

(xvii) [5 points] In **cuckoo** hashing, each item has more than one hash value, but only uses one of them.

(xviii) [5 points] A planar graph with $n \geq 3$ vertices can have no more than $3n - 6$ edges. (Exact formula, please.)

6. [10 points] What is the loop invariant of the while loop in the following pseudocode?

$xn = ym + z$

```
float product(float x, int n)
 {
  // assert(n >= 0);
  float z = 0.0;
  float y = x;
  int m = n;
  while(m > 0)
   {
```

```
    if(m%2) z = z+y;
    m = m/2;
    y = y+y;
   }
  return z;
 }
```

7. [10 points] The following code is used as a subroutine for both quicksort and select. Assume A[n] is an array of integers. For simplicity, we assume that no two entries of A are equal. Write a loop invariant for the while loop.

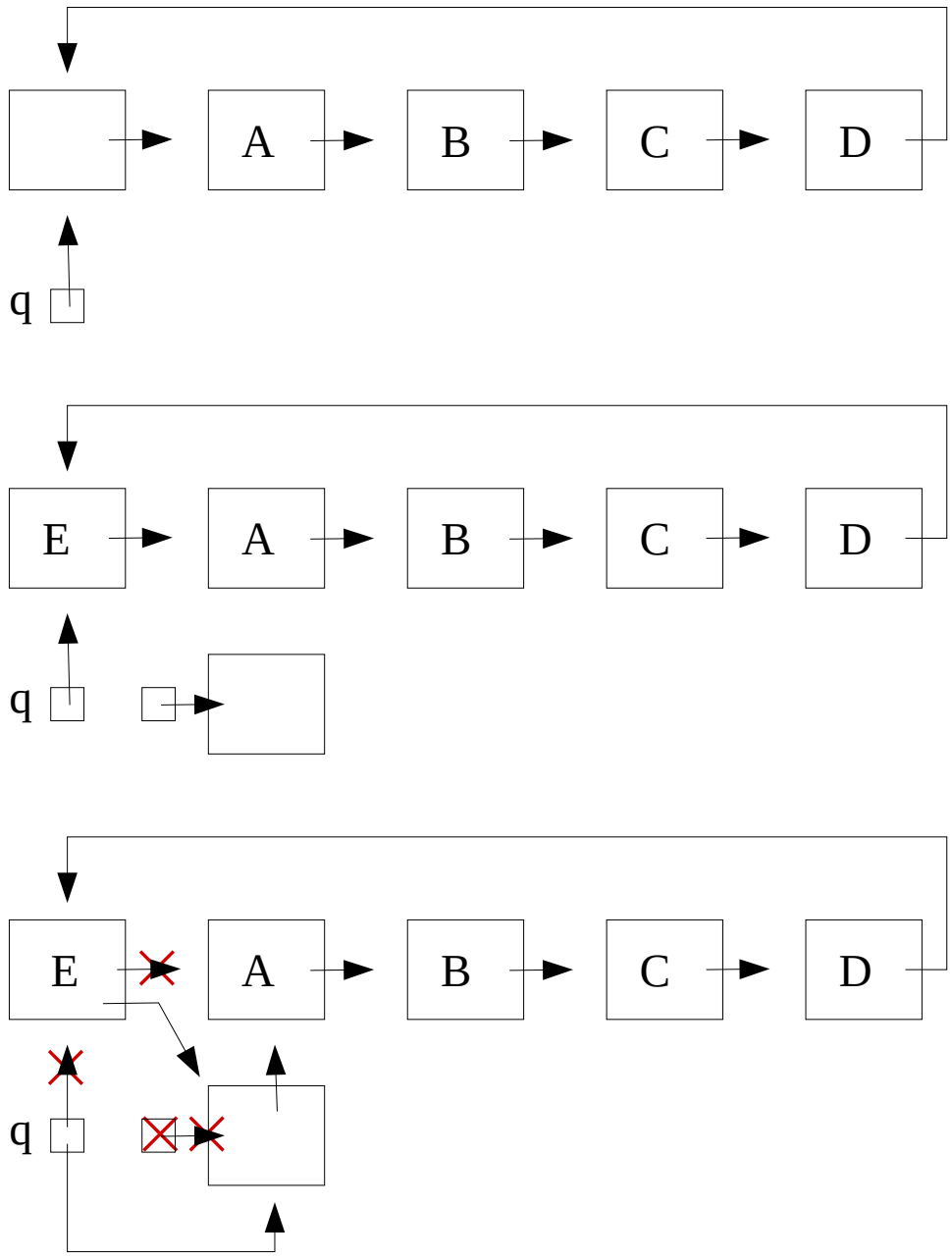
```
int pivot = A[0];
int lo = 0;
int hi = n;
while(lo < hi)
 {
  if(A[lo+1] < pivot) lo++;
  else if(A[hi] > pivot) hi--;
  else swap(A[lo+1],A[hi]);
 }
```

lo $\leq$ hi and $A[i] \leq$ pivot for all $0 \leq i \leq$ lo and $A[i] >$ pivot for all hi $< i \leq n$

8. [10 points] Use dynamic programming to compute the Levenshtein edit distance between the strings "bedafc" and "abcdef."

		a	b	c	d	e	f
	0	1	2	3	4	5	6
b	1	1	1	2	3	4	5
e	2	2	2	2	3	3	4
d	3	3	3	3	2	3	4
a	4	3	4	4	3	3	4
f	5	4	4	5	4	4	3
c	6	5	5	4	5	5	4

9. [10 points] Sketch a circular linked list with dummy node which implements a queue. The queue has four items. From front to rear, these are A, B, C, D, and show the insertion of E into the queue. Show the steps. Don't erase deleted objects; instead, simply cross them out.



10. [10 points] List properties of a good hash function used for a search structure.

    (a) Deterministic

    (b) Fast to compute

    (c) Equally spread across table, even if the data are similar. Unequal but similar data should hash independently.

11. [20 points] Write pseudo-code for the Floyd/Warshall algorithm. Let the vertices be $\{1, 2, \ldots n\}$. Let $W(i, j)$ be the weight of the arc (i, j), if any. Let $W(i, j) = \infty$ if there is no arc. Compute $V(i, j)$, the minimum weight of any path from i to j, and $B(i, j)$, the backpointer for that minimum path.

For all i and j, $V(i,j) = W(i,j)$ and $B(i,j) = i$
$V(i,i) = 0$ for all 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, and $B(i,k) = B(j,k)$
      }

12. Walk through polyphase mergesort with the array given below.

```
input:   ACBXFREYGMQSNDZ
file 1:  ACFRGMQSDZ
file 2:  BXEYN
file 3:  ABCXGMNQS
file 4:  EFRYDZ
file 5:  ABCEFRXY
file 6:  DGMNQSZ
file 7:  ABCDEFGMNQRSXYZ (sorted)
```

13. [20 points] A compiler stores an array `A[8][10][18]` into main memory in row major order, with base address B, and each entry of A requires one place in main memory. Write a formula for the main memory address of `A[i][j][k]` for integers i, j, and k within range.

    $B + i * 10 * 18 + j * 18 + k$

14. Consider the following C++ code.

```
int george(int n)
 {
  if(n == 0) return 1;
  else return george(n/2)+george(n/2-1)+n*n;
 }
```

    (a) [10 points] What is the asymptotic complexity of `george(n)`?

    The recurrence is: george$(n) = 2 *$ george$(n/2) + n^2$
    george$(n) = \Theta(n^2)$

    (b) [10 points] What is the asymptotic time complexity of the recursive code given above?

    The recurrence is: $T(n) = 2T(n/2) + 1$
    $T(n) = \Theta(n)$

(c) [10 points] What is the asymptotic time complexity of a bottom-up dynamic programming algorithm to compute `george(n)`?

Compute george(i) for all i in increasing order. The time complexity is $\Theta(n)$

(d) [10 points] What is the asymptotic space complexity of a top-down computation of `george(n)` using memoization?

We compute memos for george($n/2^p$) for all $p = 2^k \leq n$. a total of $\Theta(\log n)$ memos.

15. [20 points] Consider an array implementation of a stack of integers, as given below. Fill in the code which implements the needed operators of a stack.

```
const int N = // whatever
struct stack
 {
  int item[N];
  int size; // number of items in the stack
  // bottom of the stack is at item[0];
 };

void initialize(s&stack)
 {
  s.size = 0;
 }

void push(s&stack,int i)
 {
  s.item[s.size] = i;
  s.size ++;
 }

 bool empty{s&stack)
  {
   return s.size == 0;
  }

 int pop(s&stack)
   // input condition: s.size > 0
  {
   s.size--;
   return s.item[s.size];
  }
```

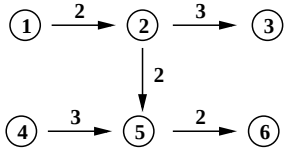
16. [10 points] In class, we implemented a minheap as an almost complete binary tree implemented as an array. Suppose the minheap is initialized as shown in the first line of the array shown below. Show the evolution of the structure when deletemin is executed.

A	C	F	D	Q	H	L	R	Z
Z	C	F	D	Q	H	L	R	
C	Z	F	D	Q	H	L	R	
C	D	F	Z	Q	H	L	R	
C	D	F	R	Q	H	L	Z	

17. [10 points] Starting from the configuration given, show the evolution of the structure when B is inserted.

C	D	F	R	Q	H	L	Z	
C	D	F	R	Q	H	L	Z	B
C	D	F	B	Q	H	L	Z	R
C	B	F	D	Q	H	L	Z	R
B	C	F	D	Q	H	L	Z	R

18. [20 points] You are given an acyclic directed graph G whose vertices the integers 1, 2, $\ldots, n$. For each arc (i, j), the arc weight $W(i, j)$ is given, and $i < j$. Give a dynamic programming algorithm which calculates the directed path through G of maximum weight. For example, in the digraph shown below, the maximum weight directed path is (1,2,5,6). You need not write pseudo-code if you can explain the algorithm without it.



Let InNbrs$(j) = \{i : (i, j) \in E\}$, the in-neighbors of j.

$V(1) = 0$
maxv $= 0$
bestlast $= 1$
for all j from 2 to n
{
$V(j) = 0$
for all $i \in$ InNbrs(j)
{
temp $= V(i) + W(i, j)$
if(temp $> V(j)$)
{
$V(j) =$ temp
back$(j) = i$
}
}
if($V(j) >$ maxv)
{
maxv$= V(j)$
bestlast$= j$
}
}

The maximum weight path ends at the vertex **bestlast** and has weight V(bestlast) = **maxv**. That path can be recovered, in backwards order, by starting at **bestlast** and following the backpointers until reaching a vertex i such that $V(i) = 0$.