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

Answers to Third Examination November 20, 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."

Please do not write so small that I can't read it!

Not everybody read this message. I needed a magnifying class for two papers!

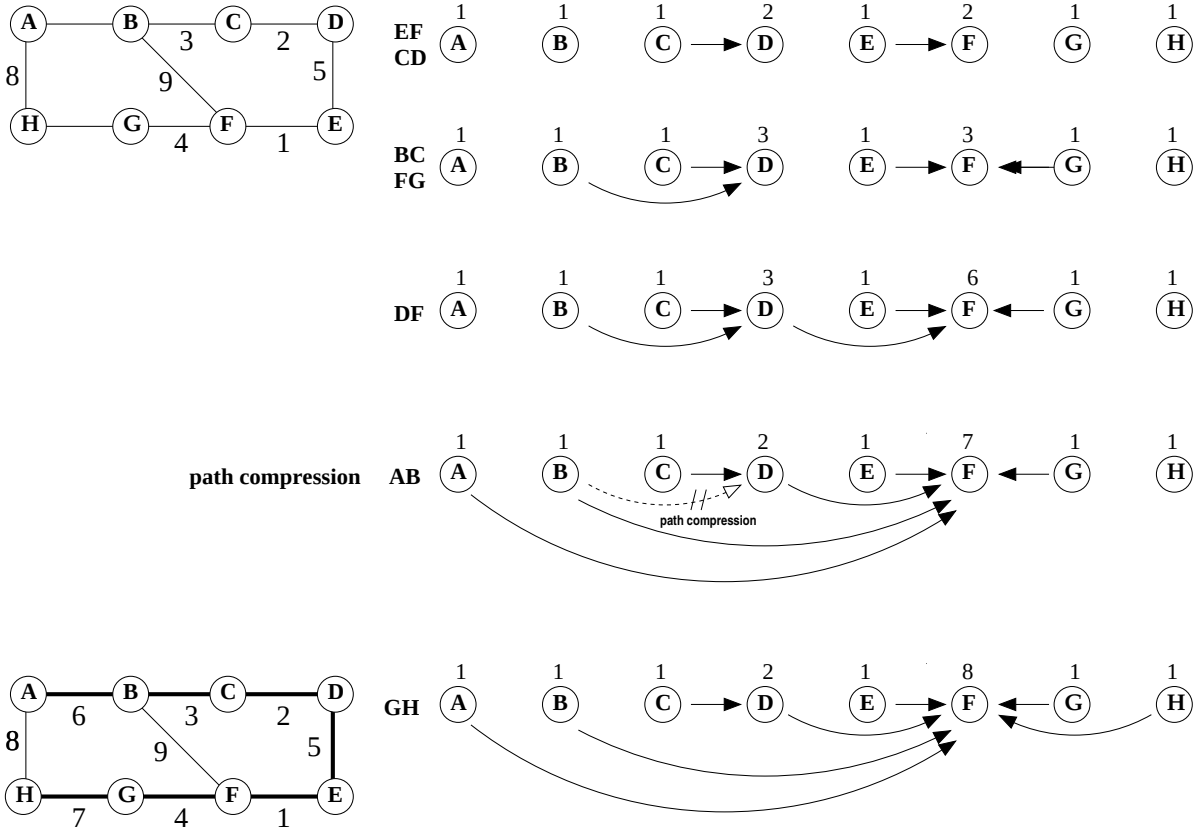
The entire examination is 395 points.

1. [10 points] What are the important properties of a good hash function?
   1. Deterministic.
   2. Appears random.
   3. Fast to compute.
2. [10 points] Name two greedy algorithms introduced in class this semester. **Huffman's Kruskal's**.
3. [10 points] In closed hashing, **collisions** can resolved by the use of probe sequences. However, probe sequences are unnecessary if a **perfect** hash function is used.
4. [5 points] In a **cuckoo** hash table, an item can eject an previously stored item, which then has to use an aternative hash value.
5. [5 points] The worst case time complexity of quicksort on a list of length n is $O(n^2)$.
6. [10 points] The height of a treap with n items is $O(n)$ in the worst case, but $O(\log n)$ in the average case.
7. [5 points] In an open hash table, the items at each index of the table must be held in a **search structure** which is frequently shown as a list.
8. [5 points] The asymptotic complexity of the Floyd/Warshall algorithm is $\Theta(n^3)$
9. [5 points] The asymptotic complexity of Dijkstra's algorithm algorithm is $O(m \log n)$
10. [10 points] A directed graph is **strongly connected** if there is a directed path from any vertex to any other vertex.
11. [5 points] **Huffman's** algorithm finds a binary code so that the code for one symbol is never a prefix of the code for another symbol.
12. [5 points] An acyclic directed graph with 8 vertices must have at least **8** strong components. (Must be exact answer.)
13. [5 points] In **open hashing** or **separate chaining** there can be any number of items at a given index of the hash table.

14. [10 points] Write the prefix expression equivalent to the infix epression $a * -(b \wedge c \wedge d) + e$
(Don't forget that $\wedge$ means exponentiation.)

$+ * a \sim \wedge b \wedge cde$

15. [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.

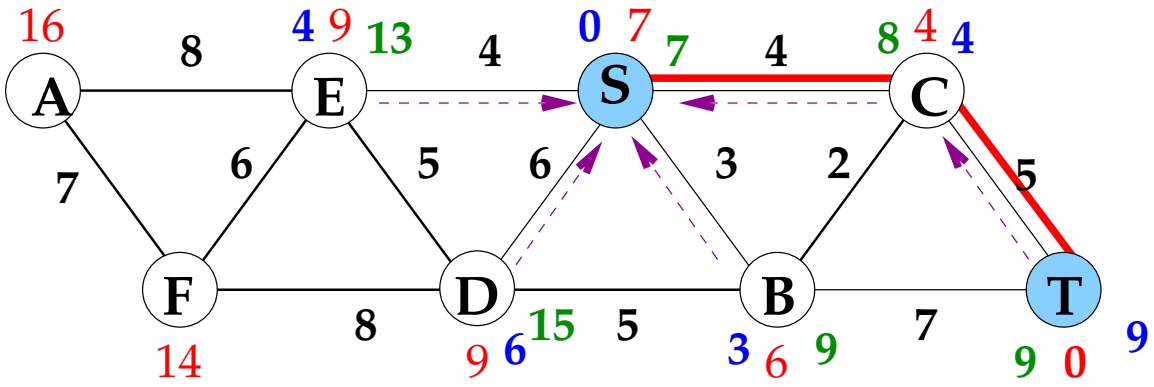


16. [20 points] A hash table has size $M = 32$. Each datum stored in the table is expressed as a string of 8 bits. A universal hash function is given by the bit-matrix shown below. Find the hash value for the datum $x = 11010110$

$$\begin{array}{l} 10111100 \\ 11010110 \\ 00001011 \\ 00011110 \\ 00111010 \end{array} \times \begin{array}{c} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{array} = \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{array}$$

$h(x) = 30$

17. [20 points] Work the A* algorithm for the following weighted digraph, where the heuristic values are circled, to avoid the need to print color. Draw pointers and write values of g and f in pencil or any color pen. I'll be able read it if you try to be neat. To make neatness easier, I have made the diagram large.



18. [20 points] You are trying to construct a cuckoo hash table of size 8 holding 8 names. Each of the names listed below has two possible hash values, as indicated in the array. Put the items into the table in alphabetic order, if possible. Instead of erasing ejected items, simply strike them out.

Abe	2	7
Ben	3	6
Deb	0	6
Eli	1	7
Kat	1	2
Mel	0	4
Ron	1	5
Sue	5	7

0	~~Deb~~ Mel
1	~~Eli~~ ~~Kat~~ ~~Ron~~ ~~Eli~~ ~~Ron~~ ~~Kat~~ ~~Eli~~ Ron
2	~~Abe~~ ~~Kat~~ ~~Abe~~ Kat
3	Ben
4	
5	~~Ben~~ ~~Sue~~ ~~Ron~~ Sue
6	Deb
7	~~Eli~~ ~~Abe~~ ~~Eli~~ ~~Sue~~ Abe

There seems to be no solution.

Proof: The 5 names Abe, Eli, Kat, Ron and Sue must share the 4 hash values 1, 2, 5 and 7. By the pigeonhold principle, there must be a collision.

19. [20 points] Explain how to implement a sparse array using a search structure.

Let A be the sparse virtual array. Let S be a search structure which contains ordered pairs of the form (i, x), where $A[i] = x$. To fetch the value of $A[i]$, search S for a pair (i, x). If that pair is found, return x, otherwise return a default value, such as 0. To store a value x for $A[i]$, search S for a pair (i, y). If that pair is found, replace y by x. That pair is not found, insert the pair (i, x) into S.

20. Consider the following recursive C++ subprogram.

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

(a) [10 points] What is the asymptotic value of the function $f(n)$ computed by the recursive code?
The recurrence is $f(n) = 2f(n/2) + n^2$.
$f(n) = \Theta(n^2)$

(b) [10 points] What is the asymptotic time complexity of the computation of $f(n)$ using the recursive code? (Hint: not the same answer as above.)
The recurrence is $T(n) = 2f(n/2) + 1$.
$T(n) = \Theta(n)$

(c) [10 points] What is the asymtotic time complexity of a computation of f(n) using memoization?

We need to compute and store the values of $f(m)$ for certain values of m. How many values? Each m must be close to $n/2^k$ for approximately $\log_2 n$ values of k. For each such k, there can be at most 3 values of m. Thus, we compute and store $\Theta(\log n)$ memos. Thus, the time complexity of the memoization algorithm is $\Theta(\log n)$ if you don't consider the cost of store and fetch. If that cost is (for example) the logarithm of the size of the search structure, our overall time complexity is $\Theta(\log n \log\log n)$.

21. True or False.

(a) [5 points] **T** In SHA256 hashing, collisions are so unlikely that it is assumed they never occur.

(b) [5 points] **F** Open hashing uses open addressing.

(c) [5 points] **F** Open hashing uses probe sequences.

(d) [5 points] **T** False overflow for a queue can be avoided by circular list implementation.

(e) [5 points] **T** If a stack is implemented as a linked list, the head of the linked list should hold the top item of the stack.

(f) [5 points] **F** Kruskal's algorithm uses dynamic programming.

(g) [5 points] **F** There will be no collisions if the size of a hash table is at least the square of the number of data items.

22. Solve each recurrence, expressing each answer in terms of n using Θ notation.

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

$3(1/3)^2 + 3(2/3)^2 = 13/9 > 1$. Therefore, we must find d such that $3(1/3)^d + 3(2/3)^d = 1$. We find that $d = 3$. Thus $F(n) = \Theta(n^3)$.

(b) [10 points] $F(n) = F(n/3) + F(2n/3) + 1$

$(1/3)^0 + (2/3)^0 = 2 > 1$. Therefore, we must find d such that $(1/3)^d + (2/3)^d = 1$. We find that $d = 1$. Thus $F(n) = \Theta(n)$

(c) [10 points] $G(n) = 2G(n/4) + \sqrt{n}$

$2(1/4)^{\frac{1}{2}} = 1$, therefore $G(n) = \Theta(\sqrt{n} \log n)$

(d) [10 points] $H(n) = \log n + 1$ $H(n) = \Theta(\log^* n)$

(e) [10 points] $H(n) = 4H(2n/5) + H(3n/5) + 2n^2$

$4(2/5)^2 + (3/5)^2 = 1$. Therefore $H(n) = \Theta(n \log n)$.

(f) [10 points] $G(n) = 4(G(n/2) + 5n^2$

$4(1/2)^2 = 1$, therefore $G(n) = \Theta(n \log n)$.

(g) [10 points] $F(n) = F(n - \log n) + \log^2 n$

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

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

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

23. Find the time complexity of each of these code fragments in terms of n, using Θ notation.

(a) [10 points]

```
for(int i = 1; i < n; i = 2*i)
 cout < "Hello world!";
```

$\Theta(\log n)$

(b) [10 points]

```
for(int i = n; i > 1; i = sqrt(i))
 cout < "Hello world!";
```

Assume that `sqrt(i)` returns the floor of the square root of i.

$\Theta(\log \log n)$

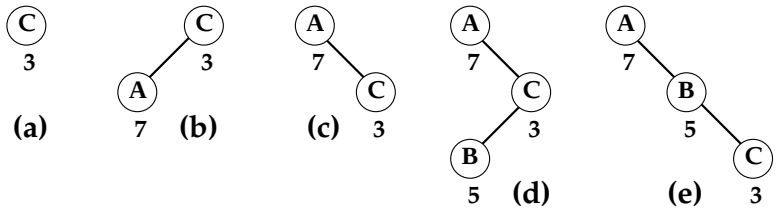
(c) [10 points]

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

$\Theta(n)$

24. [10 points]

You need to store the items C, B, and A, in that order, into a treap. The priority for A is 7, for B is 5, and for C is 3. Use maxheap order. Draw the resulting treap after each insertion, and show each rotation.



25. [20 points]j Find the Levenshtein edit distance from the word "mennoover" to the word "maneuver." Show the matrix.

		m	a	n	e	u	v	e	r
	0	1	2	3	4	5	6	7	8
m	1	0	1	2	3	4	5	6	7
e	2	1	1	2	2	3	4	5	6
n	3	2	2	1	2	3	4	5	6
n	4	3	3	2	2	3	4	5	6
o	5	4	4	3	3	3	4	5	6
o	6	5	5	4	4	4	4	5	6
v	7	6	6	5	5	5	4	5	6
e	8	7	7	6	5	6	5	4	5
r	9	8	8	7	6	6	6	5	4

		m	e	n	n	o	o	v	e	r
	0	1	2	3	4	5	6	7	8	9
m	1	0	1	2	3	4	5	6	7	8
a	2	1	1	2	3	4	5	6	7	8
n	3	2	2	1	2	3	4	5	6	7
e	4	3	2	2	2	3	4	5	5	6
u	5	4	3	3	3	3	4	5	6	6
v	6	5	4	4	4	4	4	4	5	6
e	7	6	5	5	5	5	5	5	4	5
r	8	7	6	6	6	6	6	6	5	4

The edit distance is 4.