Goodluck! THANKS FOR THE GREAT CLASS!
-Kevin

Personal Information:

Name: KEY

Student ID #: [redacted]

-This exam is closed note. You may not use any computing devices including calculators. Code will be graded on proper behavior/output and not on style, unless otherwise indicated.
-If you write your answer on scratch paper, please clearly write your name on every sheet and write a note on the original sheet directing the grader to the scratch paper. We are not responsible for lost scratch paper or for answers on scratch paper that are not seen by the grader due to poor marking.

Problem	Description	Earned	Max
0	Mechanical		7
1	Preserving Abstract		10
2	Critical Thinking		12
3	Runtimes		10
4	Minimum Spanning		8
5	Radix		6
6	Sorting		9
7	Parallelism		8
8	Graphs		14
9	MST		6
10	Critical Thinking		10
TOTAL	**Total Points**	100	**100**

0. Mechanical

a) Of the options below, circle the Abstract Data Types

1. Stack
2. LinkedList with first element representing top of a Stack
3. Priority Queue
4. MinBinaryHeap
5. HashTable
6. UpTree
7. Dictionary

b) For each of the following, circle the asymptotically tighter (more specific, smaller domain of possibilities) runtime bound. If they are equally tight, circle "the same"

1. **$O(n\log(n))$ vs $O(n\log(\log(n)))$ or the same**
2. **$O(2^n)$ vs. $O(n^2)$ or the same**
3. **$\theta(n^3)$ vs. $O(n^3)$ or the same**
4. **$O(n\log^2(n))$ vs. $O(n^2\log(n))$ or the same**

1. Preserving Abstractions

Given the following code, write client code using DogPark such that a call to okToPlay() encounters a NullPointerException. Then explain briefly how to prevent a NullPointerException from being called by client code. *Hint: There are multiple potential causes for a NullPointerException, please fix all of them.*

```
class Dog {
	public String name;
	public String[] breeds;

	public Dog(String n, String[] b) {
		name = n;
		breeds = b;
	}
}

class DogPark {
	private Dog resident;
	private String[] acceptedBreeds;

	public DogPark(Dog d, String[] b) {
		if (d == null || d.breeds == null || b == null) {
			throw new IllegalArgumentException();
		}
		resident = d;
		acceptedBreeds = b;
	}

	public boolean okToPlay() {
		for (String b : resident.breeds) {
			boolean found = false;
			for (String ab : acceptedBreeds) {
				if (b.equals(ab)) {
					found = true;
				}
			}
			if (!found) {
				return false;
			}
		}
		return true;
	}
}
```

(Please write answer on the next page)

Please write your answer to Question 1 on this page

Ways to cause NullPointerError

1)
```
DogPark dp = new DogPark(d, b);
dp.breeds = null;
dp.okToPlay();
```

2)
```
d.breeds[1] = null;
DogPark dp = new DogPark(d, b);
dp.okToPlay();
```

3)
```
DogPark dp = new DogPark(d, b)
d.breeds[1] = null;
```

Fixes

1) Make breeds field final

2) check for null entries in Dog constructor. Check for null entries in d.breeds in DogPark

3) Copy-in Dog object when creating DogPark.

2. Critical Thinking I
Given a value 'x' and an array of integers, return whether any two of the numbers add up to 'x'. For example:

input array: [3, 11, 56, 4, 2, 9, 1, 7]
x: 20
result: True

You will solve this question 2 times. First, optimize the minimization of runtime. Second, optimize the minimization of auxiliary space. You may write java code or psuedocode. Your psuedocode should roughly resemble real code structure and syntax. It is better to have an inefficient solution than no solution at all. You may assume the array is not empty or null. There is additional space on the next page.

ONE POSSIBLE SOLUTION

Solution 1 (optimize for runtime):

```
boolean findSum (int x, int[] nums)
    Set s = new Set
    for (int n in nums)
        s.add (n)
    for (int n in s)
        if (s.contains (x - n))
            return true
    return false
```

Solution 2 (optimize for space): ONE POSSIBLE SOLUTION

```
boolean findSum (int x, int[] nums)
    sort (nums)
    left = 0
    right = nums.length - 1
    while left < right:
        if nums[left] + nums[right] == x:
            return true
        else if nums[left] + nums[right] < x
            left ++;
        else
            right --
    return false
```

3. Runtimes

Fill in this table with the **worst-case** asymptotic running time of each operation when using the data structure listed. Assume the following:

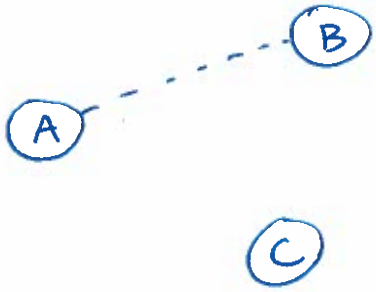
- Items are comparable (given two items, one is less than, equal, or greater than the other) in O(1) time.
- For insertions, it is the client's responsibility not to insert an item if there is already an equal item in the data structure (so the operations do not need to check this).
- For insertions, assume the data structure has enough room (do not include any resizing costs).
- For deletions, assume we do not use lazy deletion.

we didn't penalize for logic

	Insert	Lookup	Delete	getMin	getMax
	(take an item and add to the structure)	(take an item and return if it is in the structure)	(take an item and remove it from the structure if present)	(return smallest item in the structure)	(return largest item in the structure)
sorted array	O(N) shift	O(log N)	O(N) shift	O(1)	O(1)
unsorted array	O(1)	O(N)	O(N)	O(N)	O(N)
array minHeap	O(log n)	O(N)	must find first O(N)	O(1)	O(N)
AVL Tree	O(log N)	→	→	→	→
hashtable using separate Chaining	O(1) prepend O(n) append	O(N)	O(N)	O(N)	O(N)

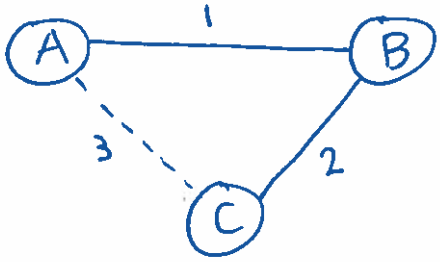
4. Minimum Spanning

(a) Draw a weighted undirected graph with exactly 3 nodes that has exactly 0 minimum spanning trees.



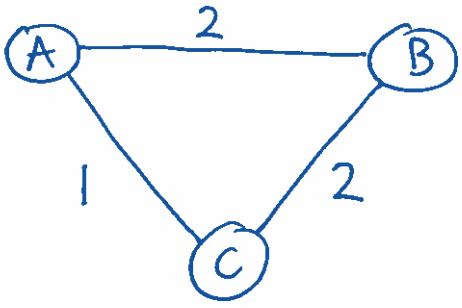
- 0 or 1 edges

(b) Draw a weighted undirected graph with exactly 3 nodes that has exactly 1 minimum spanning tree.



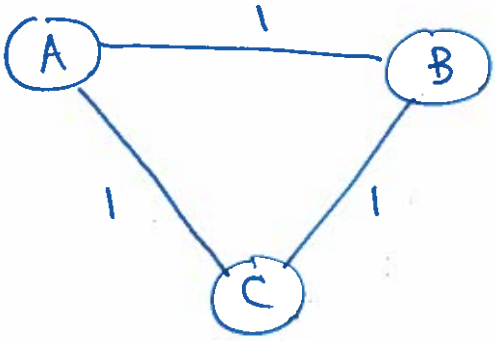
* Don't need 3rd edge, but if they do, must be bigger cost than other 2

(c) Draw a weighted undirected graph with exactly 3 nodes that has exactly 2 minimum spanning trees.



- 2 equal edge costs
- 1 different/lower edge cost

(d) Draw a weighted undirected graph with exactly 3 nodes that has exactly 3 minimum spanning trees.



- 3 edges all with same cost

5. Radix Sort

Suppose we use radix sort to sort the numbers below, using a radix of 10. Show the state of the sort after each of the first two passes, not after the sorting is complete.

Numbers to sort (in their initial order):

17, 58, 8, 1, 1872, 103, 98, 97, 1456, 21, 777, 3, 2015, 96

Results after first pass:

0	1	2	3	4	5	6	7	8	9
	1 21	1872	103 3		2015	1456 96	17 97 777	58 8 98	

Results after second pass:

0	1	2	3	4	5	6	7	8	9
1 103 3 8	2015 17	21			1456 58		1872 777		96 97 98

6. Sorting short answer:
In all questions about quick-sort, assume n is large enough that any use of a cut-off is not relevant. You do not need to justify your answer.

(a) What is the worst-case asymptotic running time of heap-sort?

$O(n \log(n))$

(b) What is the worst-case asymptotic running time of merge-sort?

$O(n \log(n))$

(c) What is the worst-case asymptotic running time of quick-sort?

$O(n^2)$ when input constructed in a way such that median of 3 or first index is always a bad choice.

(d) Can heap-sort be done in-place?

Yes

(e) Can quick-sort be done in-place?

Yes

(f) Consider one item in an array that is sorted with mergesort. In asymptotic (big-Oh) terms, how many times can that one item move to a different location?

$O(\log(N))$

(g) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the leftmost element in the range-to-be-sorted?

$O(N^2)$

(h) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the rightmost element in the range-to-be-sorted?

$O(N^2)$

(i) What is the asymptotic running time of quick-sort if the array is already sorted (or almost sorted) and the pivot-selection strategy picks the middle element in the range-to-be-sorted?

$O(n \log(n))$

7a. Parallelism and Concurrency

For each of the following, indicate whether it can be easily computed with:

- A parallel map operation
- A parallel reduce operation
- Neither

a) Replace each string in an array with an all-capitals version of the same string MAP

b) The number of strings in an array that are not the empty string Reduce

c) The median of an array of numbers neither

d) The largest index in an array of numbers that holds a negative number Reduce

7b. True or False

a) A computer with 1 processor (only 1 core) can make use of parallelism (TRUE / (FALSE))

b) A computer with 1 processor (only 1 core) can make use of concurrency ((TRUE) / FALSE)

c) For a given block of code, the **span** is always less than the **work** (TRUE / (FALSE)) span and work can be equal

d) For a given block of code, the **work** is always less than the **span** (TURE / (FALSE))

8. Graph Representations

These three questions about graphs all have the same subparts. Note that for parts (iii), (iv), and (v), your answer should be in terms of an arbitrary k, not assuming k = 4.

(a) Suppose a directed graph has k nodes, where each node corresponds to a number (1, 2, ..., k) and there is an edge from node i to node j if and only if i < j.

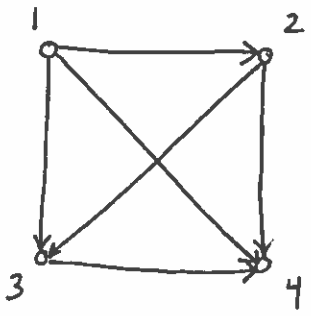
i. Draw the graph (using circles and arrows) assuming k = 4.
ii. Draw an adjacency matrix representation of the graph assuming k = 4.
iii. In terms of k, exactly how many edges are in the graph?
iv. Is this graph dense or sparse?
v. In terms of k (if k is relevant), exactly how many correct results for topological sort that does this graph have? (assuming an arbitrary value of k)

(iii) $\sum_{i=1}^{k-1} i$

(iv) Dense

(v) 1

(i)



(ii)

	1	2	3	4
1	F	T	T	T
2	F	F	T	T
3	F	F	F	T
4	F	F	F	F

(b) Suppose a directed graph has k nodes, where one "special" node has an edge from itself to every other node except itself and there are no other edges at all in the graph.

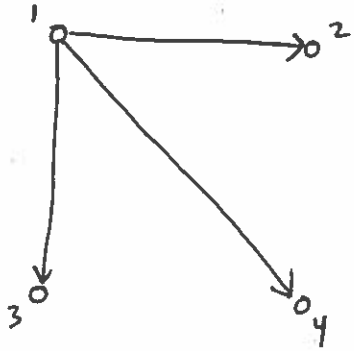
i. Draw the graph (using circles and arrows) assuming k = 4.
ii. Draw an adjacency matrix representation of the graph assuming k = 4.
iii. In terms of k, exactly how many edges are in the graph?
iv. Is this graph dense or sparse?
v. In terms of k (if k is relevant), exactly how many correct results for topological sort that does this graph have? (assuming an arbitrary value of k)

(ii) k-1

(iv) Sparse

(v) (k-1)!

(i)



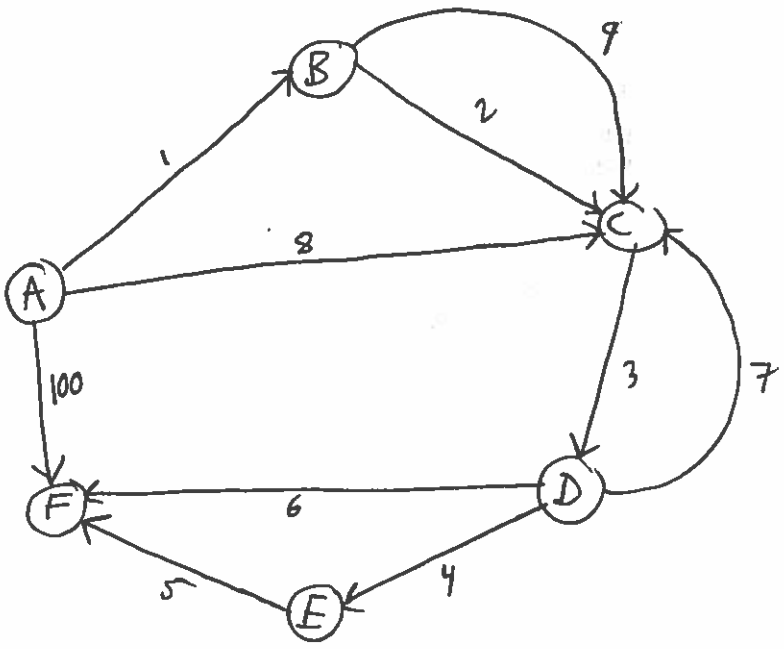
(ii)

	1	2	3	4
1	F	T	T	T
2	F	F	F	F
3	F	F	F	F
4	F	F	F	F

8. Graph Representations (continued)

(c) Draw a weighted directed **cyclic** graph with 6 nodes (A, B, C, D, E, F) that meets all the following criteria:

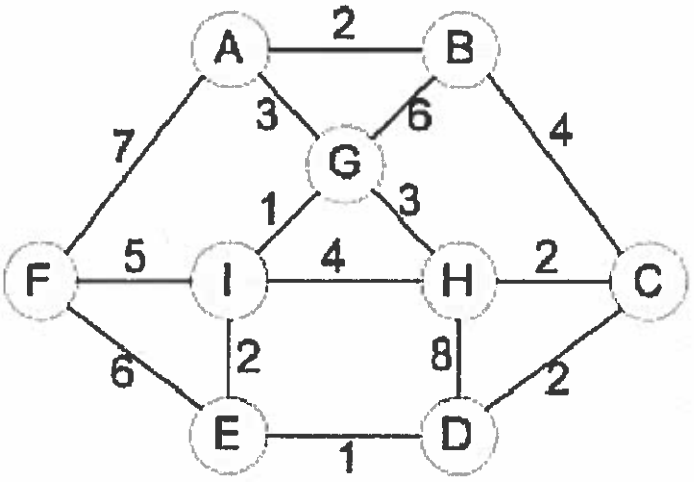
i. At least 10 total edges (not including self edges)
ii. Breadth First Search from node A to F returns a path without the lowest edge cost.
iii. Every edge weight is unique
iv. One path from A to F has a length for exactly 4 edges (not including self edges)
v. Has at least 1 cycle

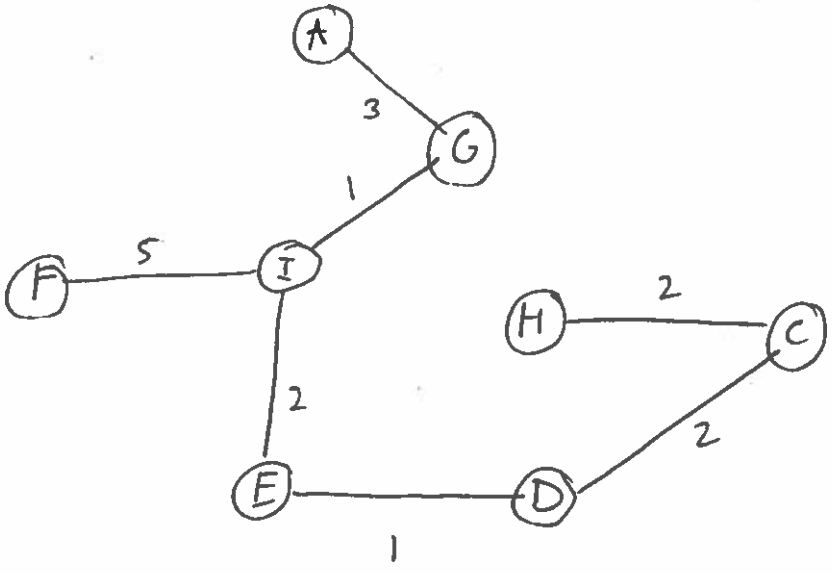


ONE POSSIBLE SOLUTION

9. MST

Given the following undirected graph, draw one minimum spanning tree of the graph. You do not need to show your work, however, showing work may help award partial credit.





10. Critical-er thinking

Given an array of integers with the condition that each integer is within 'k' indices of its correctly sorted position in the array, efficiently sort the array and return it. For example:

Given array: [1, 3, 5, 2, 4, 6]
k: 2

Ouput array: [1, 2, 3, 4, 5, 6]

In this example we see that 3 and 4 are initially 1 index away from their correct location. The numbers 2 and 5 are both 2 indices away from their proper location.

You may write java code or psuedocode. You may assume that the array being passed in is not empty or null. Credit will be based on correctness and the asymptotic runtime of your solution.

```
//assumes k < n
partial_sort(int k, int[] nums) {
    PriorityQueue pq = new MinHeap;
    for(int i=0; i <= k; i++) {
        pq.add(nums[i]);
    }
    int next = k+1;
    for(int i=0; i < nums.length; i++) {
        nums[i] = pq.removeMin();
        if(next < nums.length){
            pq.add(nums[next]);
            next++;
        }
    }
    return nums
}
```

runtime: O(n log(k))

space: O(k)