CS 61B Spring 2023

Final

Tuesday May 9, 2023

Intro: Welcome to the CS61B Final!

Your Name: _____

Your SID: _____ Login: _____

Location: _____

SID of Person to your Left: _____ Right: _____

Write the statement *"I have neither given nor received any assistance in the taking of this exam."* below.

_____

_____

Signature: _____

Tips:

- For answers which involve filling in a ◯ or ☐, **please fill in the shape completely.** If you change your response, **erase as completely as possible**. Incomplete marks may affect your score.
- ◯ indicates that only one circle should be filled in.
- ☐ indicates that more than one box may be filled in.
- You may not need to use all provided lines, but we **will not give credit for solutions that go over number of provided lines.**
- You may not use ternary operators, lambdas, streams, or multiple assignment.
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues or complex corner cases you're not sure about.
- Not all information provided in a problem may be useful, and you may not need all lines.
- Unless otherwise stated, all given code on this exam should compile. All code has been compiled and executed before printing, but in the unlikely event that we do happen to catch any bugs in the exam, we'll announce a fix. Unless we specifically give you the option, the correct answer is not 'does not compile'.

This page is intentionally left blank (except for this sentence and the page number).

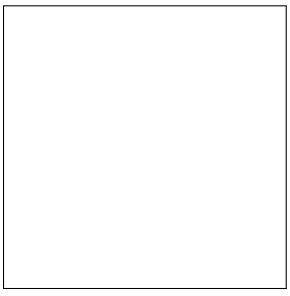
I Got A ...

(800 Points)

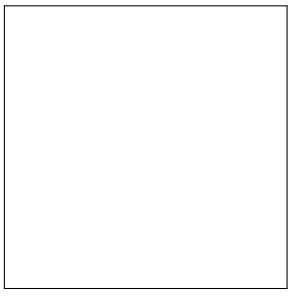
```
class A {
    public void f() {
        B me = (B) this;
        g(this);
    }
    public void g(A x) {
        System.out.println("Mom");
        x.h();
    }
    public void h() {
        System.out.println("I got a +2");
    }
}

class B extends A {
    public void g(B x) {
        System.out.println("Brother");
        x.h();
    }
    @Override
    public void h() {
        System.out.println("I got a PB!");
    }
    public static void main(String[] args) {
        // CODE HERE
    }
}
```

Write what the `main` method in class B will print if we inserted and ran the code in `// CODE HERE`. **If it errors, write CE for compile error or RE for runtime error.**

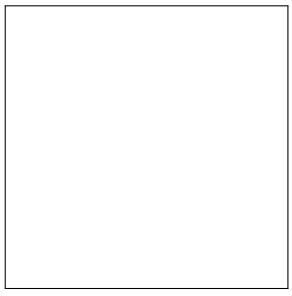
```
A kam = new A();
kam.f();
```



```
B park = new A();
park.f();
```



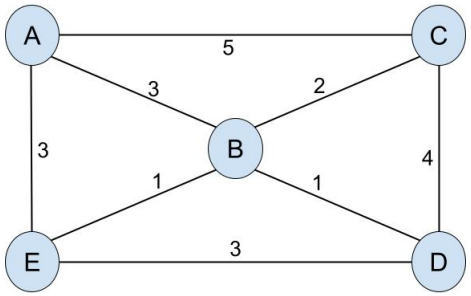
```
A mitch = new B();
mitch.f();
```



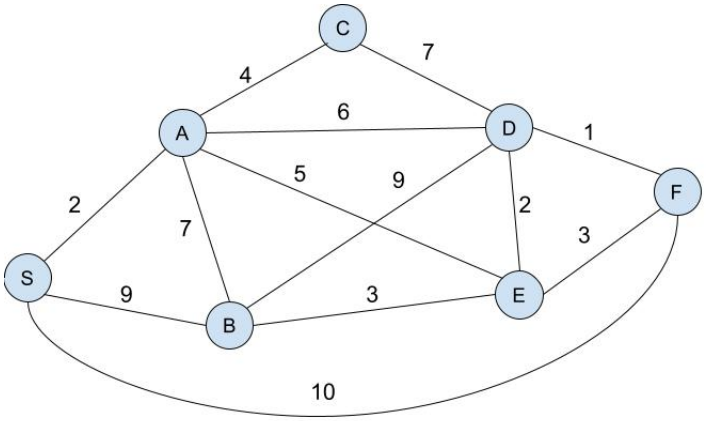
Edgy Questions

(1200 Points)

(a) ○ True ○ False Kruskal's algorithm and Prim's algorithm will produce the same MST if Prim's algorithm is run from B, even if their tie-breaking schemes are different in the graph to the right.



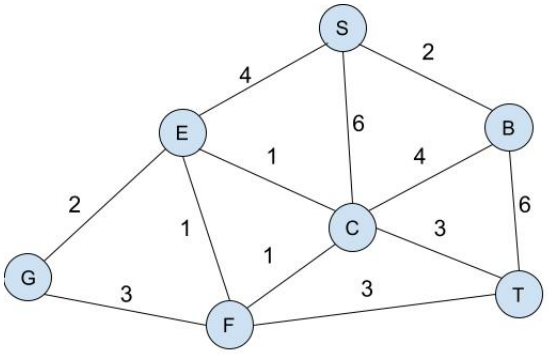
(b) Consider the graph G below.



Suppose that we have started running Dijkstra's from S on and terminated the program when the Dijkstra's finished finding the shortest path to E. What other nodes' shortest path from S have still have priority infinity?

☐ S ☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

(c) Consider the graph G with the heuristic h below.



Node	h(v, T)
S	3
B	6
C	2
E	2
F	5
G	12
T	0

Suppose we run A* from node S to node T on graph G. Break ties alphabetically. What is the order that vertices (including S and T) are visited? Please separate vertices with spaces.

(d) A binary tree is constructed of nodes that are instances of the following class:

```
public class Node {
   public int val;
   public Node left;
   public Node right;
}
```

Consider the following method:

```
public static Node mystery(Node root) {
   if (root.right == null) { return root; }
   else { return mystery(root.right); }
}
```

Which of these statements is correct about the method regardless of the contents of the tree when passed a reference to the root node of a binary tree?

☐ It returns the last node visited by an in-order traversal

☐ It returns the last node visited by a post-order traversal

☐ It returns the last node visited by a level-order traversal

☐ None of the above

Value Town

1200 Points

Mihir really wants to rate the value for each food he eats at Berkeley. He makes a Food object to save the food name and cost per meal. He then stores Foods as keys and ratings as values in a HashMap. Unfortunately, he ran into bugs in his program and scattered print statements to check the behavior.

For this problem, assume the MyHashMap is initialized with an array of length 4. Our MyHashMap uses external chaining as implemented in class to insert new items. You may assume MyHashMap resizes by doubling the size of the array after the load factor is greater than the the maximum load factor of 1.

```
1  public class Food {
2      public String name; public int price;
3      public Food(String name, int price) { this.name = name; this.price = price; }
4
5      @Override
6      public int hashCode() { return price; }
7      public static void main(String[] args) {
8          HashMap<Food, Integer> ratings = new MyHashMap<>();
9
10         Food burrito = new Food("burrito", 8);
11         ratings.put(burrito, 5);
12         ratings.put(new Food("taco", 6), 3);
13         System.out.println(ratings.getOrDefault(burrito, -1));
14         // getOrDefault: Returns the value or -1 if the map contains no mapping for the key.
15
16         burrito.price += 3;
17         System.out.println(ratings.getOrDefault(burrito, -1));
18
19         ratings.put(new Food("burger", 13), 1);
20         ratings.put(new Food("pizza", 7), 4);
21         System.out.println(ratings.getOrDefault(burrito, -1));
22
23         ratings.put(new Food("sandwich", 9), 2);
24         System.out.println(ratings.getOrDefault(burrito, -1));
25     }
26 }
```

(a) What is the output of line 13? _____

(b) What is the output of line 17? _____

(c) What is the output of line 21? _____

(d) What is the output of line 24? _____

(e) The bucket at index 0 is empty. ○True ○False

(f) The resulting map has a bucket that contains at least 2 Food objects. ○True ○False

(g) Suppose Mihir runs `ratings.put(new Food("sandwich", 9), 3);` below line 24 again. The number of elements in the map will increase. ○True ○False

Asymptotics

(2000 Points)

Give the runtime of the following `enigma` functions in terms of N. Your answers should be as simple as possible, excluding unnecessary constants like log bases and lower-order terms. Assume there is no limit on the size of an **`int`** (otherwise all run times are technically constant).

enigma1

Θ( )

```
public static void enigma1(int N) {
    for (int i = 0; i < Math.pow(2, N); i += 5) {
        for (int j = N; j >= 0; j -= 1) {
            System.out.println("Never gonna give you up");
        }
    }
}
```

enigma2

Θ( )

```
public static void enigma2(int N) {
    for (int i = N * 8; i > 1; i /= 4) {
        for (int j = 0; j < 10000; j += N) {
           System.out.println("Never gonna let you down");
        }
    }
}
```

enigma3

Θ( )

```
public static void enigma3(int N) {
    for (int i = 2; i < N; i *= 2) {
        for (int j = 0; j < N * N; j += i/2) {
           System.out.println("Never gonna turn around");
        }
    }
}
```

enigma4

Θ( )

```
public static void enigma4(int N) {
    enigma4(N, N);
}
public static void enigma4(int x, int n) {
    if (x == 1) { return; }
    else {
        for (int i = 0; i < n; i += 1) {
            enigma4(x - 1, n);
        }
    }
}
```

Now, consider the following function ben. For the questions below, assume that the pseudocode is translated properly into Java and the program compiles.

```
public static void ben(int N) {
    if (N <= 625) { return; }
    for (int i = 0; i < N; i += 1) {
        System.out.println("And desert you.");
    }
    int answer = _____________;
    for (int j = 0; j < answer; j += 1) {
        ben(N / answer);
    }
}
```

What is the runtime if we set `answer = 0` ?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

What is the runtime if we set `answer = 2` ?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

What is the runtime if we set `answer = N/2` ?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

What is the runtime if we set `answer = NlogN` ?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

What is the runtime if we set `answer = N^2` ?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

What is the runtime if we set `answer = N^N`?

○ $\Theta(\log N)$ ○ $\Theta(N)$ ○ $\Theta(N \log N)$ ○ $\Theta(N(\log N)^2)$ ○ $\Theta(N^2)$ ○ None of the above

This page is intentionally left blank (except for this sentence and the page number).

Wordle

(2400 Points)

Write a class to store a Wordle dictionary. Your solution must allow for the following functions:

`insert(String word)`: Inserts a new word of length W into the dictionary and takes $O(W)$ time. **All `word` arguments are unique, non-null, and consists of only CAPITAL letters or the empty string**.

`contains(String word)`: Returns true if the dictionary contains the word and takes $O(W)$ time. **All `word` arguments are unique, non-null, and consists of only CAPITAL letters or the empty string**.

`wordList()`: Returns a list of all words in the dictionary, in alphabetical order. This must take $O(WN)$ time, where N is the number of words in the dictionary.

`getRandomWord(Random random)`: Returns a random word from the dictionary. This must take $O(W)$ time, so calling `wordList` will exceed your runtime. All words must be equally likely to be returned by your function regardless of the words in the dictionary.

You may assume all string operations in the reference sheet and string concatenation run in $O(1)$ time.

```
class Wordle { // we will use a modified trie data structure!
    private boolean valid;
    private Wordle[] children;
    private int size;

    public Wordle() {
        valid = false;
        children = new Wordle[26];
        size = 0;
    }

    private int charToInt(char c) { return c - 'A'; } // Converts A to 0, B to 1,... Z to 25
    private char intToChar(int i) { return (char) ('A' + i); } // Converts 0 to A, 1 to B,... 25 to Z

    public void insert(String word) {
        size += 1;
        if (word.length() == 0) {

            ____________________________________________________________________________________;
            return;
        }

        int index = ____________________________________________________________________________;

        if (____________________________________________________________________________________) {

            ____________________________________________________________________________________;
        }
        ________________________________________.insert(____________________________________);
    }
```

```
public boolean contains(String word) {
    return ( (__1__) && (__2__) ) || ( (__3__) && (__4__) && (__5__) );
}
```

Put the letter that corresponds to the code to run `contains`. An answer choice may be used more than once.

(a) `this.valid`
(b) `this == null`
(c) `word.length() == 0`
(d) `this.contains(word.substring(1))`
(e) `children[charToInt(word.charAt(0))]==null`
(f) `children[charToInt(word.charAt(0))].contains(word.substring(1))`
(g) `!this.valid`
(h) `this != null`
(i) `word.length() != 0`
(j) `!this.contains(word.substring(1))`
(k) `children[charToInt(word.charAt(0))]!=null`
(l) `!children[charToInt(word.charAt(0))].contains(word.substring(1))`

1._____ 2._____ 3._____ 4._____ 5._____

```
public List<String> wordList() {
    List<String> words = new ArrayList<>();

    if (this.valid) { ____________________________________________________________________________; }
    for (int i = 0; i < 26; i += 1) {
        if (children[i] != null) {

            for (_____________________________________________________________________________) {

                ____________________________________________________________________________;
            }
        }
    }
    return words;
}

public String getRandomWord(Random random) {
    int randNum = random.nextInt(this.size);

    if (___________________________________ && ___________________________________) { return ""; }
    int letter = -1;
    while (randNum >= 0) {
        letter += 1;

        if (_____________________________________________________________________________ != null) {

            ____________________________________________________________________________;
        }
    }
    return ____________________________________________________________________________;
}
}
```

Sorta Sorting

(1400 Points)

Emily loves geese, but they always wander about. Typically, Emily needs to do some sort-esque work with the geese's weight, but comparison based sorts take at least $O(N \log N)$ time which gives her geese time to run away! We need to find a way to perform her work in linear time.

For this entire question, arr is an array of at least 100 nonnegative integers (0 is a nonnegative integer). `sort` is an comparison-based sort that sorts in ascending order in $O(N \log N)$ time **that non-destructively returns a copy of a sorted array.**

Consider the following functions:

Part A:

```
// arr is an array of at least 100 nonnegative integers
public static int foo(int[] arr) {
    int[] sortedArr = sort(arr);
    return sortedArr[9];
}
```

(i) Describe in 10 words or less what `foo` does.

(ii) Rewrite `foo` so it runs in $O(N)$ time instead of $O(N \log N)$ time.

```
public static int foo(int[] arr) {

    MinHeap<Integer> h = new MinHeap<>();

    for (int i = 0; ____________________________________; ____________________________________) {

        ____________________________________________________________________________________;
    }
    for (int i = 10; ____________________________________; ____________________________________) {


    ____________________________________________________________________________________;

        ____________________________________________________________________________________;
    }
    return ____________________________________________________________________________________;
}
```

Part B:

```
// arr is an array of least 100 nonnegative integers
public static int bar(int[] arr) {
    int[] sortedArr = sort(arr);
    int i = -1;
    for(int j = 0; j < sortedArr.length; j += 1) {
        if (sortedArr[j] - i >= 2) {
            return i + 1;
        }
        i = sortedArr[j];
    }
    return i + 1;
}
```

(i) Select the option that properly describes what `bar` does.

○ Returns the smallest nonnegative integer
○ Returns the smallest duplicate nonnegative integer
○ Returns the smallest nonnegative integer not in the list
○ Returns the smallest nonnegative integer that is not duplicated in the list

(ii) Rewrite `bar` so it runs in $O(N)$ time instead of $O(N \log N)$ time.

```
public static int bar(int[] arr) {
    boolean[] counts = new boolean[arr.length];
    for (int i = 0; i < arr.length; i += 1) {

        if (_____________________________________________________________________________) {

            _____________________________________________________________________________;
        }
    }
    for (int i = 0; i < arr.length; i += 1) {

        if (_____________________________________________________________________________) {

            _____________________________________________________________________________;
        }
    }

    return _____________________________________________________________________________;
}
```

Actually Sorting

(1200 Points)

Part 1:

(a) What is the output of Hoare's partition algorithm using the first index as a pivot?

Before Value	4	6	2	5	1	7	3	8
After Value								

(b) ○ True ○ False There exists a comparison algorithm that sorts any list of 2023 numbers in at most 2024 comparisons in the worst case.

(c) ○ True ○ False Any comparison-based sorting algorithm can be made stable by modifying the comparison operation to also consider the original index of the elements being compared.

Part 2:

(a) You are sorting a list of N distinct integers.

(i) What is the runtime of quick sort if your quick sort algorithm always picks the 10th smallest element as the pivot? $\Theta($_____$)$

(ii) What is the runtime of quick sort if your quick sort algorithm always picks the 10th percentile (10% smallest) element as the pivot? $\Theta($_____$)$

(b) Now consider a new sorting algorithm, `quickMergeSort`. The algorithm is given in pseudocode below.

```
quickMergeSort(list) {
        If list.size <= 1 return list
        Pick a pivot in constant time
        Partition around pivot
        mergeQuickSort(left partition)
        mergeQuickSort(right partition)
}
mergeQuickSort(list) {
        If list.size <= 1 return list
        Split list into two halves
        left half = quickMergeSort(left half)
        right half = quickMergeSort(right half)
        merge(left half, right half)
}
```

(i) Whats the runtime if `quickMergeSort` always picks the median as the pivot? $\Theta($_____$)$

(ii) Whats the runtime if `quickMergeSort` always picks the minimum as the pivot? $\Theta($_____$)$

Part 3:

The intermediate steps in performing various sorting algorithms on the same input list are shown. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence.

Select the algorithm it illustrates from among the following choices: LSD (least significant digit) radix sort, MSD (most significant digit) radix sort, insertion sort, selection sort, and heap sort.

(a) ○ LSD Radix ○ MSD Radix ○ Insertion ○ Selection ○ Heap

```
1430, 3292, 7684, 9002, 1001, 595, 4243, 1338, 4393, 130, 193
1430, 130, 1001, 3292, 9002, 4243, 4393, 193, 7684, 595, 1338
1001, 9002, 1430, 130, 1338, 4243, 7684, 3292, 4393, 193, 595
```

(b) ○ LSD Radix ○ MSD Radix ○ Insertion ○ Selection ○ Heap

```
1338, 1430, 3292, 7684, 193, 595, 4243, 9002, 4393, 130, 1001
193, 1338, 1430, 3292, 7684, 595, 4243, 9002, 4393, 130, 1001
193, 595, 1338, 1430, 3292, 7684, 4243, 9002, 4393, 130, 1001
```

(c) ○ LSD Radix ○ MSD Radix ○ Insertion ○ Selection ○ Heap

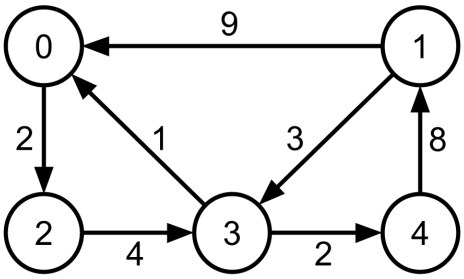
```
1430, 3292, 7684, 9002, 1001, 595, 4243, 1338, 4393, 130, 193
9002, 7684, 4393, 4243, 3292, 1001, 595, 193, 1338, 1430, 130
130, 4393, 4243, 3292, 1001, 595, 193, 1338, 1430, 7684, 9002
```

The Javaugean Stables

(2600 Points)

Part A:

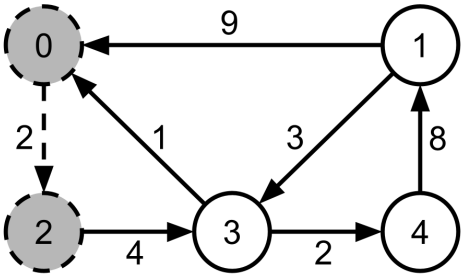
8 months after successfully defeating the Lambdanean Hydra, Heracles is tasked with another labor: cleaning the Javaugean Stables.



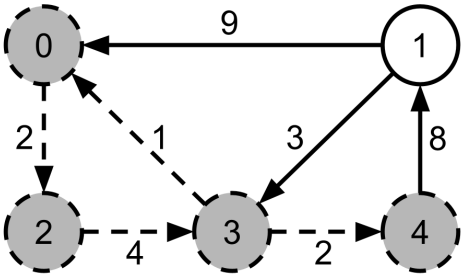
The Javaugean Stables are represented as a directed graph with positive edge weights, with nodes denoting individual stables, and directed edges representing paths between them. You may assume that the graph is fully connected, and that the stables can be fully flooded at some depth.

The Alpheus river runs through stable 0, and Heracles plans to divert its waters through all the stables in order to clean them. Water can flow along paths as long as its depth is at least the weight of the edge. **Help Heracles determine the minimum depth of water needed in order to flood every stable with water starting from stable 0.**

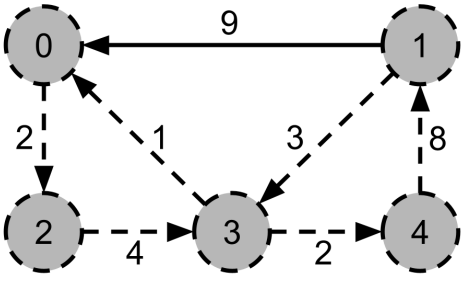
Your code must run in $O(E \log E)$ time, and your code's runtime may not depend on the weights of the edges.



In the above example, if Heracles sets the depth of water to 2, water can flow along the edge 0 to 2, so stable 2 gets flooded.



If Heracles sets the depth of water to 4, stable 3 and stable 4 also get flooded.

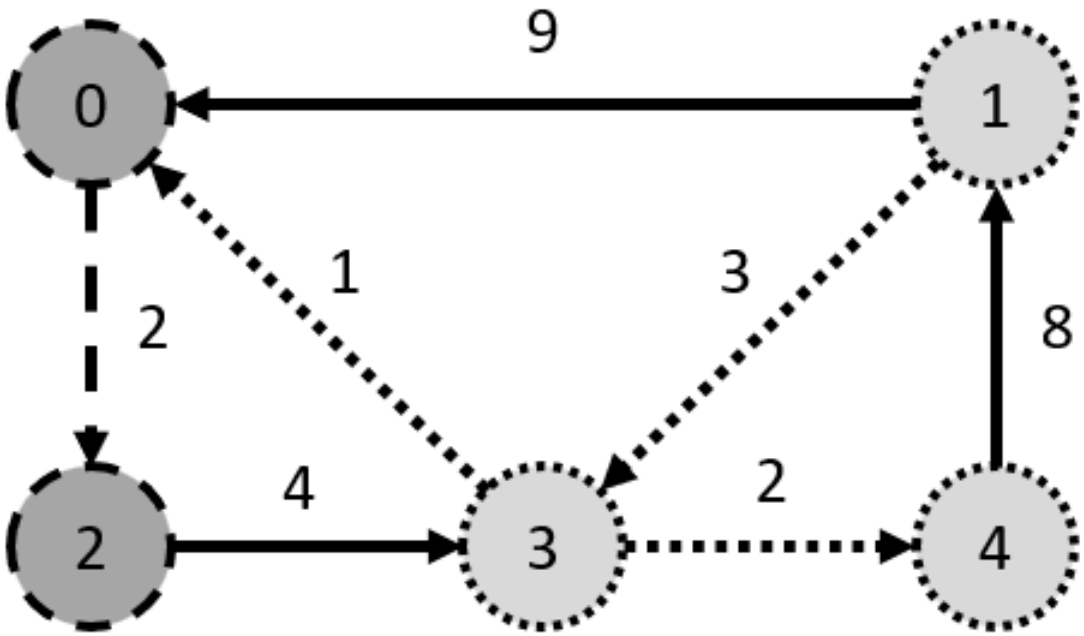


In order to flood all stables, Heracles needs to set the depth of water to at least 8.

```
public class StablesQuestion {
    public static int Stables(Graph g) {
        PriorityQueue<GraphEdge> h = new PriorityQueue<>();
        Queue<Integer> q = new Queue<>();

        boolean[] visited = new boolean[_____________________________________________];
        int depth = 0;

        q.insert(_____________________________________________________________________);

        while (_____________________________________ || _____________________________________)) {
            if (q.size() > 0) {

                int n = _____________________________________________________________;

                visited[n] = _____________________________________________________________;

                for (_____________________________________________________________) {

                    _____________________________________________________________;
                }
            } else {

                GraphEdge smallestedge = _____________________________________________;

                if (!visited[_____________________________________________________]) {

                    _____________________________________________________________;

                    depth = _____________________________________________________;
                }
            }
        }
        return depth;
    }
}
```

Part B:

Heracles realizes that the Peneus river flows through stable 1, and can also be used to clean stables. By using both rivers, Heracles hopes to reduce the depth he needs to dig. Write an algorithm to determine the new minimum depth.



In the above example, Heracles only needs to dig to a depth of 3 to flood all stables now. Stables 0 and 2 are flooded from stable 0 (Alpheus river) via the dashed lines, and stables 1, 3, and 4 are flooded from stable 1 (Peneus river) via the dotted lines.

```
// Hint: Find a way to change the graph so you can use your answer to part A.
public static int TwoRiverStables(Graph g) {
    Graph newGraph = new Graph(g);

    ____________________________________________________________________;
    return Stables(newGraph);
}
```

Nothing on this page is worth any points.

61B

(0 Points)

Please pick a integer between 1 to 10000, inclusive on both ends, that you believe will be the median of all the integers guessed for this question.

Feedback

(0 Points)

Leave any feedback, comments, concerns, or drawings below!

String Class API

```
public class String {
    /** Returns the length of this string. */
    public int length() { ... }

    /** Returns the character at the specified index (position) */
    public char charAt(int index) { ... }

    /** Returns a new string that is a substring of this string starting at the beginIndex
     *  For example: "asdfjkl".substring(4) evaluates to "jkl" */
    public String substring(int beginIndex) { ... }
}
```

Random Class API

```
public class Random {
    /** Creates a new random number generator. */
    public Random() { ... }

    /** Creates a new random number generator using a seed. If two instances of Random are created
     *  with the same seed, they will generate and return identical sequences of numbers. Runs in
     *  Θ(1) time. */
    public Random(long seed) { ... }

    /** Returns the next pseudorandom int from this random number generator's sequence. Runs in
     *  Θ(1) time. */
    public int nextInt() { ... }

    /** Returns a pseudorandom int value between 0 (inclusive) and the specified value (exclusive),
     *  drawn from this random number generator's sequence. Runs in Θ(1) time. */
    public int nextInt(int bound) { ... }
}
```

MinHeap Class API

```
public class MinHeap<E> {
    /** Create a MinHeap<E>. Runs in Θ(1) time. */
    public MinHeap() { ... }

    /** Inserts an element into the heap. Runs in Θ(log(N)) time. */
    public void insert(E element) { ... }

    /** Deletes and returns the minimum element in the heap. If empty, return null.
     *  Runs in Θ(log(N)) time. */
    public E deleteMin() { ... }

    /** Determines whether the heap is empty or not. Runs in Θ(1) time. */
    public boolean isEmpty() { ... }
```

```
    /** Returns the size of the heap. Runs in Θ(1) time. */
    public int size() { ... }

    /** Returns, but does not delete, the minimum element in the heap. This method is
     *  non-destructive. Runs in Θ(1) time. */
    public E findMin() { ... }
}
```

Graph Class API

```
public class Graph {
    /** Creates a new graph with V nodes (numbered from 0 to V-1). Runs in Θ(V) time. */
    public Graph(int V) { ... }

    /** Creates a deep copy of graph g. Runs in Θ(|E|) time */
    public Graph(Graph g) { ... }

    /** Adds an edge to the graph. If an edge already exists between g.from and g.to, it is replaced.
     Runs in Θ(1) time. */
    public void addEdge(GraphEdge g) { ... }

    /** Returns a set of all edges leaving the given node. Runs in Θ(|V|) time. */
    public Set<GraphEdge> neighbors(int from) { ... }

    /** Returns the number of vertices. Runs in Θ(1) time. */
    public int size() { ... }
}
```

GraphEdge Class API

```
public class GraphEdge implements Comparable {
    public int from;
    public int to;
    public int weight;
    /** Creates a directed edge of the given weight between the given nodes. Runs in Θ(1) time. */
    public GraphEdge(int from, int to, int weight) { ... }

    /** Compares two edges according to their weights. For example, an edge with weight 4 will be
    less than an edge with weight 9. Runs in Θ(1) time. */
    public int compareTo(GraphEdge e) { ... }
}
```

Queue Class API

```
public class Queue<E> {
   /** Create a queue*/
   public Queue() { ... }
```

```
    /** Create a queue, then inserts all elements of lst one at a time into the queue. Runs in Θ(N)
     time.*/
    public Queue(List<E> lst) { ... }

    /** Inserts the specified element into this queue returning true on success. Runs in Θ(1) time.*/
    public boolean add(E e) { ... }

    /** Retrieves, but does not remove, the element at the head of the queue, or returns null if this
     queue is empty. Runs in Θ(1) time.*/
    public E peek() { ... }

    /** Retrieves and removes the element at the head of the queue. Runs in Θ(1) time.*/
    public E remove() { ... }

    /** Returns the size of the queue. Runs in Θ(1) time.*/
    public int size() { ... }

}
```

ArrayList Class API

```
public class ArrayList<E> implements List<E> {
    /** Appends the specified element to the end of this list. Runs in Θ(1) time. */
    public boolean add(E e) { ... }

    /** Returns the element at the specified position in this list. */
    public E get(int index) { ... }

    /** Replaces the element at the specified position in this list with the specified element. */
    public E set(int index, E element) { ... }

    /** Returns the number of elements in this list. */
    public int size() { ... }
}
```

Math Class API

```
public class Math {
    /** Returns the value of the first argument raised to the power of the second argument. */
    public static double pow(double a, double b) { ... }

    /** Returns the smaller of two int values. */
    public static int min(int a, int b) { ... }

    /** Returns the greater of two int values. */
    public static int max(int a, int b) { ... }
}
```

Edgy Questions, part B: Skip this question

Queue Class API: insert refers to the same function as add. Either method name will be accepted.

Actually Sorting Part 1 (a): Run Hoare Partitioning using the **0th index** (4) as the pivot.

Javaugean Stables Part A: PriorityQueue should be replaced with MinHeap