

Name: \_\_\_\_\_

UW NetID: \_\_\_\_\_

**CSE 332 Summer 2018: Final Exam Part 2**  
(closed book, closed notes, no calculators)

**Instructions:** Read the directions for each question carefully. We can only give partial credit based on the work you write down, so show your work.

For questions where you are drawing pictures, please circle your final answer.

Unless otherwise noted, any algorithms or code you write should be as efficient as possible, both in  $O()$  terms and with respect to constant factors.

Take a deep breath.

Every tree is a forest.

You got this.

**Good Luck!**

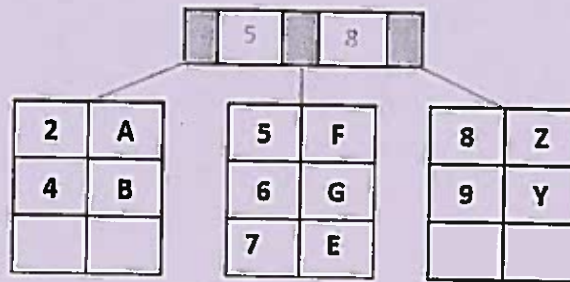
Total: 80 points. Time: 60 minutes.

Question	Max Points	Score
7. B-Trees	6	
8. Hash Tables	13	
9. Minimum Spanning Trees	14	
10. Shortest Paths	7	
11. Graph Representations	14	
12. Using Graphs	12	
13. P vs. NP	14	
<b>Day 2 Total</b>	<b>80</b>	

## 7 B-Trees

[6 points]

Below is a B-Tree storing integer keys with character values. Insert the (key, value) pairs (3,F) and (1,W) in that order into the B-Tree below. You should draw two B-trees: one for after (3,F) was inserted and one after both were inserted.



## 8 Hashing

[13 points]

Insert the following keys into the hash tables below, where the hash function is just  $\%TableSize$ . In the top table, resolve collisions with quadratic probing. In the bottom table resolve by separate chaining into a sorted linked list (with the smallest element at the head of the list). If an insertion fails, record which key failed, but attempt to insert any later keys in the list. Do not resize the tables. [8 points]

Insert these keys: 22, 14, 32, 42, 37, 13

0	1	2	3	4	5	6	7	8	9

0	1	2	3	4	5	6	7	8	9

What are the load factors of the top and bottom hash tables? [1 point]

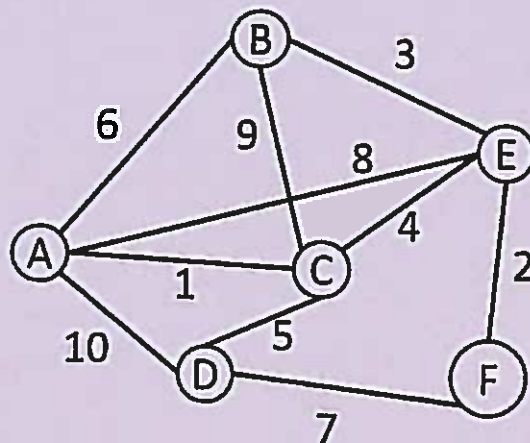
Under what conditions can we guarantee that insertion will succeed in a hash table using quadratic probing? [2 points]

What is the worst case running time to insert into each of these hash tables? Justify your answer (with 1-2 sentences) in each case. [2 points]

## 9 MSTs

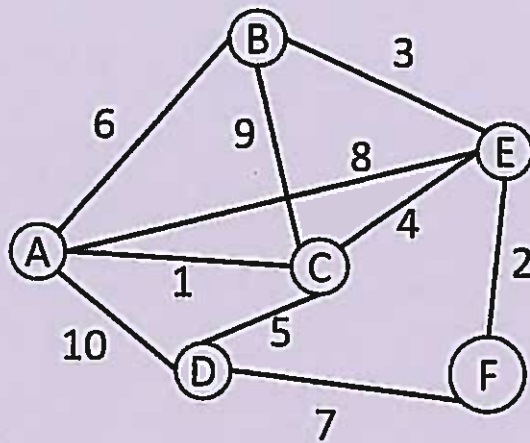
[14 points; 7 points per algorithm]

In class, we discussed two algorithms for finding a minimum spanning tree of a graph. Execute both algorithms for the graph below. We have provided space to show your work. You must show your work, crossing out old variable values when they are overwritten (as shown in lecture and section). Highlight the edges that end up in the final spanning tree.



Vertex	Distance	Best Edge	Processed
A			
B			
C			
D			
E			
F			

Which algorithm did you execute?



Edge	Include?	Reason
(A,C)		
(E,F)		
(B,E)		
(C,E)		
(C,D)		
(A,B)		
(D,F)		
(A,E)		
(B,C)		
(A,D)		

Which algorithm did you execute?

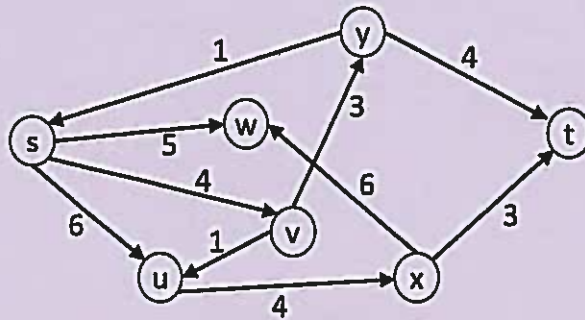


## 10 Shortest Paths

[7 points]

Run Dijkstra's algorithm on the graph below with starting point  $s$ . Fill out the table. As you're performing the algorithm cross out previous values as shown in lecture and section.

If you ever need to break a tie between two vertices in your algorithm, choose the one which is first alphabetically.



Vertex	Distance	Predecessor	Processed
s			
t			
u			
v			
w			
x			
y			

What is the shortest path from  $s$  to  $t$ ? Describe how you found it using the results of the algorithm.

## 11 Graph Representation

[14 points; 7 points each]

1. Hank Levy and the rest of the CSE leadership team need your help. A very important donor wants to tour CSE2 tomorrow, but there's a problem: they haven't started carpeting the hallways, and it would be embarrassing for the donor to step on uncarpeted floors. Hank's goal is to carpet enough hallways to allow him to give a tour (using only carpeted hallways), and to get the carpeting done as quickly as possible. Hank shows you a graph representation of the building. He has a vertex for each room the donor *simply must see* and edges representing the unfinished hallways connecting them. Each edge is also labeled with the time it would take to carpet that hallway. Hank has a (single) carpeting team ready to work until the donors arrive, but they need to know which hallways to carpet and in what order to allow for the tour to be fully-carpeted.

Will you be able to tell Hank how to direct the carpeting team? If so, describe an efficient algorithm to run, and briefly explain why it produces the right answer. You may call any of the graph algorithms we've discussed in class. If not, informally argue why you shouldn't be expected to find the optimal carpeting scheme.



2. Puppy Dubs's handlers have decided he doesn't follow enough accounts on Twitter. They tell you they'd like to follow any account that:

- Follows Puppy Dubs or
- Follows someone who follows Puppy Dubs.

You are given a graph representation of Twitter, stored as an adjacency list (see class definitions below). In the graph there is an edge from user1 to user2 if and only if user1 follows user2. Recall that "follows" is not symmetric, so the graph is directed.

Give pseudocode for an efficient algorithm to find everyone Puppy Dubs should follow. You may call any of the graph algorithms we've discussed in class. You may assume all classes have appropriate constructors and iterators.

**HINT:** the most efficient algorithm begins by altering the graph.

```
public class Graph{
    Vertex[] Vertices;
    //methods and constructors omitted.
}
```

```
public class Vertex{
    HashTable<V> outNeighbors;
    //any other fields required to run your algorithms.
}
```

Continue your algorithm on the next page if you need more space.

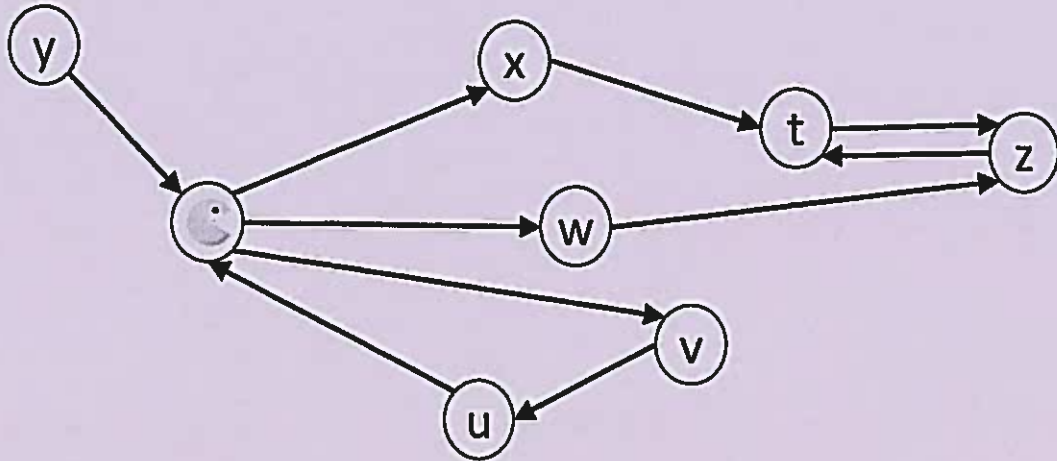
**Continue your algorithm description here, if necessary**

**What is the running time of your algorithm (briefly justify your answer)?**

## 12 Using Graphs

[12 points]

Pac-Man just arrived in a new world, represented as a directed graph. Each vertex represents a location with a single Pac-Dot (the food that Pac-Man eats), and there is an edge from vertex  $u$  to vertex  $v$  if Pac-Man can move from  $u$  to  $v$ . Pac-Man would like to eat all of the Pac-Dots, but he is worried that the graph may not be connected enough.

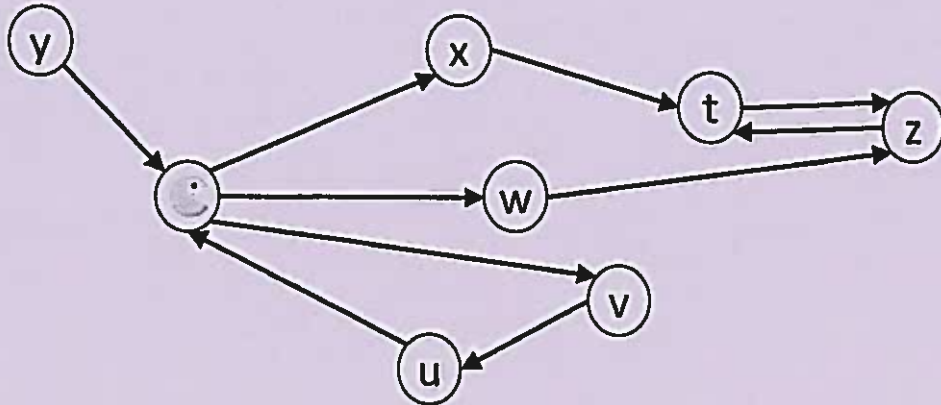


What are the strongly connected components of the graph above? The name of the vertex Pac-Man is sitting on is  $s$ . [3 points]

On this particular graph (with Pac-Man at his current location) can he eat all the Pac-dots? Justify your answer [2 points]

[7 points] There is a general strategy (i.e. an efficient algorithm) for Pac-Man to tell whether he can eat all the Pac-Dots. Describe such an algorithm. You may use any algorithms discussed in class as black boxes.

This algorithm is trickier than previous questions. We will award partial credit for answers which handle special cases. We recommend you think about the example graph:



What is the running time of your algorithm (justify your answer)?

### 13 P, NP

[14 points]

1. What do the letters NP stand for? [2 points]
2. For the following problems, circle all the classes to which the problem is known to belong [2 points each]

Determine if the minimum spanning tree of a graph has a total weight of at least  $k$     P            NP            NP-complete            NP-hard

Given a graph, determine if there is a path of length at least  $k$  from  $s$  to  $t$ .    P            NP            NP-complete            NP-hard

Given a graph, determine if there is a pair of vertices at distance at least  $k$  from each other    P            NP            NP-complete            NP-hard

Briefly justify why each of these statements are true or false. [2 points each]

3. True or false: If we discover an efficient (i.e. polynomial time) algorithm for any NP problem, then we will have an efficient algorithm for all problems in NP.
4. True or false: If  $P = NP$ , then all decision problems can be solved in polynomial time.
5. True or false: NP-complete problems are rare – only a few dozen are known.



Blank page for scratch work.

## Some Useful Facts

When we're using the tree method to solve a recurrence, we usually use the following steps:

0. Draw a few levels of the tree.
1. Let the root node be at level 0. Give a formula for the size of the input at level  $i$ .
2. What is the number of nodes at level  $i$ ?
3. What is the work done at the  $i^{\text{th}}$  recursive level?
4. What is the last level of the tree?
5. What is the work done at the base case?
6. Write an expression for the total work done.
7. Simplify until you have a "closed form" (i.e. no summations or recursion).

Geometric series identities:

$$\sum_{i=0}^k c^i = \frac{c^{k+1} - 1}{c - 1} \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1 - c} \text{ if } |c| < 1$$

Common Summations:

$$\sum_{i=0}^n i = \frac{n(n+1)}{2} \quad \sum_{i=0}^n i^2 = \frac{n(n+1)(2n+1)}{6} \quad \sum_{i=0}^n i^3 = \frac{n^2(n+1)^2}{4}$$

Log identities:

$$a^{\log_b(c)} = c^{\log_b(a)} \quad \log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$

**Master Theorem:**

Given a recurrence of the following form:

$$T(n) = \begin{cases} d & \text{if } n \leq \text{some constant} \\ aT(n/b) + n^c & \text{otherwise} \end{cases}$$

with  $a, b, c$  are constants.

If  $\log_b(a) < c$  then  $T(n)$  is  $\Theta(n^c)$

If  $\log_b(a) = c$  then  $T(n)$  is  $\Theta(n^c \log n)$

If  $\log_b(a) > c$  then  $T(n)$  is  $\Theta(n^{\log_b(a)})$