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

Answers to Examination April 10, 2024

The entire examination is 420 points.

1. True or False.

   (a) [5 points] **F** If there are 100 data items and 1000 possible hash values, a collision is so unlikely that you can, in practice, assume that it won't happen.

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

   (c) [5 points] **F** You can avoid collisions in a hash table by making the table twice as large as the data set.

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

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

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

   (g) [5 points] **T** A hash function should appear to be random, but cannot actually be random.

2. Fill in the blanks.

   (a) [5 points] In closed hashing, collisions are resolved by the use of **probe** sequences.

   (b) [10 points] (3) Which of the following three statements is closest to the truth?
   (1) In SHA256 hashing, collisions are impossible.
   (2) In SHA256 hashing, collisions occur no more than once a year in practice.
   (3) In SHA256 hashing, collisions are so unlikely that industry experts claim they never occur.

   (c) [5 points] The worst case time complexity of quicksort on a list of length n.

   $O(n^2)$

   (d) [5 points] The average case time complexity of quicksort on a list of length n, if pivots are chosen at random.

   $\Theta(n \log n)$

   (e) [5 points] A directed graph is defined to be **strongly connected** if, given any two vertices x and y, the graph contains a path from x to y.

   (f) [10 points] In an open hash table of size m holding n data items, the items at each index of the table are typically shown as linked list. However, that structure is only efficient if m/n is fairly small. In general, we should use a **search structure** at each table index.
   Pick one of these answers:
   heap
   stack
   search structure

(g) [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.

(h) [5 points] An acyclic directed graph with 9 vertices must have at least **9** strong components. (Must be exact answer.)

(i) [5 points] In **open hashing** or **separate chaining** there can be any number of items at a given index of the hash table. $O(n)$.

(j) [5 points] The asymptotic complexity of the Floyd/Warshall algorithm is ______________________.
$\Theta(n^3)$

(k) [5 points] The asymptotic complexity of Dijkstra's algorithm algorithm is $O(m \log n)$

3. For each of these recursive subprograms, write a recurrence for the time complexity, then solve that recurrence.

(a) [10 points]

```
void george(int n)
 {
  if(n > 0)
   {
    for(int i = 0; i < n; i++) cout << "hello" << endl;
    george(n/2); george(n/3);
   }
 }
```

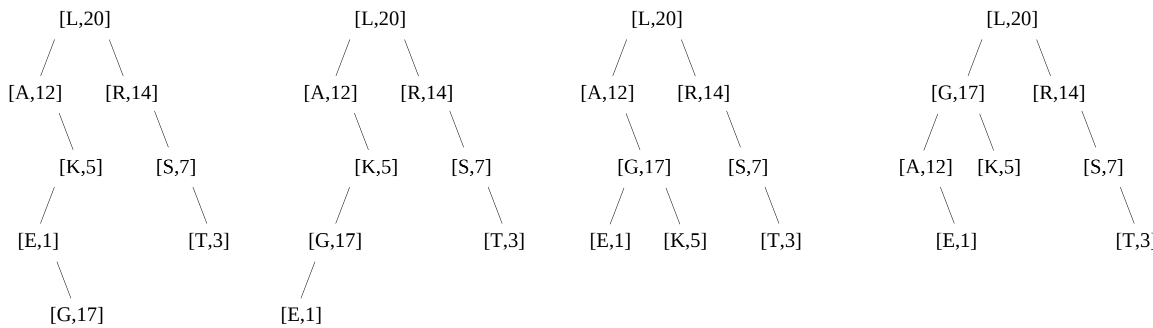
$T(n) = T(n/2) + T(n/3) + n$
$T(n) = \Theta(n)$

(b) [10 points]

```
void martha(int n)
 {
  if(n > 0)
   {
    martha(2n/3);
    martha(n/3);
    for(int i = 1; i < n; i++)
     cout << "hello world";
   }
 }
```

$T(n) = T(2n/3) + T(n/3) + n$
$T(n) = \Theta(n \log n)$

4. [20 points] The figure below shows a treap, where the data are letters and the nodes of the tree are memos, where the first component is the *key*, a letter, and the second component is a the *priority*, a random integer. Insertion of the letter G, where the priority is chosen (at random) to be 17. Show the steps.



5. [10 points] Write the prefix expression equivalent to the infix epression $-a * b - (-c - d) \wedge e$ (Don't forget that $\wedge$ means exponentiation.)

$- * \sim ab \wedge - \sim cde$

Some people wrote postfix instead. I gave partial credit. That answer is:

$a \sim b * c \sim d - e \wedge -$

6. Solve each recurrence, expressing each answer in terms of O, Ω, or Θ, whichever is most appropriate.

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

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

(b) [10 points] $H(n) = \log n + 1$

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

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

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

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

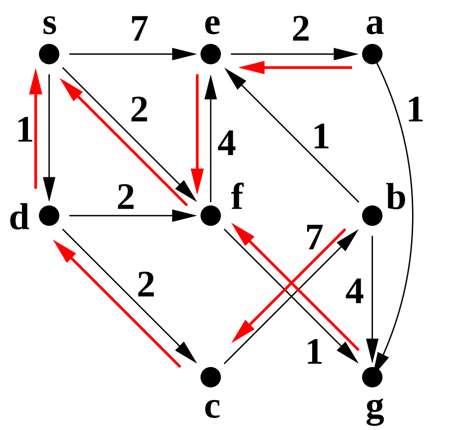
(d) [10 points] $F(n) = F(n - \log n) + \log^2 n$

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

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

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

7. [20 points] Walk through Dijkstra's algorithm for the following graph.



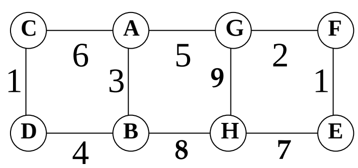
s	a	b	c	d	e	f	g
0	8	10	3	1	~~7~~ 6	2	3
*	e	c	d	s	~~s~~ f	s	f

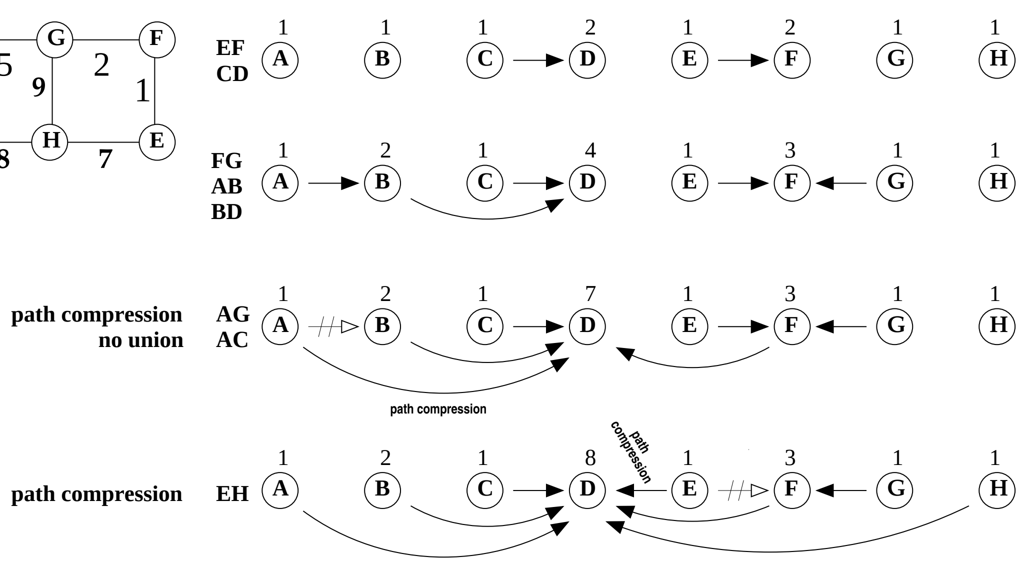
8. [20 points] Explain how to implement a sparse array using a search structure. Let A be a sparse array. The search stucture hold ordered pairs (i,x) where A[i] = x.

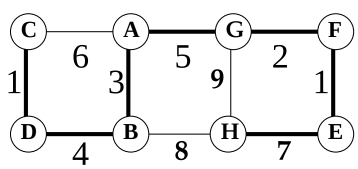
Fetch: Search for A[i]: Find a pair (i,x) and return x. If no such pair exists, return a default value.

Store: To store A[i] = x: Find a pair (i,y) and replace y by x. If no such pair is found, insert the pair (i,x) into the search structure.

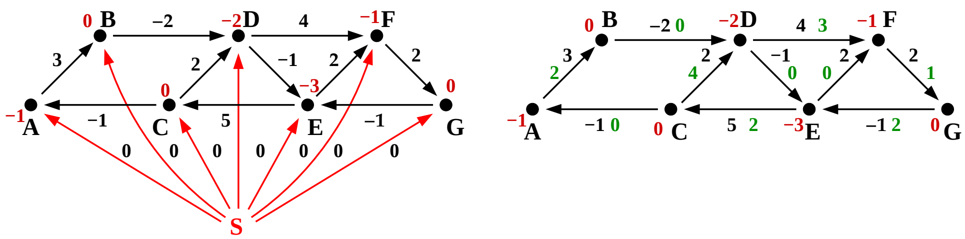
9. [20 points] Walk through Kruskal's algorithm to find the minimum spanning tree of the weighted graph shown below. Show the evolution of the union/find structure. Whenever there is choice between two edges of equal weight, choose the edge which has the alphabetically largest vertex. Whenever there 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.







10. [20 points] The left-hand figure below shows an instance of the all-pairs minpath problem. Work the first part of Johnson's algorithm on that graph, and show the adjusted weights in the right-hand figure. Do not complete the computation of Johnson's algorithm.



All red numbers must be ≤ 0, and all green numbers must be ≥ 0.

11. [20 points] Write pseudocode for the Bellman Ford algorithm. Be sure to include the shortcut that stops execution when further computation is unnecessary.

```
For all i from 1 to n V[i] = infinity
V[0] = 0
bool finished = false
while not finished
  {
   finished = true;
   For all j from 1 to m
     {
      temp = V[S[k]] + W[k]
      if(temp < V[T[k]])
        {
         V[T[k]] = temp
         back[T[k]] = S[k]
         finished = false
        }
     }
```

12. Solve each recurrence, giving asymptotic answers, using O, Ω, or Θ, whichever is most appropriate.

(a) [10 points] $F(n) \leq 4F(n/2) + n^2$

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

(b) [10 points] $G(n) \geq G(4n/5) + G(3n/5) + n^2$

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

13. [20 points] Execute heapsort for the list DNHVELX. Show the array at each step, and identify the step at which the array is a heap for the first time.

D	N	H	V	E	L	X	
D	N	X	V	E	L	H	
D	V	X	N	E	L	H	
X	V	D	N	E	L	H	
X	V	L	N	E	D	H	heapify finished
H	V	L	N	E	D	X	
V	H	L	N	E	D	X	
V	N	L	H	E	D	X	
D	N	L	H	E	V	X	
N	D	L	H	E	V	X	
N	H	L	D	E	V	X	
E	H	L	D	N	V	X	
L	H	E	D	N	V	X	
D	H	E	L	N	V	X	
H	D	E	L	N	V	X	
E	D	H	L	N	V	X	
D	E	H	L	N	V	X	

14. Give the asymptotic complexity, in terms of n, for each of these code fragments.

(a) [10 points]

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

(b) [10 points]

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

$\Theta(n)$

(c) [10 points]

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

$\Theta(n \log n)$

15. [10 points] If $A[5][7]$ is stored in column-major order, how many predecessors does $A[3][4]$ have?

    4*5 + 3 = 23

16. Consider the following recursive C++ function.

```
int f(int n)
 {
  if(n > 0) return f(n/2)+f(n/4)+f(n/4 + 1)+n;
  else return 0;
 }
```

(a) [10 points] What is the asymptotic complexity of f as a function of n, using Θ notation?

The recurrence is $f(n) = f(n/2)+2f(n/4)+n$ By the generalized master theorem, $f(n) = \Theta(n \log n)$.

(b) [10 points] What is the asymptotic time complexity of this code as a function of n, using Θ notation?

The recurrence is $T(n) = T(n/2) + 2T(n/4) + 1$ By the generalized master theorem, $T(n) = \Theta(n)$. Ans $\Theta(n)$

(c) [10 points] The following dynamic program computes $f[i]$ for all i.

```
f[0] = 0;
for(int i = 1; i <= n; i++)
  f[i] = f[i/2] + f[i/4 + 1] + i;
```

What is the asymptotic time complexity of that code as a function of n, using Θ notation?

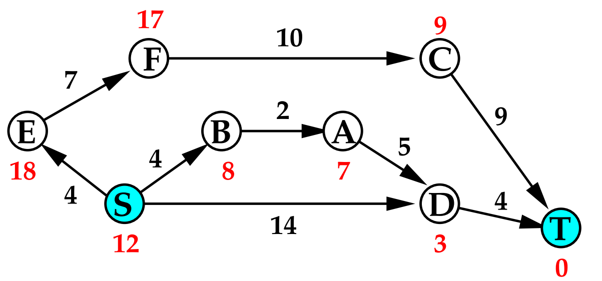
```
f[0] = 0;
for(int i = 1; i <= n; i++)
  f[i] = f[i/2] + f[i/4] + f[i/4 + 1] + i;
cout << f[n] << endl;
```

The value of $f(i)$ is computed for each i up to n. The answer is $\Theta(n)$. Ans $\Theta(n)$

Represent the subproblem f[i] by the integer i. There is one subproblem for each integer from 0 to n. The subproblems are the vertices of a directed graph. There is an arc from i to j if the computation of f[j] requires the value of f[i]. We need to find the number of predecessors of n in this directed graph. It helps to work out an example. Let n = 1785. We need to compute f for the following integers: 1785, 892, 446, 447, 223, 224, 111, 112, 55, 56, 27, 28, 29, 13, 14, 15, 6, 7, 8, 3, 4, 1, 2, 0.

Except for the smallest few, the predecessors are in blocks where each block starts with n divided by a power of 2 and has at most three members. Thus the number of predecessors is approximately $3\log_2$n. Thus the number of memos stored is $\Theta(\log n$. The search time needed is $O(\log n \log\log n)$ if the time required for a search is asymptotically the logarithm of the size of the search structure. Thus the time complexity is $O(\log n \log\log n)$.

17. [20 points] Walk through the A^* algorithm for the weighted directed graph shown below, where the pair is (S, T). The heuristic is shown as red numerals.



.

Show the arrays and the contents of the heap at each step. h is the heuristic, f is the current distance from the source, g is the sum of h and f, while back is the backpointer.

Heap: S

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0							
g	12							
back								

Heap: BDE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0		4		14	4		
g	12		12		17	22		
back			S		S	S		

Heap: ADE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0	6	4		14	4		
g	12	13	12		17	22		
back		B	S		S	S		

Heap: DE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0	6	4		11	4		
g	12	13	12		14	22		
back		B	S		A	S		

Heap: TE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0	6	4		11	4		15
g	12	13	12		14	22		15
back		B	S		A	S		D

Heap: E

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0	6	4		11	4		15
g	12	13	12		14	22		15
back		B	S		A	S		D

T is fully processed, and we are done. The shortest path from S to T is (S,B,A,D,T) obtained by following the back pointers.