

Name:   
ID:

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

---

*Advice 1:* For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

*Advice 2:* Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

**Honor code:** On my honor as a University of Colorado at Boulder student, I have neither given nor sought unauthorized assistance in this work

**Initials**

**Date**

If you violate the CU **Honor Code**, you will receive a 0.

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

---

**Instructions for submitting your solution:**

- The solutions **should be typed**, we cannot accept hand-written solutions. Here's a short intro to [Latex](#).
- In this homework we denote the asymptomatic *Big-O* notation by  $\mathcal{O}$  and *Small-O* notation is represented as  $o$ .
- We recommend using online Latex editor [Overleaf](#). Download the **.tex** file from Canvas and upload it on overleaf to edit.
- You should submit your work through [Gradescope](#) only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Canvas if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

1. (5 pts) You are given a list of jobs with start and end times. Assume that only one job can be executed at a time on a processor. Your task is to find out the minimum number of processors required so that the jobs are executed as soon as they arrive (no waiting time for a process). The input to your algorithm will be a list of tuples indicating the start and end times of the respective jobs. The output should be a number indicating the minimum number of processors required. Your algorithm should have a runtime of  $\mathcal{O}(n \log(n))$ , where  $n$  is the number of jobs. Assume you do not have access to any auxiliary functions.

**Example**

**Input:** (2, 10), (9, 11), (15, 18), (3, 4), (17, 19), (5, 13)

**Output:** 3

**Explanation:** Between the interval (9, 10) it can be seen that there will be 3 { (2, 10), (9, 11), (5, 13) } jobs that need to be executed simultaneously. Thus at the very least 3 processors are required.

- (a) (4 pts) Write down well commented pseudo-code or paste real code to solve the above problem.

```
def intervalmerge(arr):  
    # Sort the list of tuples in ascending order  
    sorted(arr, key = lambda x: x[0])  
  
    # 'm' counter is the ending point within array  
    m = 0  
  
    # 'count' will count how many times we merge 2 non  
    # overlapping schedules  
    count = 0  
  
    # function that traverses through sorted array, and  
    # count the available schedules that can be merged  
    for i in range(len(arr)):  
        # create new array 'k' temp, which will use to compare, and  
        # temp store the current element within the array  
        k = arr[i]  
        # if the element we are comparing is greater than previous value  
        if k[0] > m:  
            # case for the initial value  
            if i != 0:  
                count += 1  
            # update m as the next value within array  
            m = k[1]  
        else:  
            if k[1] >= m:  
                m = k[1]  
                count += 1  
  
    # 'm' value gives the last element of  
    # that particular interval  
    print("The number of merged intervals: ", count)  
  
# Driver code  
arr = [(2, 10), (9, 11), (15, 18), (3, 4), (17, 19), (5, 13)]  
intervalmerge(arr)
```

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (b) (1 pts) Explain your algorithms runtime and space complexity by analyzing your code. For example, stating that a sorting algorithm runs in  $\mathcal{O}(n \log(n))$  without any justification is insufficient.

Runtime:  $\mathcal{O}(n \log n)$

Space Complexity:  $\mathcal{O}(1)$

- (1) Sort all intervals in decreasing order based on their start time.

**Time Complexity:**  $\mathcal{O}(n \log n)$

- (2) Traverse through the array of sorted tuples and do:

if the element we are comparing is greater than previous value, and is not the first interval within the array, merge and increment count

**Space Complexity:**  $\mathcal{O}(n)$

Analyzing the algorithm we see that our time complexity is there for  $\mathbf{O}(n \log n)$ . The space complexity is  $\mathbf{O}(n)$  because we need the extra space for our array  $m$  when comparing the previous and current element in the array.

Name: Sasha Farhat

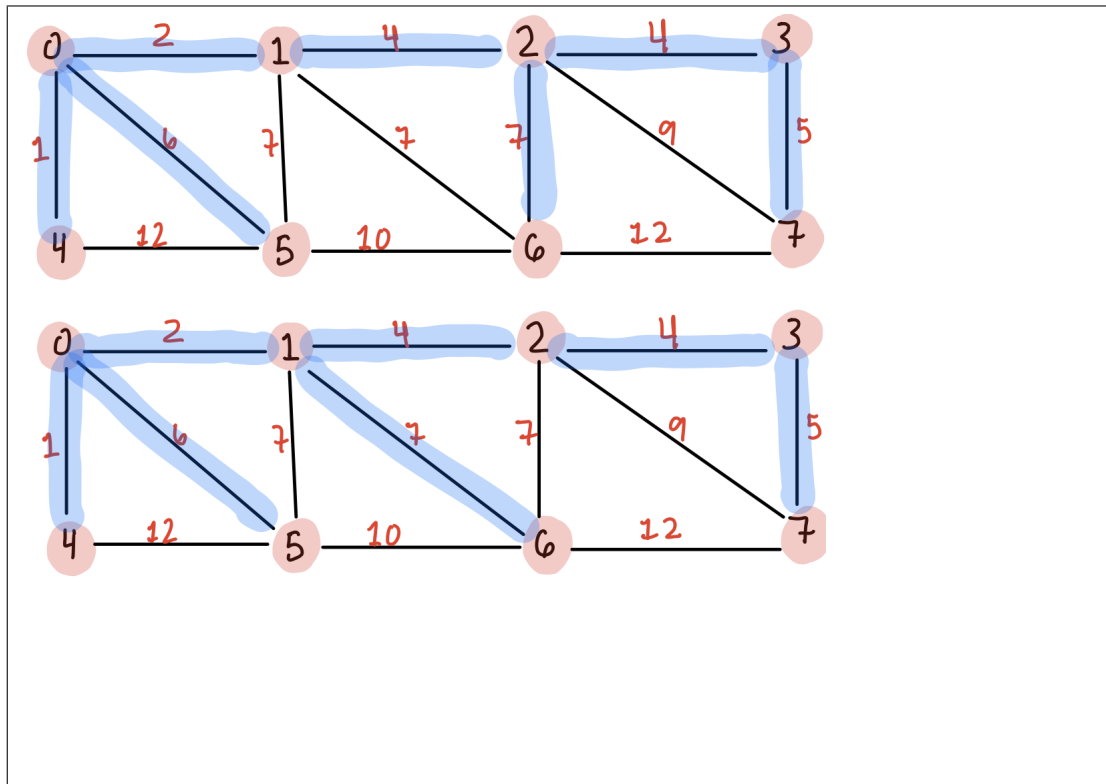
ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

2. (5 pts) You are designing a network, to connect several offices in different locations. You can contact your network cable provider about the cost of connecting any pair of offices. Your task is to design an algorithm, so that all offices are connected and the cost associated with building the network is minimized.
- The input to your algorithm is a Graph in the form of an adjacency list or matrix. The nodes in this graph represent the offices and edge weights represent the cost associated with connecting a pair of offices. The output of the algorithm will be a list of office pairs such that the total connection cost is minimized.

- (a) (2 pts) Provide an example with at least 3 offices and 3 connections. Your example must include at least 2 unique solutions.



Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (b) (3 pts) Provide well commented pseudo code or actual code to solve the above problem.

```
#class for the given graph
class Graph:
    def KruskalMST(self):

        #array that will store the resultant mst
        mstresult = []

        e = 0 # An index variable, used for result[]

        #we first sort all the edges ascending order by weight
        self.graph = sorted(self.graph, key=lambda item: item[2])

        parent = []
        rank = []

        # Create V subsets with single elements
        for node in range(self.V):
            parent.append(node)
            rank.append(0)

        #'i' is our index variable for sorted edges
        i = 0

        #'e' is index for number of edges
        e = 0

        #while the current number of edges is less than graphed edges
        while e < self.V - 1 :
            #pick the first element, (since the graph is sorted). Traverse through elements
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)

            # check for cycle, if it doesnt then add to mst
            if x != y:
                e = e + 1
                mstresult.append([u, v, w])
                self.union(parent, rank, x, y)
            # Else discard the edge

        totw = 0
        # print the contents of result[] to display the built MST
        print("MST")
        for u, v, weight in mstresult:
            totw = totw + weight
            print ("[" , u , " , " , v , " ] =" , weight)

        print("Total minimized connection cost: " , totw)

    def __init__(self, vertices):
        self.graph = []
        self.V = vertices
```

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

```
# function to add an edge to graph
def addEdge(self,u,v,w):
    self.graph.append([u,v,w])

# path comparison to find the element i
def find(self, parent, i):
    if parent[i] == i:
        return i
    return self.find(parent, parent[i])

#union function for x and y
def union(self, parent, rank, x, y):
    xroot = self.find(parent, x)
    yroot = self.find(parent, y)

    # Attach smaller rank tree under root of higher rank tree
    if rank[xroot] < rank[yroot]:
        parent[xroot] = yroot
    elif rank[xroot] > rank[yroot]:
        parent[yroot] = xroot

    # if ranks are the same, increment its rank by one
    else :
        parent[yroot] = xroot
        rank[xroot] += 1

# The main function to construct MST using Kruskal's algorithm

# Driver code
#weighted undirected graph (picture drawn in 2a)
g = Graph(8)
g.addEdge(0, 1, 2)
g.addEdge(0, 5, 6)
g.addEdge(0, 4, 1)
g.addEdge(1, 5, 7)
g.addEdge(1, 2, 4)
g.addEdge(1, 6, 7)
g.addEdge(2, 6, 7)
g.addEdge(2, 3, 4)
g.addEdge(2, 7, 9)
g.addEdge(3, 7, 5)
g.addEdge(6, 7, 12)
g.addEdge(5, 6, 10)
g.addEdge(4, 5, 12)

g.KruskalMST()
```

```
MST
[ 0 , 4 ] = 1
[ 0 , 1 ] = 2
[ 1 , 2 ] = 4
[ 2 , 3 ] = 4
[ 3 , 7 ] = 5
[ 0 , 5 ] = 6
[ 1 , 6 ] = 7
Total minimized connection cost: 29
```

Name: Sasha Farhat

ID: 105887541

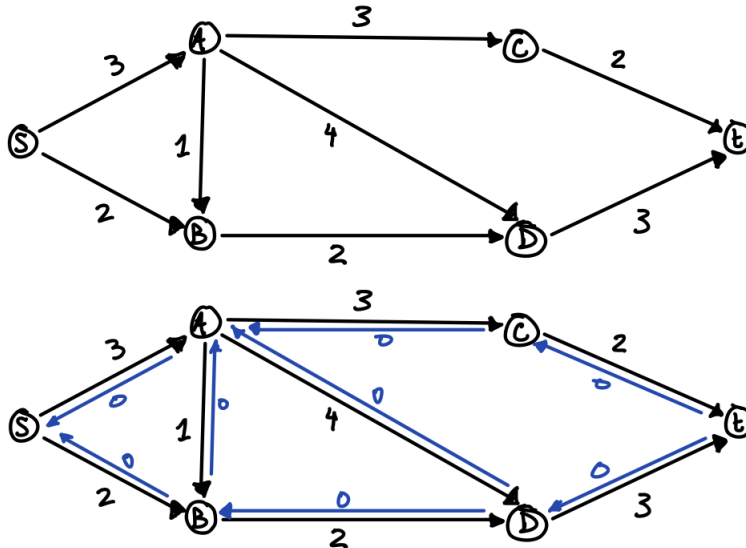
**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

3. (10 pts) Consider the graph  $G = (V, E)$  with edge capacities  $c(e)$ ,  $\forall e \in E$ , the edge capacity represents the maximum amount of flow that can pass through the edge. In addition to edge capacities, the above graph has vertex capacities  $c(v)$ ,  $\forall v \in V - \{s, t\}$  (The source and the sink vertex do not have vertex capacities). Each vertex capacity represents the maximum amount of flow that can pass through the vertex.

- (a) (4 pts) Given a source vertex  $s \in V$  and a sink  $t \in V$ . How will you modify the given graph  $G$ , so that you can find the max-flow from  $s$  to  $t$  with a known algorithm? Also, show an example input and modified graph with at least 5 vertices. This must be a drawing!

We will modify the graph in a way that the algorithm can undo its decisions by sending flow back in the opposite direction. We create a residual graph with maximum flow.





Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

(b) (4 pts) Provide an algorithm to find the max-flow in the above graph from  $s$  to  $t$ .

```
#class graph for matrix representation
class Graph:

    def __init__(self,graph):
        #xreate residual graph
        self.graph = graph
        self. ROW = len(graph)

    # using BFS we will check if there is a path from source to sink,
    # within our residual graph G
    def BFS(self,s, t, parent):

        # create boolean array (visited) and
        # initially mark all the vertices as false or unvisited
        visited = [False] * (len(self.graph))

        # this will be our queue for bfs
        queue = []

        # Mark the given node as true
        # or visited and enqueue it
        queue.append(s)
        visited[s] = True

        while queue:

            # Dequeue vertex from queue
            s = queue.pop(0)

            # enqueue through all the vertices within the neighboring visited node s
            # using our boolean array we traverse through the node,weight
            for i,w in enumerate(self.graph[s]):
                if visited[i] == False and w > 0:
                    #only count the vertices that are in range of k
                    queue.append(i)
                    visited[i] = True
                    #update parent node to current visited
                    parent[i] = s
            return True if visited[t] else False
```

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

```
# Implement Ford Fulkerson algorithm to find max flow in graph from source 's' to sink 't'
def FordFulkerson(self, source, sink):

    parent = [False]*(self.ROW)

    #our max flow is initially 0
    mflow = 0

    # while there is an augmented path within the graph
    while self.BFS(source, sink, parent) :

        # set arbitrarily large value for path
        path = 10000

        #set variable for sink to update array
        s = sink

        #while the sink is NOT the source
        while(s != source):
            #find augmented path using BFS
            path = min (path, self.graph[parent[s]][s])
            s = parent[s]

        # update residual capacities of the edges and reverse edges along the path
        v = sink
        while(v != source):
            u = parent[v]
            #decrease capacity u -> v
            self.graph[u][v] -= path
            #increase capacity v -> u
            self.graph[v][u] += path
            #update node
            v = parent[v]

        #increase maxflow
        mflow += path

    #return maxflow
    return mflow

# graph created based on my answer03 (a)
graph = [[0, 3, 2, 0, 0, 0],
          [0, 0, 1, 3, 4, 0],
          [0, 0, 0, 0, 2, 0],
          [0, 0, 0, 0, 0, 2],
          [0, 0, 0, 0, 0, 3],
          [0, 0, 0, 0, 0, 0]]

g = Graph(graph)

source = 0; sink = 5

maxflow = g.FordFulkerson(source, sink)
print ("Maximum flow is: ", maxflow)

Maximum flow is: 5

...Program finished with exit code 0
Press ENTER to exit console.
```

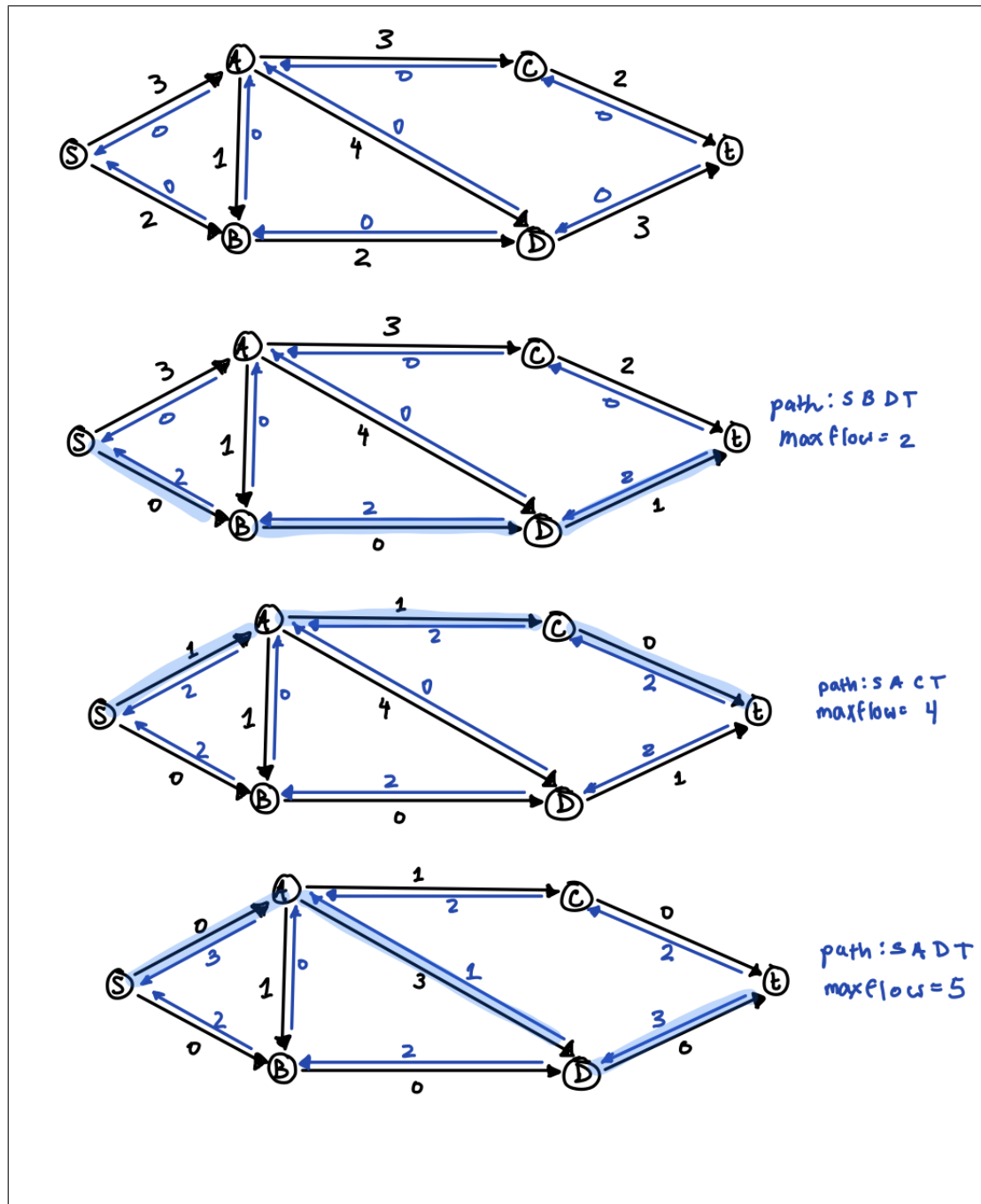
Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

- (c) (2 pts) Find the max-flow in your example graph by stepping through your algorithm. Provide a new graph image each time the flow through an edge is altered.



Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

4. (10 pts) There are  $n$  rooms numbered from  $\{1, 2, \dots, n\}$  in Williams Village. Each of the rooms has a teleportation device with a value  $v_i \ \forall i \in \{1, 2, \dots, n\}$ . The teleportation device placed in the room  $i$  with value  $v_i$  teleports to a room numbered  $(v_i \% n) + 1$ .

A room in Williams Village is said to be beautiful if you can get back to the room you started using the teleportation devices. The task is to count the number of beautiful rooms in Williams Village.

The input to your algorithm is the list of values associated with the teleportation device placed in the rooms. The output should be a number indicating the number of beautiful rooms.

**Example 1**

**Input:** [2, 3, 4, 5]

**Output:** 4

**Explanation:** All the 4 rooms that are beautiful and the corresponding paths to get back to them are as follows.

- **Room 1** The teleportation path for Room1 is Room1, Room3, and back to Room1
- **Room 2** The teleportation path for Room2 is Room2, Room4, and back to Room2
- **Room 3** The teleportation path for Room3 is Room3, Room1, and back to Room3
- **Room 4** The teleportation path for Room4 is Room4, Room2, and back to Room4

**Example 2**

**Input:** [1, 2, 3, 6]

**Output:** 2

**Explanation:** It can be seen that only Room3 and Room4 are beautiful.

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (a) (2 pts) Give a high-level explanation of how you plan to solve this problem. (How would you explain your potential solution to a younger sibling or someone who has never taken a computer science class?) Minimum 1 paragraph.

I would approach the problem in a mathematical approach. In the problem we are given the equation  $(v_i \% n) + 1$  where  $v_i = arr[i]$ . I know that this equation helps to find the next room to teleport to. So i set an unknown variable to the equation to solve,  $t = (v_i \% n) + 1$ . Next we want to see if the room we teleported to can be teleported back to the original. Using the same equation we set another unknown variable and solve for it,  $k = (v_t \% n) + 1$ . Using the two equations we can now compare the room i to k. If  $i == k$  this means that the room is we can get back to the room we started at using the teleported device thus we increment count for beautiful rooms.

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

- (b) (5 pts) Provide well commented pseudo code or actual code to solve the above problem.

```
def beautifulRooms(arr,n):

    #counter for beautiful rooms
    count = 0
    #traverse through the array
    for i in range(n+1):
        #Calculate the teleport room
        t = (arr[i-1] % n)+ 1
        #print("t: ",i,"()", t)

        #calculate if we can jump back from teleport room
        k = (arr[t-1] % n) +1
        #print("k: ",i, (), k )

        #If we are able to teleport back and and
        # if the room teleported to is not the intial room
        if(i == k and i != t):
            #Increment count
            count += 1

    return count

#driver code

arr1 = [2,3,4,5]
arr2 = [1,2,3,6]
n1 = len(arr1)
n2 = len(arr2)
beautyrooms1 = beautifulRooms(arr1,n1)
beautyrooms2 = beautifulRooms(arr2,n2)
print (arr1, "The number of beautiful rooms: ", beautyrooms1 )
print (arr2,"The number of beautiful rooms: ", beautyrooms2 )

[2, 3, 4, 5] The number of beautiful rooms: 4
[1, 2, 3, 6] The number of beautiful rooms: 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

- (c) (3 pts) Give an example input with at least 5 rooms and structure of how your algorithm explores the rooms. This must be an image! Your input can **not** be an example already given.

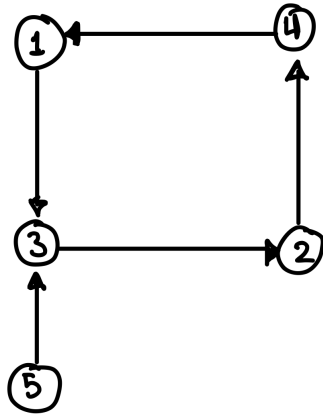
Given the array  $[2, 3, 5, 6, 7]$ , my algorithm returns 4 rooms as beautiful.  
My algorithm will calculate the first room and check if the value of room teleported to can be teleported back.

$R1 : (2\%5) + 1 = 3$  and  $R3 : (5\%5) + 1 = 1$

$R2 : (2\%5) + 1 = 4$  and  $R4 : (6\%5) + 1 = 2$

$R5 : (7\%5) + 1 = 3$

The algorithm thus explores the room as such:



Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

5. (10 pts) You are at your home in Boulder enjoying the summer and your friend challenges you to ride your bike for at least  $s$  miles starting from your home and ending at any of the scenic outlooks. You have a map of Boulder that has information about roads connecting various view points and your house. The task is to determine if there is a sequence of scenic outlooks that you can travel to starting from your house such that you have travelled at least  $s$  miles. You can not travel on the same edge more than 1 time and once you visit an outlook you cannot travel to it again. The input to your problem is a graph  $G = (V, E)$  in the form of an adjacency list or adjacency matrix and the source vertex  $s$  corresponding to your house. Your algorithm should output a boolean value indicating if it's possible to complete the challenge of riding at least  $s$  miles starting from your house.
- (a) (2 pts) Give a high-level explanation of how you plan to solve this problem. (How would you explain your potential solution to a younger sibling or someone who has never taken a computer science class?) Minimum 1 paragraph.

We model the problem as an undirected graph such that the scenic outlooks are the graph's vertices, the the edge weights on the graph are the the path distance. Since our goal is to complete a certain amount of miles such that weight  $w \geq s$ , We use bfs to to find the maximum cost of path from source. If there is a path such that the weight  $w$  of the graph is greater than  $s$ , then we return true. If we find that the maximum path in the graph is less than  $s$  then we return false.



Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (b) (6 pts) Provide well commented pseudo code or actual code to solve the above problem.

```
def boulderBycycleChallenge(g, src, w):

    #initialize k and maxweight
    maxweight = 0

    # this is the queue we use for bfs
    q = deque()

    # add source vertex to set and push it into the queue
    vertices = set([0])

    #this is our initial starting point
    q.append((src, 0, vertices))

    # while queue 'q' is not empty
    while q:

        # pop the first node from queue
        v, wpath, vertices = q.popleft()

        # if the weight of the path is greater than our current weight then update the maxweight
        if wpath > maxweight:
            maxweight = max(maxweight, wpath)

        # we check for the adjacent vertexes of the edge
        for edge in self.graph[v]:

            #we want to check that the path chosen does not create a cycle
            if visited[edge] == False:

                # add current node into the path
                s = set(vertices)
                s.add(edge.dest)
                #add the vertice to the queue
                q.append((edge.dest, wpath + edge.weight, s))

    #return boolean value comparing the maximum path weight found denoted (maxweight) and given s miles denoted (w)
    if maxweight >= w:
        return True
    if maxweight < w:
        return False
```

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (c) (2 pts) Explain your algorithms run time and space complexity by analyzing your code. For example, stating that a sorting algorithm runs in  $\mathcal{O}(n \log(n))$  without any justification is insufficient.

As there are  $V$  vertices and  $E$  edges in the graph, the time complexity for this representation is given by  $\mathcal{O}(|V| + |E|)$ . The time complexity of my code is such because it uses BFS to find the longest path within the adjacency list graph. The Space complexity of my pseudo code is  $\mathcal{O}(|V|)$ . Since my pseudo code uses bfs to queue and dequeue from the stack. Our space complexity is depended on the graph vertices and thus has complexity of  $\mathcal{O}(|V|)$ .

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

6. (20 pts) There are an even number of items arranged along a line. Each item has a value  $v_i$  associated with it. You are competing against an opponent to collect items such that total value of your items collected is maximized. At each round of the game, one player is allowed to pick an item from either end of the line.

Determine the maximum total value that can be obtained, if you are allowed to play first assuming that the opponent is equally clever. The input to your algorithm will be a list of values. The output should be a number indicating the maximum amount of value that you can obtain.

**Example**

**Input:** 22, 43, 10, 20

**Output:** 63

**Explanation:** As the first player I would first pick 20 from the right end. The opponent would then pick 22. Out of the two elements 43, 10 remaining, I can pick 43. Thus  $20+43=63$  is the maximum total value that can be obtained.

**Note:** Observe in the above example that the greedy choice of picking the max valued item does not work. If I had picked 22 instead of 20 in the first round, the maximum value that I could have obtained is 32.

- (a) (5 pts) State the base case and recursive relation that can be used to solve the above problem using dynamic programming.

**Base Case 01:**  $T(0) = 0$

**Base Case 02:**  $T(1) = c_1$

note:  $c_i$  equivalent to  $v_i$

**Recursive Relation:**  $T(n) = \max(c_n + T(n-2), T(n-1))$  for  $n \geq 1$

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

- (b) (5 pts) Provide an example input consisting of at least 5 items. Show how the dynamic programming data structure used to store previous computations is built based on the recursive relation.

c	0	1	2	3	4	5	6
arr	0	12	8	14	15	13	11

T[0]	0
T[1]	$c_1 = 12$
T[2]	$\max(c_2 + T[0], T[1]) = \max(8 + 0, 12) = 12$
T[3]	$\max(c_3 + T[1], T[2]) = \max(14 + 12, 12) = 26$
T[4]	$\max(c_4 + T[2], T[3]) = \max(15 + 12, 26) = 27$
T[5]	$\max(c_5 + T[3], T[4]) = \max(13 + 26, 27) = 39$
T[6]	$\max(c_6 + T[4], T[5]) = \max(11 + 27, 39) = 39$

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

- (c) (8 pts) Write down well commented pseudo-code or paste real code to solve the above problem using Dynamic Programming or Memoization based on the above recurrence relation.

```
def maxValueGame(arr, i, j, maxval):
    # base case 01
    if j == i:
        return max(arr[0], arr[1])

    # base case 02
    if j == i + 1:
        return max(arr[i], arr[j])

    # we maximize our decisions and minimize the opponents
    if maxval[i][j] == 0:
        # 'start' represents the maximum value from picking from the start of the arr
        start = arr[i] + min(maxValueGame(arr, i + 2, j, maxval), maxValueGame(arr, i + 1, j - 1, maxval))

        # 'end' represents the maximum value from picking at the end of the arr
        end = arr[j] + min(maxValueGame(arr, i + 1, j - 1, maxval), maxValueGame(arr, i, j - 2, maxval))

        # the maximum of the two choices is the optimal solution
        maxval[i][j] = max(start, end)
    return maxval[i][j]

# driver Code
arr = [22, 43, 10, 20]
n = len(arr) - 1
maxval = [[0 for x in range(len(arr))] for y in range(len(arr))]
value = maxValueGame(arr, 0, n, maxval)
print("Maximum value obtained: ", value)

Maximum value obtained: 63

...Program finished with exit code 0
Press ENTER to exit console.
```

Name: Sasha Farhat

ID: 105887541

CSCI 3104, Algorithms  
Final Exam Summer 2020 (60 points)

Escobedo & Jahagirdar  
Summer 2020, CU-Boulder

- (d) (2 pts) Discuss the space and runtime complexity of the code, providing necessary justification.

**Time complexity:**  $\mathcal{O}(n^2)$

**Space Complexity:**  $\mathcal{O}(n^2)$

Using memoization and dynamic programming we solve algorithm has a time and space complexity of  $\mathcal{O}(n^2)$ . Here we solve for subproblems and instead of resolving, our solution is memoized. In my code i create an extra empty array to store my previously solved recursed values  $T(n)$ . There is  $\mathcal{O}(n^2)$  subproblems because there is at most  $(i, j)$  indices such that  $1 \leq i \leq j \leq n$ . Therefore, computing a subproblem is  $\mathcal{O}(1)$  since we use memoization to compute the previous value instead of having to re-solve recurrence every time. This therefore gives a  $\mathcal{O}(n^2)$  instead of  $\mathcal{O}(nk)$  exponential time complexity.

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

---

**7. Extra Credit (3 pts) will only be considered if your final exam score is less than 100%**

Write two double spaced pages about the job or position you hope to have when you graduate from college (If you already have a job lined up, write about your next career goal). Your essay must include i) a tractable career goal, ii) a reason why you want to be hired/accepted to this position, iii) a step-by-step plan of how you intend to achieve your goal.

After graduating from CU Boulder, I hope to achieve a career within the gaming industry. My dream job lies within the game dev department as an animations engineer or game designer. Since receiving my first gaming console, the ps2, I have fallen in love with the beauty and technical aspects of video games. Playstation network is filled with original IP games and sub companies that allow developers to create games within a safe space. I would love to be hired as an animations engineer at Sony Naughty Dog. One of the main reasons why I would love to be hired at the company is because of the company's ambitious developed games. Although companies like Square Enix and Insomniac Games have produced many great games, Naughty Dog is a company that pursues to create original games not for the purpose of sales, but for the devs. I would love to work at a company where the development of a game is not solely purposed to make the consumer happy but is rather more of an art form that is created selfishly from the developers. A saying that I once heard from a Naughty Dog conference was, "our game is not made to love selfishly but made to be loved selflessly". In my free time I do art. I started with studio art as young as I can remember and began teaching myself digital art as of 3 years ago. Animation engineers/programmers create the code

Name: Sasha Farhat

ID: 105887541

**CSCI 3104, Algorithms**  
**Final Exam Summer 2020 (60 points)**

**Escobedo & Jahagirdar**  
**Summer 2020, CU-Boulder**

---

that allows characters to move in a natural and fluid way. In order to achieve this position I need to show a strong understanding of 3D design/animations as well as code. The job requires both technical skills and art skills. Preparing for the job, I have slowly crafted my art portfolio. When I have more time I hope to start learning about creating 3D animations or models. This would give me a high advantage when applying to the position. Pursuing a computer science degree therefore prepares me for the coding aspect of the job. Next summer I hope to study abroad in Japan and complete my Japanese minor while I am there. Because I won't be in the states during that summer, I will apply to gaming internships in summer 2022. My goal ultimate goal is to land an internship at Blizzard or Sony PlayStation, but if not achieved I plan to apply to many other internships that can get me the experience I seek to set my foot through the door of the gaming industry. After graduating and completing an internship, I will immediately go for my graduates degree in Game Design. Although I am not sure what school yet I would like to apply to, I know I want to get it at a university that has a strong foundation in this subject. After receiving my graduates degree, I then hope to apply to my desired jobs.