THE UNIVERSITY OF ADELAIDE
AUSTRALIA

Primary Examination, Semester 1, 2022

Algorithm and Data Structure Analysis
COMPSCI 2201, 7201

Writing Time: 120 mins

Questions	Time	Marks
Answer all 7 questions	120 mins	120 marks
		120 Total

Instructions

- Begin each answer on a new page
- Examination papers must not be removed from the examination room
- Calculators allowed
- This is a open book exam

Materials

- 1 blue book
- Lecture notes (handwritten or printed)
- Textbooks (all books permitted)
- 1 dictionary for translation purposes only

DO NOT COMMENCE WRITING UNTIL INSTRUCTED TO DO SO

Right or Wrong?

Question 1

(a) Indicate whether each of the following statements is true or false. There is one mark for each correct answer and zero marks for each incorrect answer. In the following statements, n is a positive integer.

	Statement
1	$\log(n^{10}) \in \Omega(\log(n))$
2	$2.1 \cdot n^{2.1} + 1500n^2 \in \Theta(n^{2.1})$
3	$3n^2 + 1 \in O(n)$
4	For multiplying two n digit integers, Karatsuba Multiplication's complexity is $O(n)$
5	It is known that there are NP-complete problems that are not in P
6	All connected undirected graphs must contain cycles
7	The height of a BST tree with n nodes is always in $O(\log n)$
8	The height of an AVL tree with n nodes is always in $O(\log n)$
9	The Dijkstra single-source shortest path algorithm can only work for graphs where the edge costs are integers
10	Hash-tables can maintain $O(1)$ access times in the worst case.

[10 marks]

[Total for Question 1: 10 marks]

1. Simplifies to $10(\log(n)) \in \Omega\ \log(n)$

$= \log(n) \in \Omega\ \log(n)$

$\log(n) \geq \log(n)$ true

2. simplifies to $n^{2.1} \in \Theta(n^{2.1})$

$n^{2.1} = n^{2.1}$

3. $n^2 > n$ false

Notation	Meaning
$f(n) = O(g(n))$	Highest degree term of $f(n)$ is less than or equal to highest degree term of $g(n)$
$f(n) = \Omega(g(n))$	Highest degree term of $f(n)$ is greater than or equal to highest degree term of $g(n)$
$f(n) = \Theta(g(n))$	Highest degree terms of $f(n)$ and $g(n)$ are equal
$f(n) = o(g(n))$	Highest degree term of $f(n)$ is strictly less than highest degree term of $g(n)$
$f(n) = \omega(g(n))$	Highest degree term of $f(n)$ is strictly greater than highest degree term of $g(n)$

---

4. false. karatsuba $\in O(207 \cdot n^{\log_2 3})$

$\in O(n^{1.59})$

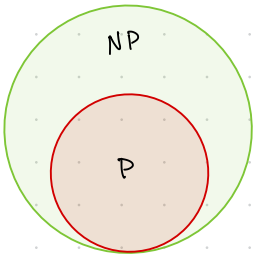
Karatsuba Recurrence Relation

$$T_K(n) \leq \begin{cases} 3n^2 + 2n & \text{if } n < 4 \\ 3 \cdot T_K\left(\lceil \frac{n}{2} \rceil + 1\right) + 6 \cdot 2 \cdot n & \text{if } n \geq 4 \end{cases}$$

$$T_K(n) \leq 207 \cdot n^{\log_2 3}$$

---

5. True.



if NP = P, then false

---

6. False. An MST is connected, undirected and has no cycles.

Minimum Spanning Tree (MST) Problem

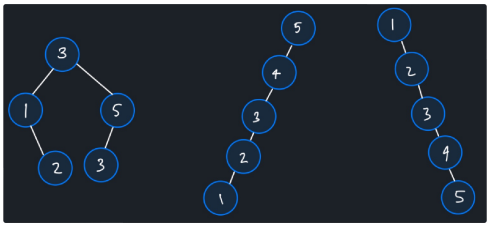
Given a connected, undirected graph $G = (V, E)$ with positive edge costs/weights:

- Edge costs are positive, implying that the connected subgraph of minimal cost does not contain a cycle.
- It is a tree spanning all nodes of the graph (called a spanning tree).

---

7. False. A perfectly balanced BST always have heights of $O(\log(n))$, but these are AVL or red black trees.

A regular BST can have height $O(n)$ - linked list



Consider the height of a BST

It's important to note that there are multiple possible binary search trees for the same set of keys. The structure of a BST can vary widely, depending on the order of insertion and deletion of keys. The image above shows valid BSTs for a set of elements.

The height of a Binary Search Tree (BST) is a crucial factor in determining the running time of search, insertion, and deletion operations. In a perfectly balanced BST of height $\log n$, the height is logarithmic in relation to the number of nodes, resulting in efficient $O(\log n)$ time complexity for these operations.

However, in the worst-case scenario, when the BST is skewed or unbalanced, the height can become linear, leading to a time complexity of $O(n)$ for search, insertion, and deletion. This occurs when the BST resembles a linked list, with each node having only one child.

Operation	Best Case	Average Case	Worst Case
Search	O(1)	O(log n)	O(n)
Insertion	O(1)	O(log n)	O(n)
Deletion	O(1)	O(log n)	O(n)

9. False. As long as the distances are non-negative, it will work.

e.g. Floats would work too!.

---

10. False. In the average case, true. But it can rarely be $O(n)$ in the worst (collisions occur on $n-1$ inserts)

> ⚠ **Hash Table Operation Complexities**
>
> | Case | Complexity | Description |
> | --- | --- | --- |
> | Best Case | $O(1)$ | Direct access as no collisions occur. |
> | Average Case | $O(1)$ | Efficient handling with minimal collisions, assuming a good hash function and low load factor. |
> | Worst Case | $O(n)$ | All keys hash to the same index, leading to a linked list (if chaining is used) or long probing sequences (in open addressing). |

Proofs and Sequences

Question 2

(a) Briefly describe the recursive multiplication algorithm for multiplying two n digit integers.

[4 marks]

(b) A simplified version of the Master Theorem is as follows:
a, b, c and d are constants. For $n = b^k$ for some integer k, consider the recurrence

$$r(n) = \begin{cases} a & \text{if} \quad n = 1 \\ cn + d \cdot r(n/b) & \text{if} \quad n > 1 \end{cases}$$

We have

$$r(n) = \begin{cases} \Theta(n) & \text{if} \quad d < b \\ \Theta(n \log n) & \text{if} \quad d = b \\ \Theta(n^{\log_b d}) & \text{if} \quad d > b \end{cases}$$

Use the above to derive the complexity of recursive multiplication for multiplying two n digit integers. For this question, you only need to consider $n = 2^k$ for some integer k.

[4 marks]

(c) Let T be a function that satisfies
1) $T(n) = n^2$ for all positive integer n that is a multiple of 10.
2) $T(n)$ is increasing in n.
Prove that $T(n) \in O(n^2)$.

[8 marks]

[Total for Question 2: 16 marks]

2 a).

The Idea

Reduce complexity of school method multiplication through Divide & Conquer! Let a and b be the numbers to be multiplied.

Step 1: Split the integers

$$a = \overbrace{12}^{a_1}\ \overbrace{34}^{a_0}$$
$$b = \underbrace{56}_{b_1}\ \underbrace{78}_{b_0}$$
$$n = digits$$
$$k = \frac{n}{2}$$

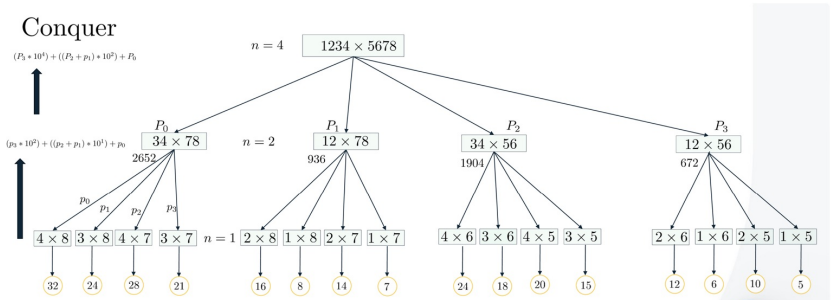
Step 2: Calculate Partial Products

$$p_0 = a_0 \times b_0 = 34 \times 78 = 2652$$
$$p_1 = a_1 \times b_0 = 12 \times 78 = 936$$
$$p_2 = a_0 \times b_1 = 34 \times 56 = 1904$$
$$p_3 = a_1 \times b_1 = 12 \times 56 = 672$$

Step 3: Add the Aligned products

$$\begin{aligned} a \times b &= p_3 \times B^{2k} + B^k \times (p_1 + p_2) + p_0 \\ &= 6720000 + 284000 + 2652 \\ &= 7006652 \end{aligned}$$

Notice, that we can split the integers even further and recursively find those multiplications



$$\mathbf{a} * \mathbf{b} = (p_3 * 10^n) + ((p_2 + p_1) * 10^{n/2}) + p_0$$

b. no clue.

c)

Hashing and Skiplists

Question 3

(a) Assume you have a hash table of size $m = 11$ and hash keys are lowercase alphabetic strings. Given is the following hash function

$$h(k) = code(lastLetter(k)) \bmod 11,$$

where $lastLetter$ extracts the last letter from its input and $code$ returns an integer representing the position of the letter in the alphabet. So, for example $h(\text{anna})$ returns 1, $h(\text{job})$ returns 2 and $h(\text{noon})$ returns 3. Draw the hashtable described above after the insertion of the elements:

`"plate" "pass" "band" "print" "case" "class" "zoo"`

when collisions are resolved through chaining.

[7 marks]

(b) Explain why, when resolving hash-table collisions via linear probing, one cannot remove an entry from the hash table by resetting the slot to NIL.

[3 marks]

(c) Assume that the size of a hash table is given by $m = 2^r$ for some integer $r > 1$. We map a key k into one of the m slots using the hash function $h(k) = k \; mod \; m$. Give one reason why this might be a poorly chosen hash function.

[4 marks]

(d) Given the following sequence of coin tosses, where H stands for head and T stands for tails:
T H T T H H T T H T H T H H T H H T T H H T H T T H T T T H T H T.
Build a skip list containing the following values: 1 40 11 85 86 5. Show the full state of the skip list after each insertion. **Hint:** explicitly state the meaning you attach to heads and tails at the start of your answer.

[5 marks]

[Total for Question 3: 19 marks]

3. a)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

table:

0	1	2	3	4	5	6	7	8	9	10
		200	case	band	plate			class	pass	print

b). because if affects other keys. keys must be shifted back into place after a deletion, so that integrity is kept. Out of place tombstones will incorrectly lead to keys not being found in the table.

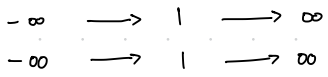
c). high number of collisions with similar keys, i.e likely that keys must have different MSB to not collide.

d). number of flips to reach a Head is height of entry: T H = height of 2

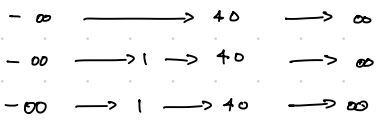
THTTHHTTHTHTHHTHHTTHHTHTTHTTTHTHT.

Build a skip list containing the following values: 1 40 11 85 86 5.

1. insert (1), height = 2

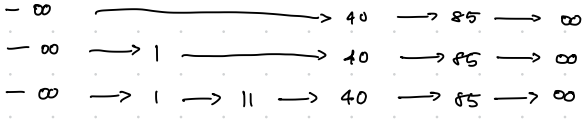


2. insert (40), height = 3

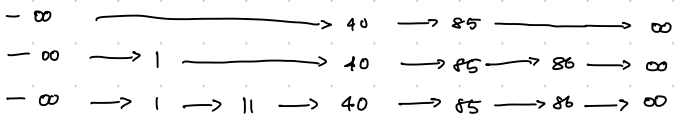


3. insert (11), height = 1

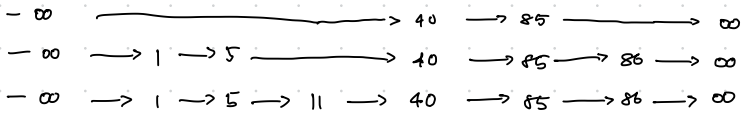
4. insert (85), height = 3



5. insert (86), height = 2



6. insert (5), height = 2



Trees

Question 4

(a) Briefly describe the process for deleting a value from a binary search tree.

[6 marks]

(b) Is the following statement correct?

Let v be the largest node of a binary search tree T. The parent of v must be the second largest node.

Please prove the above statement or provide a counter example.

[5 marks]

(c) Draw a sequence of diagrams showing the insertion of the values:

```
[9,1,2,3,8,7,4,6,5]
```

into an empty AVL tree, in the order shown above.

You must:

- Show the resulting tree immediately after each insertion step (that is *before* any balancing has taken place).
- Show the resulting tree after balancing operation(s).

[10 marks]

(d) What are the maximum numbers of rotations needed for

1) inserting a new value into an AVL tree with n nodes

2) deleting a value from an AVL tree with n nodes

[6 marks]

[Total for Question 4: 27 marks]

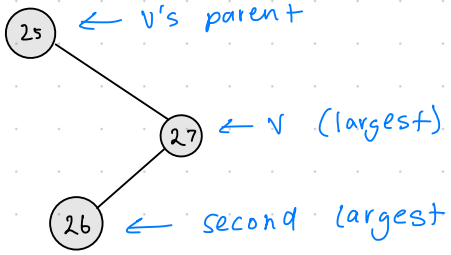
4 a).

BST Deletion

The `delete` function removes a node with the given `key` from the AVL tree. It follows these steps:

1. If the `root` is `None`, return `root` as there is nothing to delete.
2. If the `key` is less than the `root`'s key, recursively delete the key from the left subtree.
3. If the `key` is greater than the `root`'s key, recursively delete the key from the right subtree.
4. If the `key` is equal to the `root`'s key, we have found the node to delete:
   - If `key` is stored at a leaf, delete this leaf and the incoming edge.
   - If `key` has 1 child, redirect the pointer pointing to `k` to `k`'s child and delete `k`.
   - If `key` has 2 children:
     - Search in the tree for the largest element `x` smaller than `k`.
     - In the left subtree of `k`, follow the right path as long as possible to find `x`.
     - Swap `x` and `k` and recursively delete `k`.
5. Return the updated `root` of the subtree.

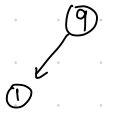
b).



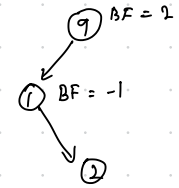
statement is false!

c). 1. insert (9)

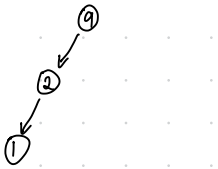
2. insert (1)



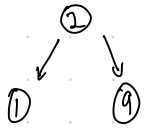
3. insert 2



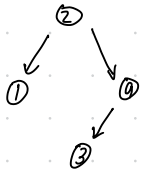
left rotation at 1



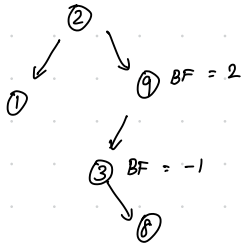
right rotation at 9



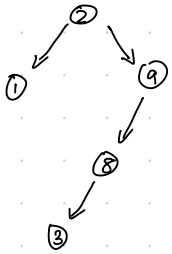
4. insert 3



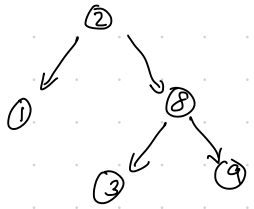
5. insert 8



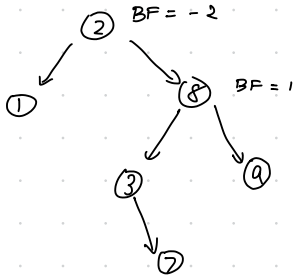
left rotation at 3



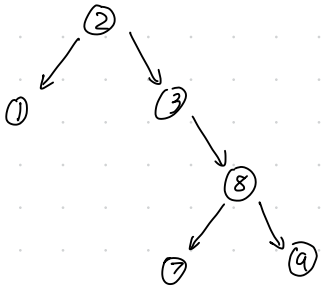
right rotation at 9



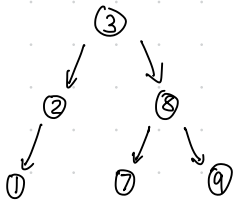
6. insert 7



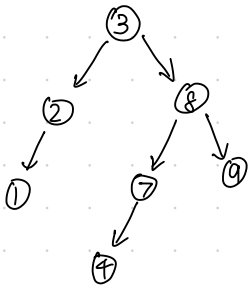
right rotation at 8



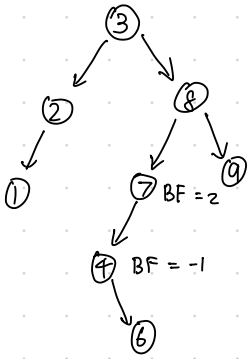
left rotation at 2



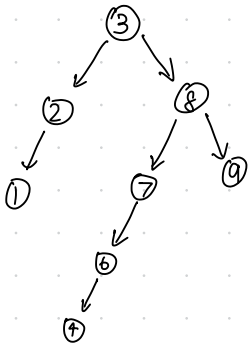
7. insert 4



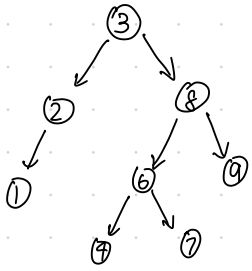
8. insert 6



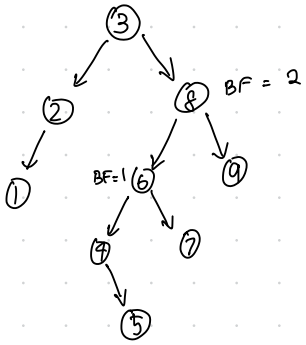
left rotation at 6



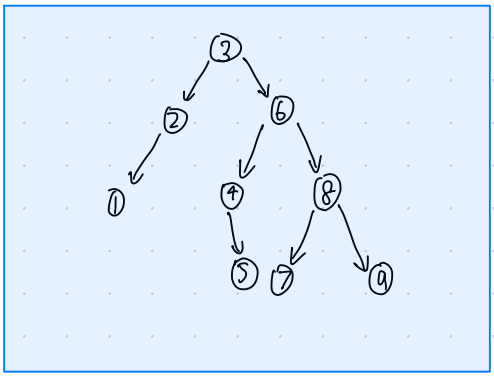
right rotation at 7



insert 5



right rotation at 8



d). 2 rotations at most for insertion - O(1)
log(n) rotations at most for deletion - O(log n)

Graph Representations and Traversals

Question 5

(a) For each of the following problems state whether you think the best graph representation of the problem is an adjacency list or an an adjacency matrix. Include an explanation of your choice in your answer.

i. You need to trace the ancestors of living members of the extended British royal family back 600 years.

[2 marks]

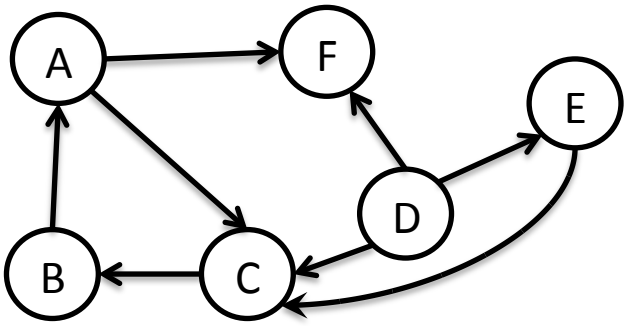
ii. You need to tabulate the minimum road distances between all cities with more than $50,000$ people in Japan

[3 marks]

iii. For an engineering project you are given a large list of tasks; the estimated time required for each task; and a set of dependencies between tasks. You need to calculate an estimated time of completion for the project.

[2 marks]

(b) Use the strongly connected components algorithm described in lectures to find the strongly connected components of following graph. Start your traversal at node A. In your answer show your working including any graph artefacts and other information you use while you apply the algorithm.



[7 marks]

(c) What is the running time of depth-first search, as a function of the number of vertices $|V|$ and the number of edges $|E|$, if the input graph is represented by an adjacency matrix instead of an adjacency list? Does it depend on the number of edges?

[4 marks]

[Total for Question 5: 18 marks]

5 a)

i). Adjacency List. Unlikely that each node (person) will be connected to majority of other nodes – this would make a very sparse graph.

An adjacency matrix would waste a lot of space and it's constant time access isn't that useful for this.

ii). Adjacency Matrix. Since we want to find edges that connect 1 node to all other nodes, we would have a very dense graph (assuming only cities with pop 50,000 are stored).

Finding distances would also be fast for GPS applications because of it's O(1) lookup time.

iii). Assuming the dependencies are other tasks. An adjacency list. Since tasks won't be connected to most other tasks, the graph will be sparse. The list also allows you to recursively search linked lists to see what tasks need to be completed.

b). DFS Pass 1

1. Stack ↑ | processing | visited ↑
   A

2. Stack ↑ | processing | visited ↑
   F, C | A

3. Stack ↑ | processing | visited ↑
   C | F | A

4. Stack ↑ | processing | visited ↑
   B | C | F, A

5. Stack ↑ | processing | visited ↑
   processing: B | C, F, A

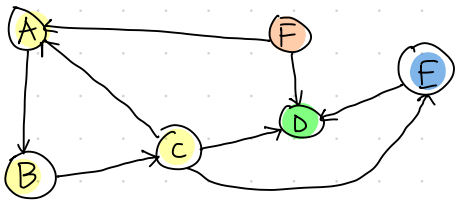
6. Stack ↑ | processing | visited ↑
   D | B, C, F, A

7. Stack ↑ | processing | visited ↑
   E | D | B, C, F, A

8. Stack ↑ | processing | visited ↑
   processing: E | D, B, C, F, A

9. Stack ↑ | processing | visited ↑
   processing: | E, D, B, C, F, A

DFS Pass 2 – reverse edges



starting from E

SCCs : {A, B, C}, {F}, {D}, {E}

c) $O(E+V)$ for adjacency list

$O(V^2)$ for matrix (check all possible edges)

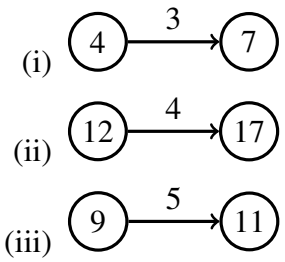
Shortest Path Algorithms

Question 6

(a) Consider a weighted directed graph $G = (V, E, w)$ and let X be a shortest $s - t$ path for $s, t \in V$. If we double the weight of every edge in the graph, setting $w'(e) = 2 \cdot w(e)$ for each $e \in E$, then will X be still a shortest $s - t$ path in G? Explain your answer.

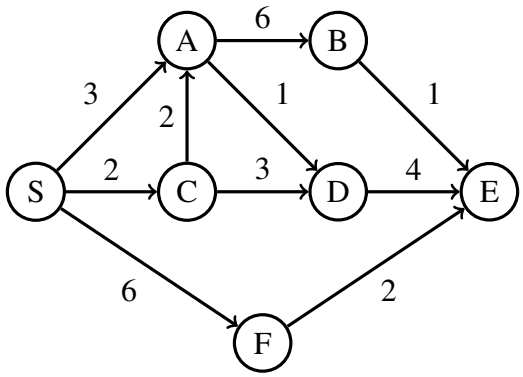
[3 marks]

(b) Draw the result of relaxing the following edges. Note: in the diagram the number in each node represents the current known minimum distance to the source.



[3 marks]

(c) Run Dijkstra's algorithm on the following directed graph, starting at vertex S. What is the order in which vertices get removed from the priority queue? What is the resulting shortest-path tree?



[7 marks]

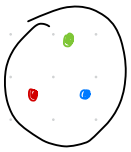
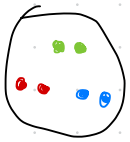
(d) Briefly explain, with the help of an example, why negative weight cycles in graphs prevent the calculation of finite path lengths to nodes on the far side of such cycles.

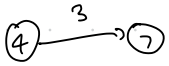
[3 marks]

[Total for Question 6: 16 marks]

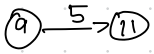
6 a). Yes, it will be. The shortest edge at each iteration is still the same, just twice the size (as is everything else)

It's like saying what are the odds of picking a green marble

 $= \frac{1}{3}$  $= \frac{2}{6} = \frac{1}{3}$

b) i). 

ii) 

iii). 

c) 1. visited : {S} unvisited : {A, B, C, D, E, F}

Nodes	S	A	B	C	D	E	F
Distance	0	3	∞	2	∞	∞	6
Parent	/	S	/	S	/	/	S

2. visited : {S, C} unvisited : {A, B, D, E, F}

Nodes	S	A	B	C	D	E	F
Distance	0	3	∞	2	5	∞	6
Parent	/	S	/	S	C	/	S

3. visited : {S, C, A} unvisited : {B, D, E, F}

Nodes	S	A	B	C	D	E	F
Distance	0	3	9	2	4	∞	6
Parent	/	S	A	S	A	/	S

4. visited : {S, C, A, D} unvisited : {B, E, F}

Nodes	S	A	B	C	D	E	F
Distance	0	3	9	2	4	8	6
Parent	/	S	A	S	A	D	S

5. Visited : { S, C, A, D, F } unvisited : { B, E }

Nodes	S	A	B	C	D	E	F
Distance	0	3	9	2	4	8	6
Parent	/	S	A	S	A	D	S

6. Visited : { S, C, A, D, F, E } unvisited : { B }

Nodes	S	A	B	C	D	E	F
Distance	0	3	9	2	4	8	6
Parent	/	S	A	S	A	D	S

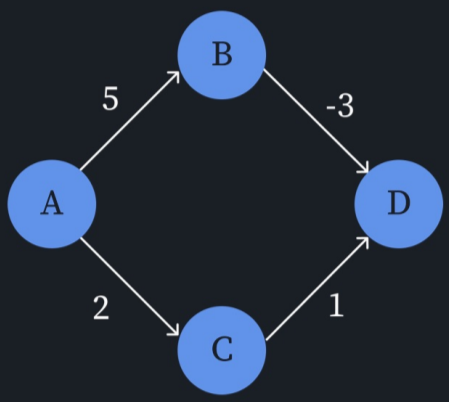
7. Visited : { S, C, A, D, F, E, B } unvisited : { }

Nodes	S	A	B	C	D	E	F
Distance	0	3	9	2	4	8	6
Parent	/	S	A	S	A	D	S

d)

⚠ Dijkstra's Algorithm and Negative Edge Weights

Consider the following graph:



If we run Dijkstra's algorithm starting from node A, it will first visit node C with a distance of 2. Then, it will visit node D with a distance of 3 (via the path A -> C -> D). At this point, node B is still in the queue with a distance of 5.

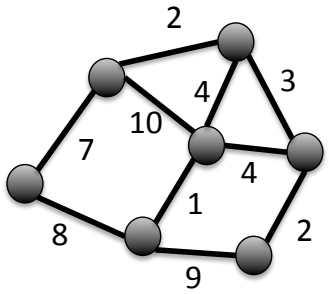
When the algorithm dequeues node B, it will find that the distance to D from B is 2 (5 - 3). However, Dijkstra's algorithm will not update the distance to D because it has already been visited and marked as final.

The actual shortest path from A to D is A -> B -> D with a total distance of 2. Dijkstra's algorithm misses this path because it greedily marks the distances as final and does not revisit nodes to update their distances when a negative edge is encountered.

Minimum Spanning Trees and P vs NP

Question 7

(a) Draw two different minimum spanning trees for the graph below.



In your answer show the final trees including the weights on the links of the trees.

[6 marks]

(b) Give an example problem that is NP-Complete.

[2 marks]

(c) Give an example problem that is in P.

[2 marks]

(d) Is the minimum spanning tree problem in P? Please explain your answer.

[4 marks]

[Total for Question 7: 14 marks]

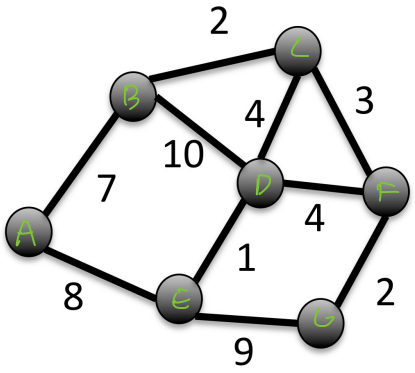
End of exam

7. a) Using Kruskal's Algorithm

step 1: sort edges in ascending order

{ 1, 2, 2, 3, 4, 4, 7, 8, 9, 10 }

step 2: union / find on each node / edge starting from smallest node.



0. sets : {A}, {B}, {C}, {D}, {E}, {F}, {G}

MST : { }

1. sets : {A}, {B}, {C}, {D, E}, {F}, {G}

MST : { D $\overset{1}{\longleftrightarrow}$ E }

2. sets : {A}, {B}, {C}, {D, E}, {F, G}

MST : { D $\overset{1}{\longleftrightarrow}$ E, F $\overset{2}{\longleftrightarrow}$ G }

3. sets : {A}, {B, C}, {D, E}, {F, G}

MST : { D $\overset{1}{\longleftrightarrow}$ E, F $\overset{2}{\longleftrightarrow}$ G, B $\overset{2}{\longleftrightarrow}$ C }

4. sets : {A}, {B, C, F, G}, {D, E}

MST : { D $\overset{1}{\longleftrightarrow}$ E, F $\overset{2}{\longleftrightarrow}$ G, B $\overset{2}{\longleftrightarrow}$ C, C $\overset{3}{\longleftrightarrow}$ F }

5. This is where MST diverges. Can pick either of the 4 edges

sets : {A}, {B, C, F, G, D, E}

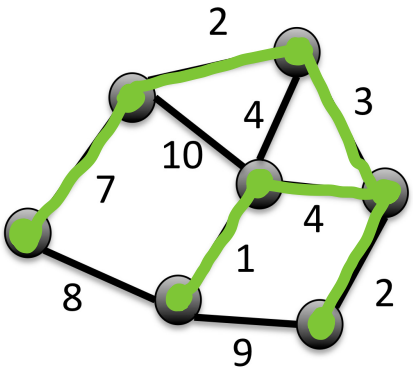
MST : { D $\overset{1}{\longleftrightarrow}$ E, F $\overset{2}{\longleftrightarrow}$ G, B $\overset{2}{\longleftrightarrow}$ C, C $\overset{3}{\longleftrightarrow}$ F, D $\overset{4}{\longleftrightarrow}$ F OR D $\overset{4}{\longleftrightarrow}$ C }

6. sets : {A, B, C, F, G, D, E}

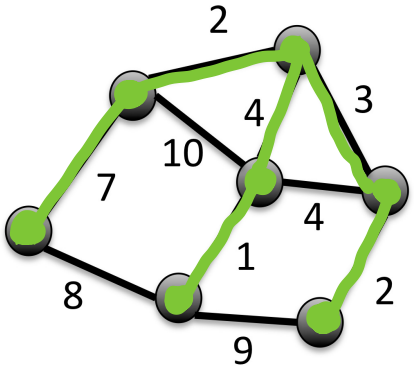
MST : {D $\overset{1}{\longleftrightarrow}$ E, F $\overset{2}{\longleftrightarrow}$ G, B $\overset{2}{\longleftrightarrow}$ C, C $\overset{3}{\longleftrightarrow}$ F, D $\overset{4}{\longleftrightarrow}$ F OR D $\overset{4}{\longleftrightarrow}$ C, A $\overset{7}{\longleftrightarrow}$ B}

DONE!

Final MSTs :



OR



b. The Boolean Satisflability Problem.

c) Minimimum Spanning Trees – O(ElogE)

d). Yes, it can be solved using Prim- O(ElogV) or kruskal - O(ElogE) both of which $\in$ O(P)