Correctness

Question 1

Consider the following algorithm to determine the parity of the sum of a list

Note that *not 0 = 1, not 1 = 0*

1 Mark

```
def parity(L[1..n])
  p = 0
  i = 1
  ###here###
  while i <= n
    if L[i] % 2 == 1
       p = not p
    i += 1
  return p
```

1, 2, 8, 5, 7, 4, 3
0 0 0 1 0

Write down a useful invariant for the algorithm above. Your invariant must be true at the line marked with the comment "###here###", i.e. just before the check of the while condition.

p represents the parity sum of element in L[1...i]

Question 2

Show that the invariant you identified in Question 1 holds at the line marked #Here, before the loop is entered for the first time. before loop : sum is 1 ⇒ p = 0 since list now is only 1

0.5 Mark

Question 3

Show that your invariant is true at the line marked ###here### each time the loop runs (excluding *i=1*, since you will have proved that in Question 2.

Make sure your argument is logically sound.

3 Marks

• when i = k ⇒ sum = 1 + L[k]
if L[k] is odd ⇒ sum is even ⇒ p = 1
else ⇒ sum is odd ⇒ p = 0

Question 4

Since you have now shown that the invariant you chose in Question 1 holds for all iterations of the loop, now argue that the algorithm is correct.

0.5 Mark

Example: L = [1, 2, 8, 11]

• i = 1 ⇒ L[1] = 2 ⇒ { p = 0, sum = 1+2 = 3
• i = 2 ⇒ L[2] = 8 ⇒ { p = 0, sum = 1+2+8 = 11
• i = 3 ⇒ L[3] = 11 ⇒ { p = 1, sum = 1+2+8+11 = 22

• First iteration i = 1 ⇒ L[i] is odd ⇒ 1 + L[1] is even ⇒ p = 1

• Assume that + L[k] is odd ⇒ 1 + ... L[k] is even ⇒ p = 1

• Prove for i = k+1
+ L[k+1] is odd ⇒ 1 + ... L[k+1] = 1 + ... L[k] + L[k+1] ⇒ p = 0
1 + even number
then sum = odd number ⇒ p = 0

by induction math p represent the parity of sum elements in L[1...n]

Complexity and Recurrence Relation

Question 5

3 Marks

f(n) is Θ(g(n)) means that f grows asymptotically as fast g, as opposed to O(g(n)) which only gives an upper bound.

For a constant *c*, consider the recurrence relation given by:

- $T(n) = c$; if $n=1$
- $T(n) = 2*T(n/2) + c*n^2$; if $n>1$.

Which of the following statements is true?

Select one:

- a. $T(n) = \Theta(n^2)$
- b. $T(n) = \Theta(n^2 * \log n)$
- c. $T(n) = \Theta(n^2 * \log n * \log n)$
- d. $T(n) = \Theta(n^3)$
- e. $T(n) = \Theta(n^3 * \log n)$

Question 6

1 Mark

Given an algorithm which runs in *O(NlogN)* time, which of the following are **possible auxiliary space complexities** for this algorithm?

Mark all that are possible.

Select one or more:

- a. O(1)
- b. O(N)
- c. O(NlogN)
- d. O(N^2)

Question 7

What is the **auxiliary space complexity** of the following algorithm (expressed in big-O)?

1 Mark

The input ***n*** **is always a positive integer.**

```
f(n):
  lst = [None]*n    # O(n)
  for i in range(n):
    lst = [i] * n
```

Select one:

- a. O(1)
- b. O(log(N))
- c. O(N)
- d. O(N^2)

Quick Sort and Quick Select

Question 8

3 Marks

Consider a list of lap times for a stage in the game of Fall Guys for Twitch Rivals, represented as a list of *N* unsorted positive floats with values from *0.0* seconds to *100.00* seconds.

To pass this stage, participants would need to clock in a lap time no more than *20.0* seconds.

As the game master however, you would want at least 30% of the players to move on to the next stage; with the following rules:

- If you are within the top-30% fastest time and within the *20.0* seconds requirement, you would move onto the next stage without any issues.
- If you are within the top-30% fastest time but is above the *20.0* seconds requirement, then you would be penalized for the next round. The penalty is how much slower are you compared to the median of the top-30%; if the time is below the median then there is no penalty.

Describe an efficient algorithm using quick select to determine the sum of all penalties; in O(N) time complexity with the assumption that you have access to a quickselect algorithm which runs in O(N) time.

```
temp := QuickSelect(arr, 30% * len(arr))
// store the sublist of the smallest 30% elements from arr to temp

mid = Median(temp)
// find median of the top 30% fastest time

sum = 0

for i = 0 -> len(temp) do:
    if temp[i] > mid then
        sum += mid - temp[i]

return sum
```

Dynamic Programming

Question 9

For this question you **must** answer in the following format:

Write the *indices* of the houses that you have chosen, in ascending order, separated **only** by a single comma with no spaces. For example if the answer were houses 7, 8 and 9, you would write 7,8,9

2 Marks

Recall the following problem from the **Dynamic Programming studio**:

You are trying to sell to a row of houses. You know the profits which you will earn from selling to each house *1..n*. If you sell to house *i*, you cannot sell to houses *i+1* or *i-1*. **What is the maximum profit you can obtain?**

Suppose that you have following *DP array*, where cell *i* of the DP array contains the **maximum profit** you can obtain by selling to a subset of houses *1..i*.

Determine which houses you should sell to in order to **maximise your profit** (i.e determine the optimal solution to the problem which this DP array is solving).

i	1	2	3	4	5	6	7
DP[i]	20	20	30	50	60	100	100

1, 3, 5, 7

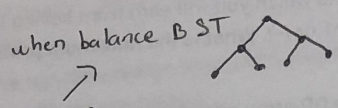
Hash Tables and Dictionary

Question 10

2 Marks

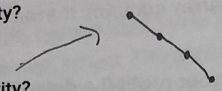
What is the best and worst case time complexity for looking up an item into a separate chaining hash table, where the chains are implemented using Binary Search Trees (BST)?

- M is the size of the table (i.e. the number of BST)
- N is the number of items currently in the table



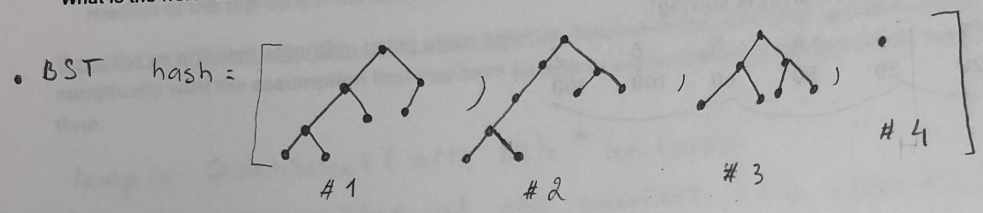
What is the best case complexity?

. O(N + M) . O(M) . O(N) . O(1) . O(log N)
. O(log M)



What is the worst case complexity?

. O(N + M) . O(M) . O(N) . O(1) . O(log N)
. O(log M)

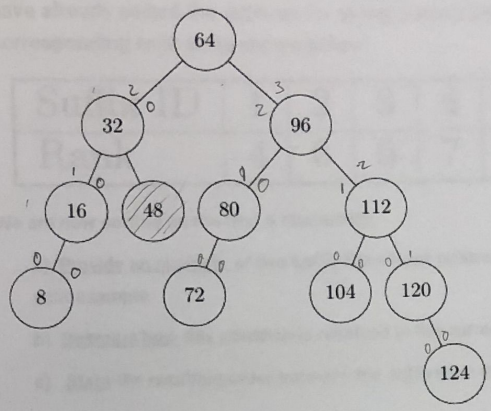


Self-Balancing Trees

Question 11

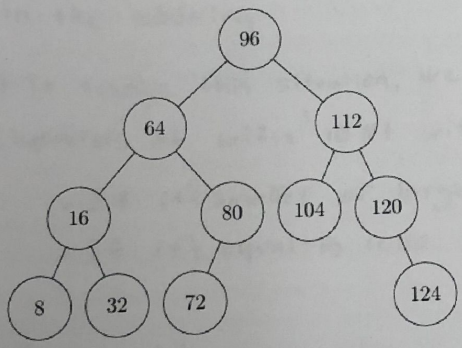
Which of the following would be the result of deleting 48 from the following AVL tree?

1 Mark

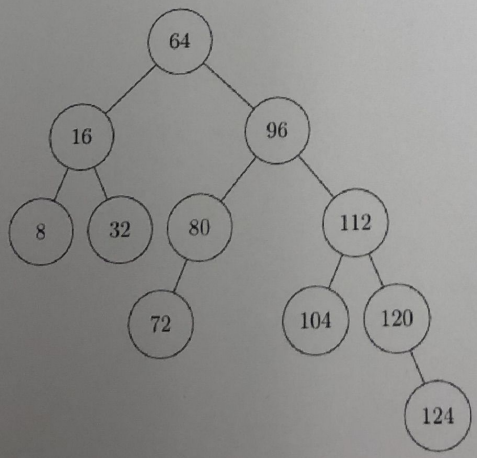


Select one:

a.



b.



Trie, Suffix Tree and Suffix Arrays

Question 12

Assume that we are constructing the **suffix array** for a string S using the **prefix doubling approach**. We have **already sorted** the suffixes for string S according to their **first 2 characters**; with the corresponding **rank array** shown below:

2 Marks

Suffix ID	1	2	3	4	5	6	7	8	9	10	11
Rank	4	6	5	7	4	6	3	5	7	2	1

We are now **sorting on the first 4 characters.**

a) Provide an example of two suffix IDs whose relative order has not been determined at this point. Justify your example.

b) Describe how this situation is resolved in the current iteration of prefix doubling.

c) State the resulting order between the suffixes in your example, after this resolution.

a) The two suffixs ID #1 & #5 share the relative order since they both have the rank at 4, which mean they have the same two initial characters in the substring

b) To resolve this situation, we could process to compare the next two characters of suffix ID #1 with the next two characters of suffix ID #5

- If it's smaller or larger then we reinitialize the rank
- If it's equal ⇒ redo the comparing next two chars process

c)

Graph Representation

Question 16

3 Marks

For each of the following operations, determine its time complexity.

In this question,

- V refers to the number of vertices in the graph
- E refers to the number of edges in the graph
- $N(x)$ refers to the number of neighbors of vertex x.

Assume that in the adjacency list representation, the interior lists are unsorted.

Determining if an edge from u to v exists in an adjacency matrix	. O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Determining if an edge from u to v exists in an adjacency list	. O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Finding all neighbors of u in an adjacency matrix	. O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Finding all neighbors of u in an adjacency list	. O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Performing a complete BFS traversal in an adjacency matrix	. O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Performing a complete BFS traversal in an adjacency list	O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)

Graph and Shortest Distance

Question 17

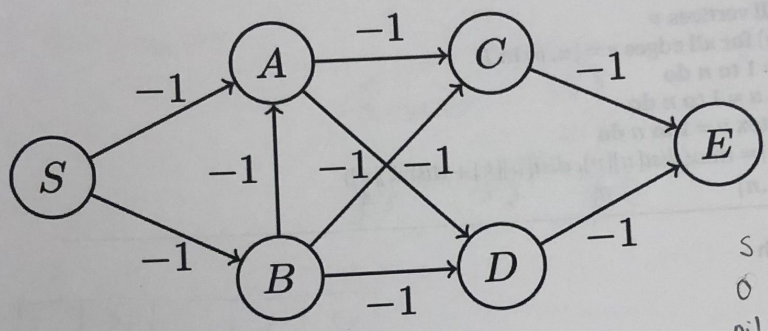
Consider the following version of the Bellman-Ford algorithm:

1 Mark

Algorithm 59 Bellman-Ford

```
1: function BELLMAN_FORD(G = (V, E), s)
2:     dist[1..n] = ∞
3:     pred[1..n] = null
4:     dist[s] = 0
5:     for k = 1 to n − 1 do
6:         for each edge e in E do
7:             RELAX(e)
8:     return dist[1..n], pred[1..n]
```

and the following directed graph



S	A	B	C	D	E
0	∞	∞	∞	∞	∞
nil	nil	nil	nil	nil	nil

Let S be the source node for the execution of the Bellman-Ford algorithm.

If the edges are relaxed in the following order (S,B), (A,C), (B,C), (S,A), (B,A), (C,E), (D,E), (A,D), (B,D).

What is the value of dist[C] after the first iteration of the outer loop is done?

S	A	B	C	D	E
0	-1	-1	-2	-2	-3
nil	S	S	A	A	C

Select one:

- a. dist[C]=∞
- b. dist[C]=-3
- c. dist[C]=-2

Question 18

What is the value of dist[D] after the first iteration of outer loop of Bellman-Ford in the scenario above?

1 Mark

Select one:

- a. dist[D]=∞
- b. dist[D]=-3
- c. dist[D]=-2

Question 19

What is the value of dist[E] after the first iteration of outer loop of Bellman-Ford in the scenario above?

1 Mark

Select one:

- a. dist[E]=∞
- b. dist[E]=-4
- c. dist[E]=-3

Question 20

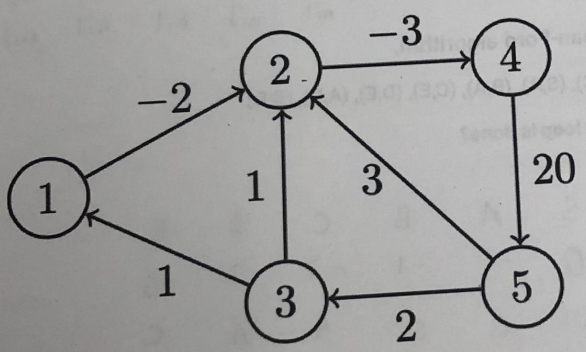
2 Marks

Consider the Floyd-Warshall algorithm

Algorithm 63 Floyd-Warshall

```
1: function FLOYD_WARSHALL(G = (V, E))
2:     dist[1..n][1..n] = ∞
3:     dist[v][v] = 0 for all vertices v
4:     dist[u][v] = w(u, v) for all edges e = (u, v) in E
5:     for each vertex k = 1 to n do
6:         for each vertex u = 1 to n do
7:             for each vertex v = 1 to n do
8:                 dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v])
9:     return dist[1..n][1..n]
```

and the following directed graph



	1	2	3	4	5
1	0	-2		-5	
2		0		-3	
3	1	-1	0	-4	
4				0	20
5		3	2	0	0

After the outer loop of the algorithm finished two iterations, what is the sum of all values in the array *dist* that are not equal to infinity? Just type the numerical answer without punctuation or spaces.

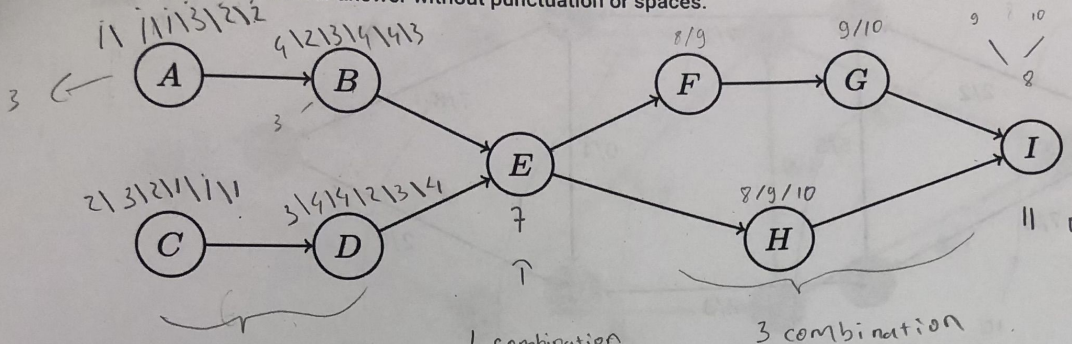
Answer = 11

Directed Acyclic Graph
Question 21

How many distinct topological orderings does the following directed acyclic graph have?

Just type the numerical answer without punctuation or spaces.

3 Marks



1 combination

3 combination

6 combination

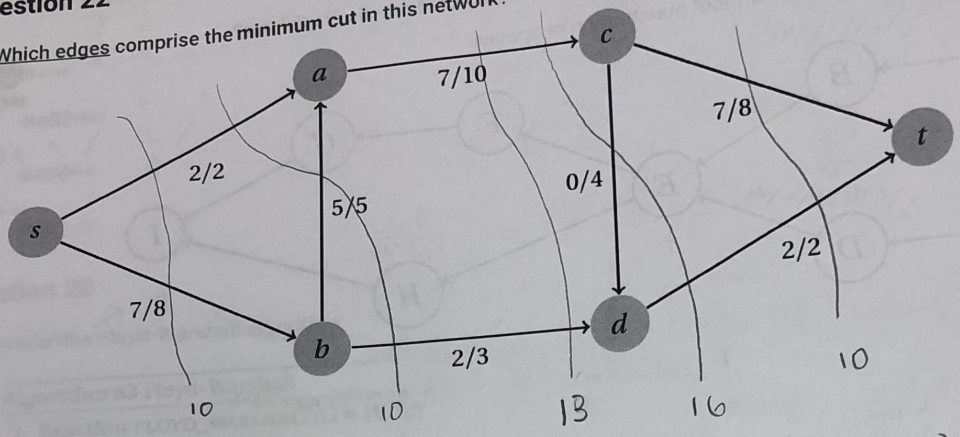
Answer = 18

Network Flow

1.5 Marks

Question 22

Which edges comprise the minimum cut in this network?



Select one or more:

1. s->a
2. s->b
3. b->a
4. a->c
5. b->d
6. c->d
7. c->t
8. d->t

{(s→a), (s→b)} ; {(s→a), (b→a), (b→d)} ; {(c→t), (d→t)}

Answer: (s→a), (d,t), (b→a)

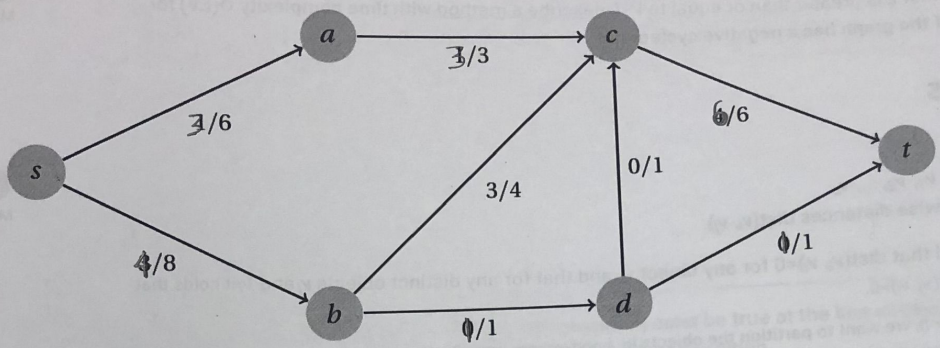
need to find edge has pathflow equals capacity if there is a flow

Question 23

What is the maximum possible flow from s to t in this network?

1.5 Marks

Write your answer with digits, not as a word.



7

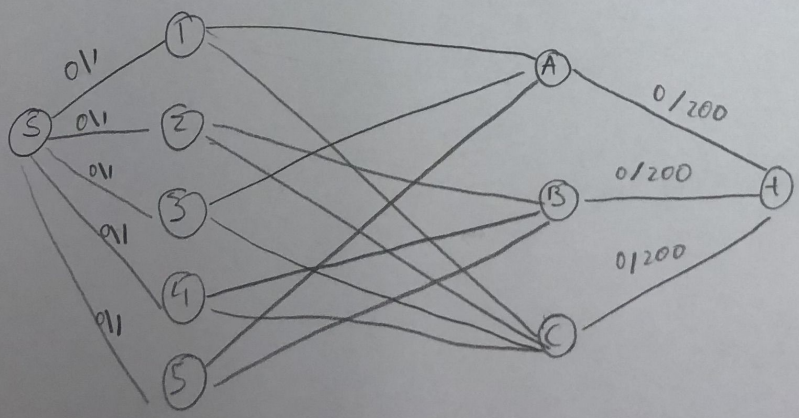
(24) we will use Bellman Ford algorith to calculate the shortest distance from the source to all other node. After that we check for every edge (x,y) in graph if distance from x to y larger than distance at x + weight(x,y) then the graph has negative cycle

```
for n : 1 -> V(G) do
    for edge(x,y) in E(G) do
        if dis[y] > dis[x] + weight(x,y)
            dis[y] = dis[x] + weight(x,y)
            parent[y] = x

for edge (x,y) in E(G) do
        if dis[x] != ∞ and dis[y] > dis[x] + weight(x,y)
            return True

return False
```

(26) This is bipartite matching problems. I will connect the source with all the student nodes, then I will connect every venue nodes with by the student nodes, which choose that venue as their preferrence. The venue node then will conncet with the sink node.
Every edge from source node to student nodes has capacity of 1
Every edge from student nodes to venue nodes has capacity of 1
Every edge from venue nodes to sink node has capacity of 200



Using Ford - Fulkerson we can find the maximum students to fit their pretferred venues.