National University of Computer & Emerging Sciences, Karachi

Fall-2021 Department of Computer Science
Final Exam
29 December 2021, 09:00 AM – 09:30 AM
Part (A)

Course Code: CS2009	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Dr. Fahad Sherwani, Dr. Farrukh Saleem, Waheed Ahmed, Waqas Sheikh, Sohail Afzal	
Student Roll No:	**Section:**

Instructions:

- Must be submitted with in 30 minutes
- No extra sheets allowed. Must solve in given space
- You are allowed to submit paper before 30 minutes and start part(b)

Time: 30 minutes **Max Marks: 10**

Question # 1 **[0.5*6 = 3 marks]**

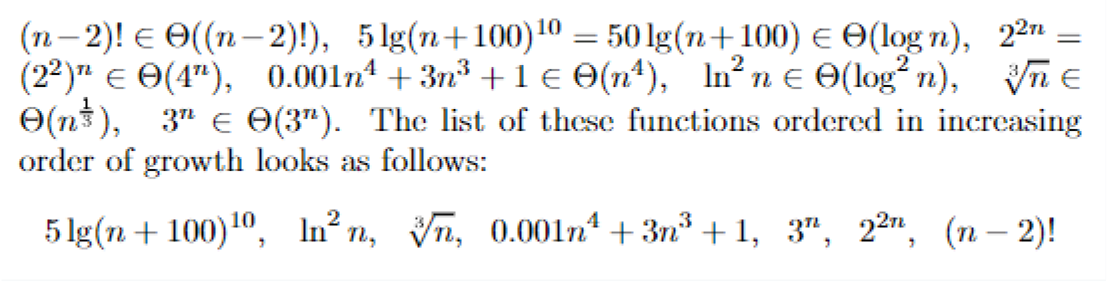
Answer the following questions. You must explain in only 3-4 lines.

(a) Describe Big theta.

c1g(n)<=f(n)<=c2g(n)

(b) List the following functions according to their order of growth from the lowest to the highest.

$$(n-2)!, 5\log(n+100)^{10}, 2^{2n}, 0.001n^4+3n^3+1, \ln^2 n, \sqrt[3]{n}, 3^n.$$



$(n-2)! \in \Theta((n-2)!)$, $5\lg(n+100)^{10} = 50\lg(n+100) \in \Theta(\log n)$, $2^{2n} = (2^2)^n \in \Theta(4^n)$, $0.001n^4 + 3n^3 + 1 \in \Theta(n^4)$, $\ln^2 n \in \Theta(\log^2 n)$, $\sqrt[3]{n} \in \Theta(n^{\frac{1}{3}})$, $3^n \in \Theta(3^n)$. The list of these functions ordered in increasing order of growth looks as follows:

$$5\lg(n+100)^{10},\quad \ln^2 n,\quad \sqrt[3]{n},\quad 0.001n^4+3n^3+1,\quad 3^n,\quad 2^{2n},\quad (n-2)!$$

(c) Explain why the statement, "The running time of algorithm A is at least $O(n^2)$," is meaningless.

We just care about the upper bound and the lower bound of T(n)$T(n)$.
The statement: T(n)$T(n)$ is at least O(n^2)$O(n2)$.

- Upper bound: Because "T(n)T(n) is at least O(n^2)O(n2)", there's no information about the upper bound of T(n)T(n).
- Lower bound: Assume f(n) = O(n^2)f(n)=O(n2), then the statement: T(n) \ge f(n)T(n)≥f(n), but f(n)f(n) could be any fuction that is "smaller" than n^2n2. For example, constant, nn, etc, so there's no conclusion about the lower bound of T(n)T(n), too.

Therefore, the statement, "The running time of algorithm AA is at least O(n^2)O(n2)," is meaningless.

(d) Define recurrence relations and enlist methods to solve them.

An equation or inequality that describes a function in terms of its value on smaller inputs. 1. Recursion tree method 2 Iteration method Substitution method 4 Master method

(e) In which conditions dynamic programing does not work. Give suitable example.

Basic property of problems which can be solved using DP are.

1. Overlapping Sub-problems.
2. 2.Optimal Sub-structure

If any problem doesn't have either of above property then problem can't be solved using DP. for example shortest path can be solved using DP but longest path can't solved, because longest path doesn't hold optimal sub-structure property

(f) Suppose there is a maximization problem, where the approximate solution has the cost of 25

And optimal solution has the cost of 30. Find the approximation ratio.

30/25 = 1.2 approximation ratio

Question # 2 **[0.5*8 = 4 marks]**

Write the complexity and the corresponding design strategy (Divide and Conquer/ Dynamic Programming / Greedy) of the given algorithms

ALGORITHMS	Worst Case	Write below whether the algorithm belongs to Divide and Conquer/ Dynamic Programming / Greedy / None of them. If an algorithm can be solved with several design techniques, write anyone of them with the corresponding complexity.
Quick Sort	$O(n^2)$	Divide and Conquer
Radix Sort	O(n*k/d)	None of them
Max-Heapify operation	$O(logn)$	None of them
Add Vertex in Adjacency Metric	$O(V^2)$	Greedy
Rod-Cutting	$O(n^2)$ / O (2^n)	Dynamic Programing
Dijkstra's (Array DS)	V^2	Greedy
Prims	Elog V	Greedy
Maximum Sub-array Sum	O(n)/ O(nlogn) /O (n^2)	Dynamic/ divide and Conquer/ None of them.

Question # 3 **[1*3 = 3 marks]**

For each of the following questions indicate whether it is true or false and justify using some example assuming a function.

For all positive f(n)

1. $\boxed{\omega(f(n)) + O(f(n)) = \Theta(f(n))}$

Solution

Let $f(n) = n^2$

$\omega(n^2) + O(n^2) = \Theta(n^2)$

$n^3 + n = \Theta(n^2)$

$\boxed{\text{False}}$

2. For all positive f(n),

$\boxed{f(n) + o(f(n)) = \text{theta}(f(n))}$

Solution

Let $f(n) = n^2$

$n^2 + o(n^2) = \Theta(f(n))$

$n^2 + n = \Theta(n^2)$

$\boxed{\text{True}}$

③ For all positive $f(n), g(n), h(n)$

if $f(n) = O(g(n))$

and $f(n) = \Omega(h(n))$

then

$g(n) + h(n) = \Omega(f(n))$

Solution

Let $f(n) = n$

$g(n) = n\log n$

$h(n) = \log(n)$

Thus

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

$\boxed{\text{True}}$

National University of Computer & Emerging Sciences, Karachi

Fall-2021 Department of Computer Science

Final Exam

29 December 2021, 09:00 AM – 12:00 AM

Part (B)

Course Code: CS2009	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Dr. Fahad Sherwani, Dr. Farrukh Saleem, Waheed Ahmed, Waqas Sheikh, Sohail Afzal	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **9 questions** on **4 pages**.
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.
-

Time: 180 minutes **Max Marks: 40**

Question # 4 [6 marks]

a) What is meant by P and NP Problems?

- P problems
  - (The original definition) Problems that can be solved by **deterministic Turing machine** in polynomial-time.
  - (A equivalent definition) Problems that are solvable in polynomial time.
- NP problems
  - (The original definition) Problems that can be solved by **non-deterministic Turing machine** in polynomial-time.
  - (A equivalent definition) Problems that are **verifiable** in polynomial time.
    - Given a solution, there is a polynomial-time algorithm to tell if this solution is correct.

b) Let X be a problem that belongs to the class NP. Then explain why the following are incorrect or correct statements?

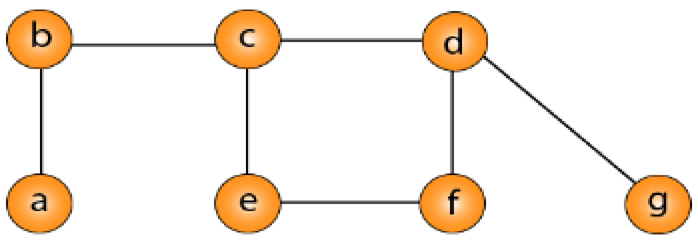
A. Explain why it is incorrect) There is no polynomial time algorithm for X
B. (Explain why it is incorrect) If X can be solved deterministically in polynomial time then P = NP
C. (Explain why it is correct) If X is NP-hard, then it is NP-complete
D. (Explain why it is incorrect) X may be undecidable

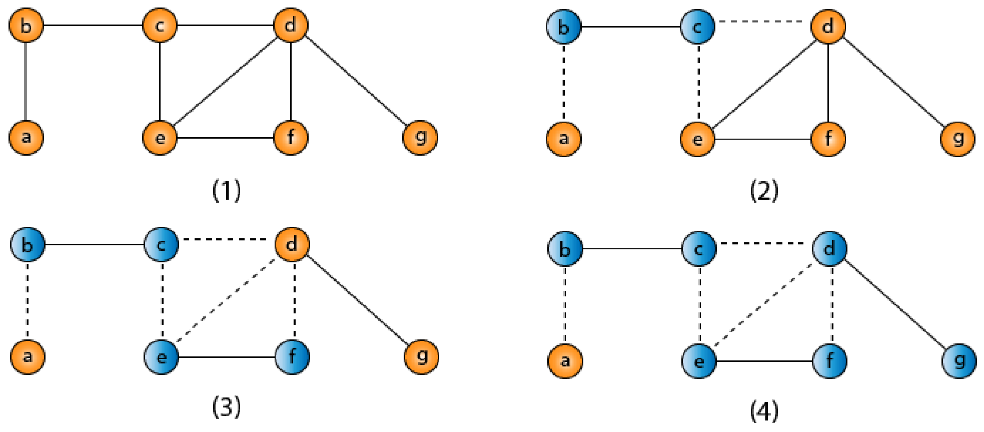
- Sol: (A) is incorrect because set NP includes both P(Polynomial time solvable) and NP-Complete . (B) is incorrect because X may belong to P (same reason as (A)) (C) is correct because NP-Complete set is intersection of NP and NP-Hard sets. (D) is incorrect because all NP problems are decidable in finite set of operations.

c) Does P ! = NP mean that no problem exists which can be solved and checked in polynomial time?

No. P != NP means that P is a proper subset of NP. That means, there are problems that are in NP but not in P.

d) Using 2-approximation greedy method studied during lectures, find the size of the vertex cover of the following graph:





Question # 5

[4 marks]

Prove that Travelling Salesman approximation algorithm is a 2-approximation algorithm**.** Give arguments and example as well.

Solution:

Polynomial running time obvious, simple MSTPrim takes $\theta(|V|^2)$, computing pre-order walk takes no longer.

Correctness obvious, pre-order walk is always a tour.

Let H^* denote an optimal tour for given set of vertices.

Deleting any edge from H^* gives a spanning tree.

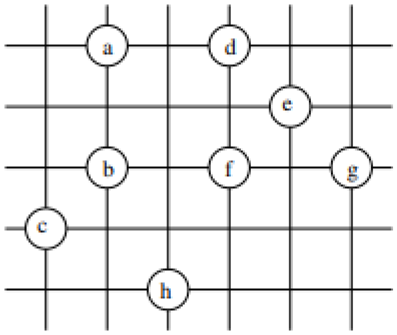
Thus, weight of **minimum** spanning tree is lower bound on cost of optimal tour: $c(T) \leq c(H^*)$

A **full walk** of T lists vertices when they are **first visited**, and also when they are **returned to**, after visiting a subtree.

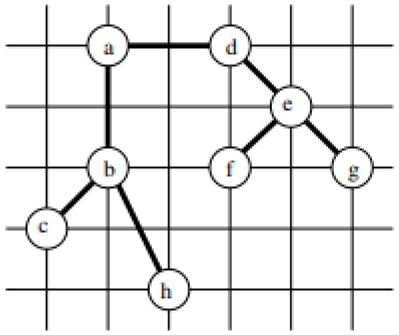
Example: a,b,c,b,h,b,a,d,e,f,e,g,e,d,a

Full walk W traverses every edge **exactly twice**, thus

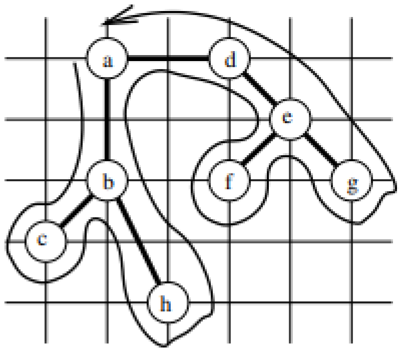
$$c(W) = 2c(T)$$



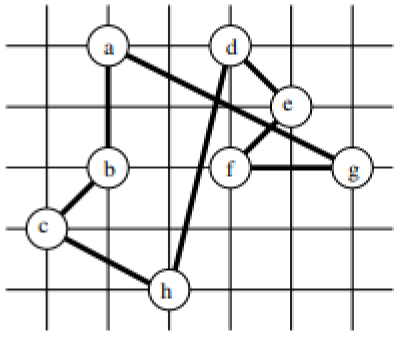
Set of points, lie in grid



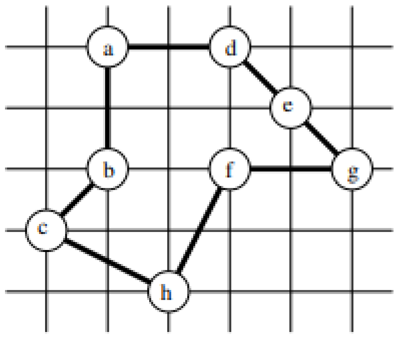
MST, root a



Pre–order walk



Resulting tour, cost ca. 19.1

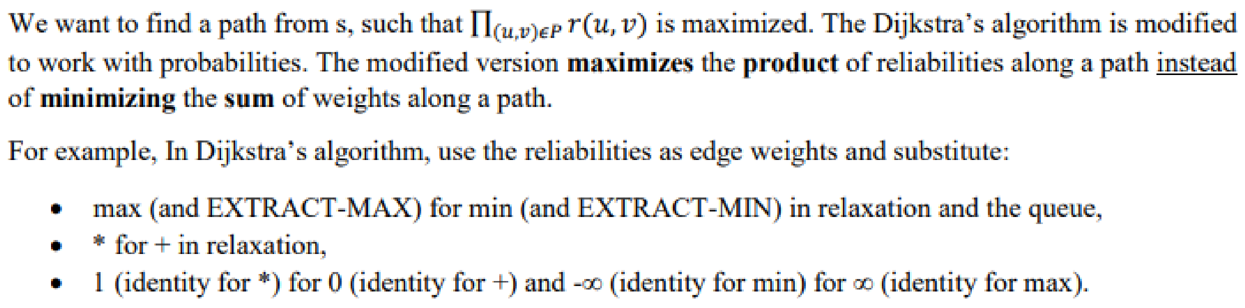


Optimal tour, cost ca. 14.7

<u>Question # 6</u> **[4 marks]**

We are given a directed graph G = (V, E) on which each edge (u,v) ϵ E has an associated value r (u,v), which is a real number in the range $0 \leq r(u,v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret r(u,v) as the probability that the channel from u to v will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.
(Note: Reliability means higher value)

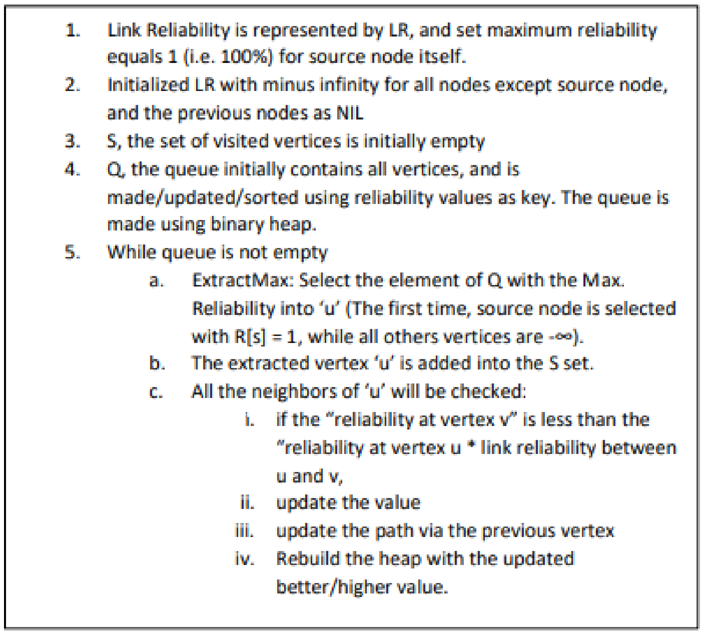
Solution:


We want to find a path from s, such that $\prod_{(u,v)\epsilon P} r(u,v)$ is maximized. The Dijkstra's algorithm is modified to work with probabilities. The modified version **maximizes** the **product** of reliabilities along a path <u>instead</u> of **minimizing** the **sum** of weights along a path.

For example, In Dijkstra's algorithm, use the reliabilities as edge weights and substitute:

- max (and EXTRACT-MAX) for min (and EXTRACT-MIN) in relaxation and the queue,
- * for + in relaxation,
- 1 (identity for *) for 0 (identity for +) and -∞ (identity for min) for ∞ (identity for max).

```
1. LR[s] ← 1
2. for all v ∈ V–{s}
     a. do LR[v] ← -∞
     b. prev [v] = nil
3. S ←∅
4. Q←V
5. while Q ≠∅ do
     a. u ← maxReliability (Q, LR)
     b. S←S ∈{u}
     c. for all v ∈ neighbors[u] do
          i. If LR[v] < LR[u] * r(u,v)
          ii. LR[v] = LR[u] * r(u,v)
          iii. prev [v] = u
          iv. updateKey(Q, v)
6. return dist
```

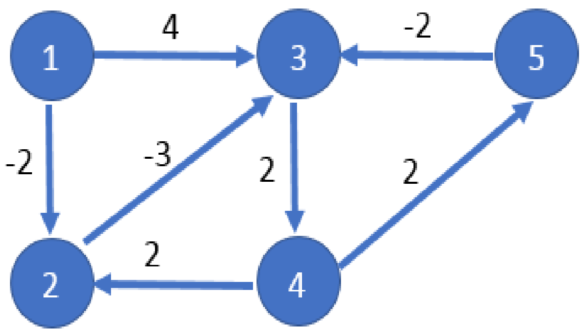

1. Link Reliability is represented by LR, and set maximum reliability equals 1 (i.e. 100%) for source node itself.
2. Initialized LR with minus infinity for all nodes except source node, and the previous nodes as NIL
3. S, the set of visited vertices is initially empty
4. Q, the queue initially contains all vertices, and is made/updated/sorted using reliability values as key. The queue is made using binary heap.
5. While queue is not empty
    a. ExtractMax: Select the element of Q with the Max. Reliability into 'u' (The first time, source node is selected with R[s] = 1, while all others vertices are -∞).
    b. The extracted vertex 'u' is added into the S set.
    c. All the neighbors of 'u' will be checked:
        i. if the "reliability at vertex v" is less than the "reliability at vertex u * link reliability between u and v,
        ii. update the value
        iii. update the path via the previous vertex
        iv. Rebuild the heap with the updated better/higher value.

Question # 7 **[3 + 1 + 1 = 5 marks]**

(a) Use Floyd Warshall algorithm to find the shortest path from every vertex to every other vertex for the graph given below. The result must contain two matrices/tables, one matrix (D) shows the shortest cost from each vertex to all other vertices, and the second matrix (Π) should show the previous vertex use to reach the destination vertex.

(b) Mark the steps on the resultant Π matrix/table to show, how it will be used, to trace the route from vertex 5 to vertex 2.

(c) From the resultant matrices, identify all those pairs of vertex for which there is no path.



Solution:

Use Floyd Warshall algorithm to find the shortest path from every vertex to every other vertex for the graph given below. The result must contain two matrices/tables, one matrix (D) shows the shortest cost from each vertex to all other vertices, and the second matrix (Π) should show the previous vertex use to reach the destination vertex.

$D0 = \begin{array}{c c c c c c} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & 4 & \infty & \infty \\ 2 & \infty & 0 & -3 & \infty & \infty \\ 3 & \infty & \infty & 0 & 2 & \infty \\ 4 & \infty & 2 & \infty & 0 & 2 \\ 5 & \infty & \infty & -2 & \infty & 0 \end{array}$	$\Pi 0 = \begin{array}{c c c c c c} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 1 & - & - \\ 2 & - & - & 2 & - & - \\ 3 & - & - & - & 3 & - \\ 4 & - & 4 & - & - & 4 \\ 5 & - & - & 5 & - & - \end{array}$
$D1 = \begin{array}{c c c c c c} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & 4 & \infty & \infty \\ 2 & \infty & 0 & -3 & \infty & \infty \\ 3 & \infty & \infty & 0 & 2 & \infty \\ 4 & \infty & 2 & \infty & 0 & 2 \\ 5 & \infty & \infty & -2 & \infty & 0 \end{array}$	$\Pi 1 = \begin{array}{c c c c c c} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 1 & - & - \\ 2 & - & - & 2 & - & - \\ 3 & - & - & - & 3 & - \\ 4 & - & 4 & - & - & 4 \\ 5 & - & - & 5 & - & - \end{array}$

$$D2 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & -5 & \infty & \infty \\ 2 & \infty & 0 & -3 & \infty & \infty \\ 3 & \infty & \infty & 0 & 2 & \infty \\ 4 & \infty & 2 & -1 & 0 & 2 \\ 5 & \infty & \infty & -2 & \infty & 0 \end{array}$$

$$\Pi 2 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 2 & - & - \\ 2 & - & - & 2 & - & - \\ 3 & - & - & - & 3 & - \\ 4 & - & 4 & 2 & - & 4 \\ 5 & - & - & 5 & - & - \end{array}$$

$$D3 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & -5 & -3 & \infty \\ 2 & \infty & 0 & -3 & -1 & \infty \\ 3 & \infty & \infty & 0 & 2 & \infty \\ 4 & \infty & 2 & -1 & 0 & 2 \\ 5 & \infty & \infty & -2 & 0 & 0 \end{array}$$

$$\Pi 3 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 2 & 3 & - \\ 2 & - & - & 2 & 3 & - \\ 3 & - & - & - & 3 & - \\ 4 & - & 4 & 2 & - & 4 \\ 5 & - & - & 5 & 3 & - \end{array}$$

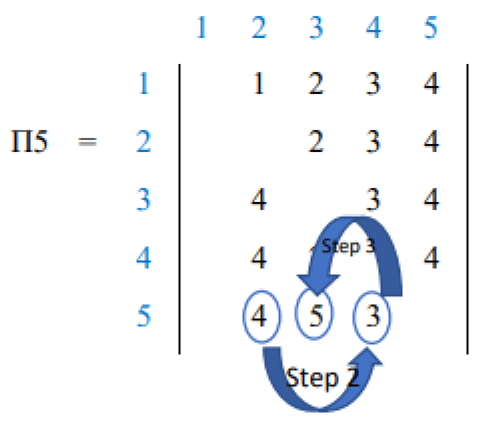
$$D4 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & -5 & -3 & -1 \\ 2 & \infty & 0 & -3 & -1 & 1 \\ 3 & \infty & 4 & 0 & 2 & 4 \\ 4 & \infty & 2 & -1 & 0 & 2 \\ 5 & \infty & 2 & -2 & 0 & 0 \end{array}$$

$$\Pi 4 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 2 & 3 & 4 \\ 2 & - & - & 2 & 3 & 4 \\ 3 & - & 4 & - & 3 & 4 \\ 4 & - & 4 & 2 & - & 4 \\ 5 & - & 4 & 5 & 3 & - \end{array}$$

$$D5 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & -2 & -5 & -3 & -1 \\ 2 & \infty & 0 & -3 & -1 & 1 \\ 3 & \infty & 4 & 0 & 2 & 4 \\ 4 & \infty & 2 & -1 & 0 & 2 \\ 5 & \infty & 2 & -1 & 0 & 0 \end{array}$$

$$\Pi 5 = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & - & 1 & 2 & 3 & 4 \\ 2 & - & - & 2 & 3 & 4 \\ 3 & - & 4 & - & 3 & 4 \\ 4 & - & 4 & 2 & - & 4 \\ 5 & - & 4 & 5 & 3 & - \end{array}$$

(b) For the Π matrix/table given below, the empty cells in-fact represents NIL (/) value. Which means that a path does not exists. Mark the steps on the resultant Π matrix/table to show, how it will be used, to trace the route from vertex 5 to vertex 2.



$$\Pi 5 = \begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & & 1 & 2 & 3 & 4 \\
2 & & & 2 & 3 & 4 \\
3 & & 4 & & 3 & 4 \\
4 & & 4 & & & 4 \\
5 & & \textcircled{4} & \textcircled{5} & \textcircled{3} & \\
\end{array}$$

In the resultant Π table, to reach from vertex 5 to vertex 2, we look at the cell (5,2), which contain the previous vertex 4 to reach vertex 2. Now, to reach vertex 4, we look at the cell (5,4), which contain previous vertex 3 to reach vertex 4 (as shown by step 2). Finally, to reach vertex 3, we look at cell (5,3), which contain previous vertex 5 (self), that is directly connected (as shown by step 3).

(c) From the resultant matrices, identify all those pairs of vertex for which there is no path.

Solution:

2 -> 1, 3 -> 1, 4 -> 1, 5 -> 1, and all self-nodes

<u>Question # 8</u> **[2+3 = 5 marks]**

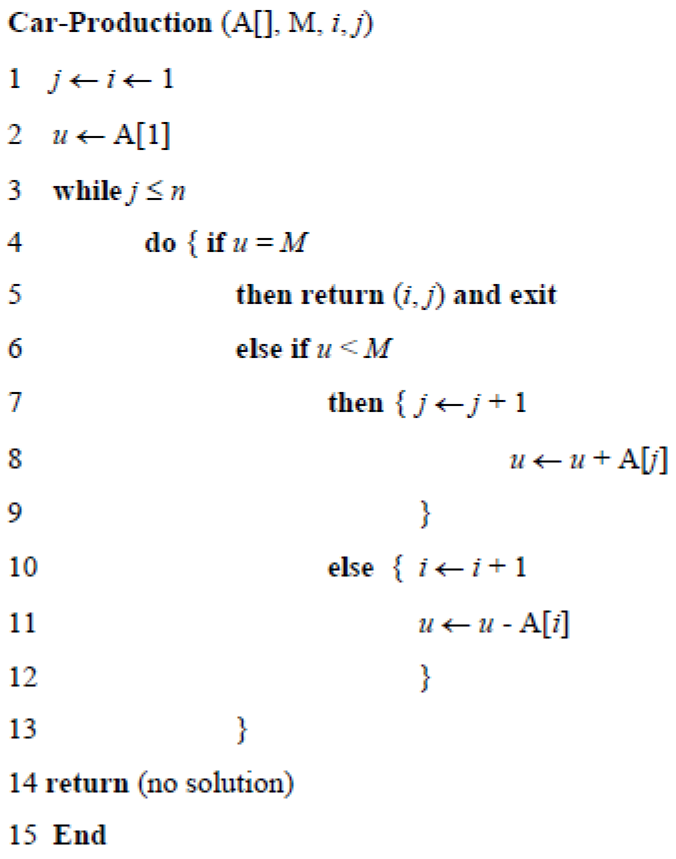
Let array A= $A[1], A[2], \ldots, A[n]$ stores the number of cars produced by a company in the past n years,. The company wants to find if there is a period of consecutive years (from i to j), such that the total number of cars produced in this period is exactly equal to M. if there exist i and j,

$1 \le i \le j \le n$, such that $\sum_{k=i}^{j} A[k] = M$.

Example : Input : A [2, 4, 6, 3, 8, 4, 1, 10], M= 15

a) Design brute force algorithm for this problem
b) Design O(n) time greedy algorithm to solve this problem.

Solution:

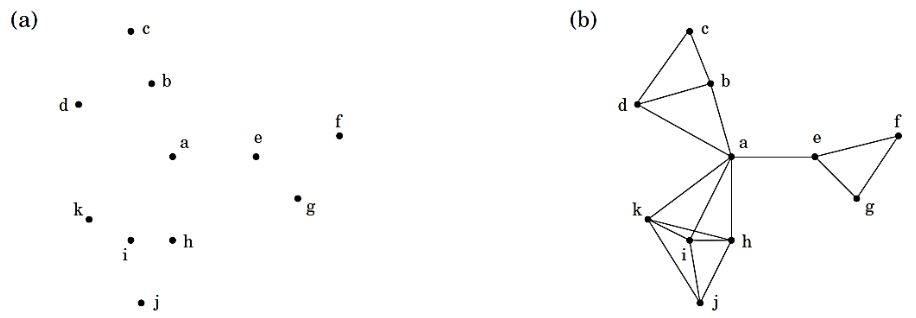


```
Car-Production (A[], M, $i$, $j$)
$j \leftarrow i \leftarrow 1$
$u \leftarrow$ A[1]
while $j \leq n$
        do { if $u = M$
                then return ($i$, $j$) and exit
                else if $u < M$
                        then { $j \leftarrow j + 1$
                                    $u \leftarrow u +$ A[$j$]
                              }
                        else  { $i \leftarrow i + 1$
                              $u \leftarrow u$ - A[$i$]
                              }
                }
return (no solution)
End
```

Question # 9 **[2+3 = 5 marks]**

The dots in below *Figure(a)* represent a collection of towns, and the edges between the towns in *Figure(b)*, shows that the two towns u and v with the edge (u,v) are at-most 30 miles apart. The town committee is deciding where to open the schools.
There are only two constraints:
a) Each school should be in a town.
b) No student should have to travel more than 30 miles to reach one of the school.

With the above given constraints, design approximation algorithm to determine the minimum number of schools needed?



Solution:

This is a typical *set cover* problem. For each town x, let S_x be the set of towns within 30 miles of it. A school at x will essentially cover. these other towns. The question is then, how many sets S_x must be picked in order to cover all the towns in the county?

```
SET COVER
Input: A set of elements B; sets S1;… Sm _ B
Output: Solution set C
U = B
C = Φ
While (U is not empty)
        select set Si with the largest number of uncovered elements
        U =U - $S_i$
        C = C U {S}
End while
Return C
```

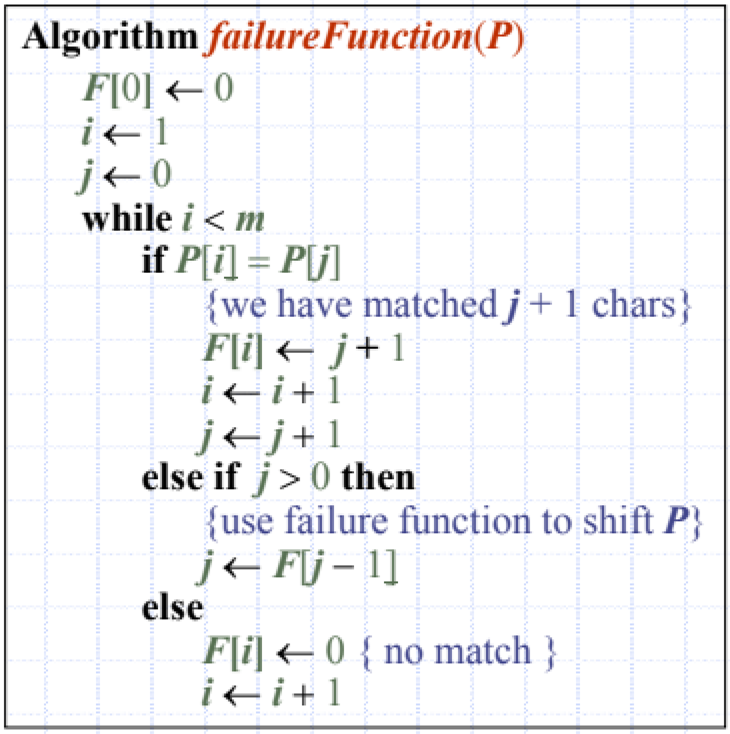
```
Mark all the input sets as uncovered
Make an empty solution set U
Repeat until all elements of B are covered:
        Pick the set Si with the largest number of uncovered elements.
        Place the Si in U
        Mark all sets connected to Si
Return U
```

<u>Question # 10</u> **[3 marks]**

Construct failure function table (that we build up in KMP string matching algorithm) for given pattern :

T:	b	a	c	b	a	b	a	b	a	b	a	c	a	c	a

P:	a	b	a	b	a	c	a



```
Algorithm failureFunction(P)
    $F[0] \leftarrow 0$
    $i \leftarrow 1$
    $j \leftarrow 0$
    while $i < m$
        if $P[i] = P[j]$
            {we have matched $j + 1$ chars}
            $F[i] \leftarrow j + 1$
            $i \leftarrow i + 1$
            $j \leftarrow j + 1$
        else if $j > 0$ then
            {use failure function to shift $P$}
            $j \leftarrow F[j - 1]$
        else
            $F[i] \leftarrow 0$ { no match }
            $i \leftarrow i + 1$
```

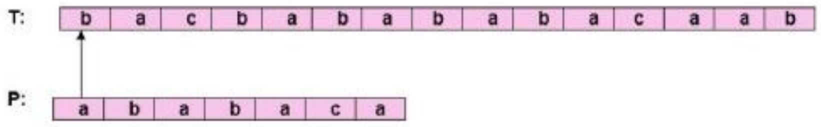
Solution:

The failure function table with computational steps:

q	1	2	3	4	5	6	7
p	a	b	A	b	a	c	a
π	0	0	1	2	3	0	1

Step1: i=1, q=0

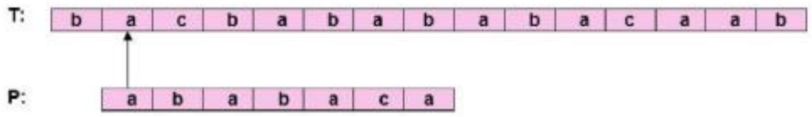
Comparing P [1] with T [1]



P [1] does not match with T [1]. 'p' will be shifted one position to the right.

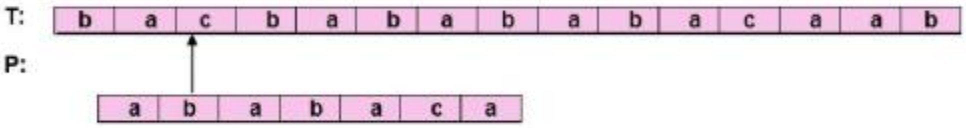
Step2: i = 2, q = 0

Comparing P [1] with T [2]



Step 3: i = 3, q = 1

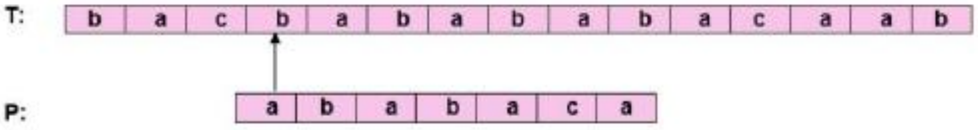
Comparing P [2] with T [3] P [2] doesn't match with T [3]



Backtracking on p, Comparing P [1] and T [3]

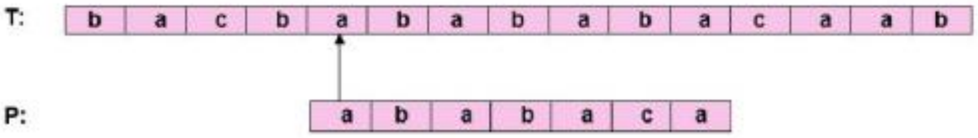
Step4: i = 4, q = 0

Comparing P [1] with T [4] P [1] doesn't match with T [4]



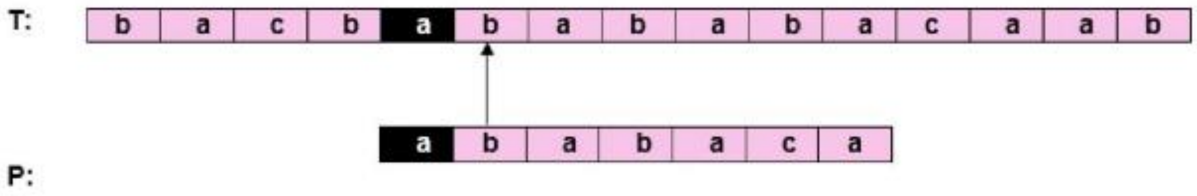
Step5: i = 5, q = 0

Comparing P [1] with T [5] P [1] match with T [5]



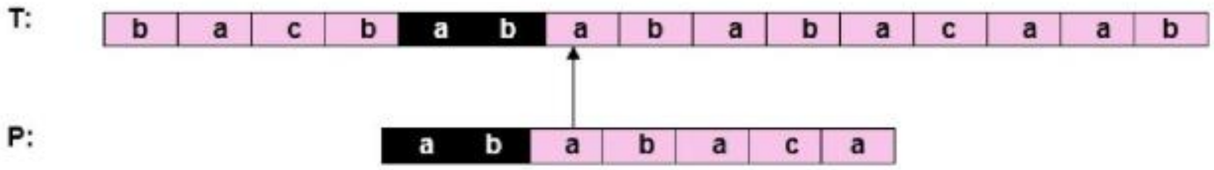
Step6: i = 6, q = 1

Comparing P [2] with T [6]   P [2] matches with T [6]



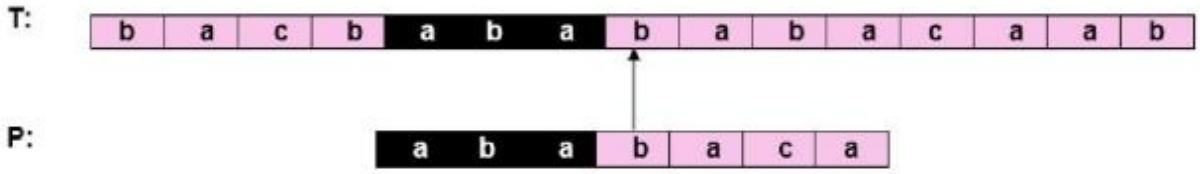
Step7: i = 7, q = 2

Comparing P [3] with T [7]   P [3] matches with T [7]



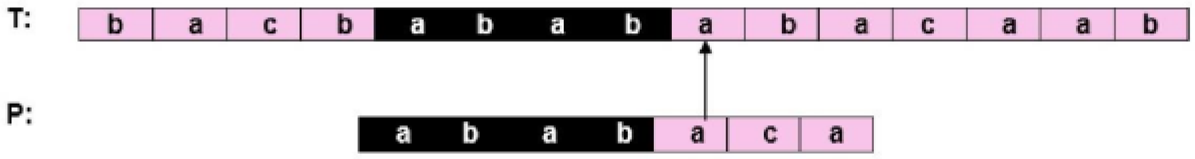
Step8: i = 8, q =3

Comparing P [4] with T [8]   P [4] matches with T [8]



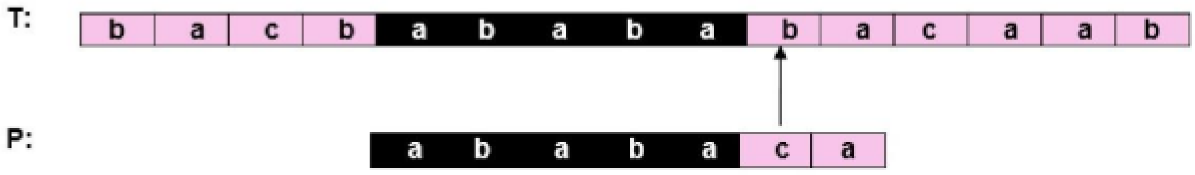
Step9: i = 9, q = 4

Comparing P [5] with T [9]   P [5] matches with T [9]



Step10: i = 10, q = 5

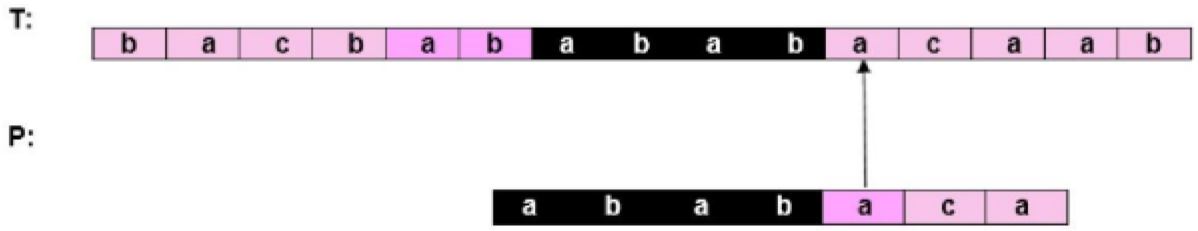
Comparing P [6] with T [10]   P [6] doesn't match with T [10]



Backtracking on p, Comparing P [4] with T [10] because after mismatch q = π [5] = 3

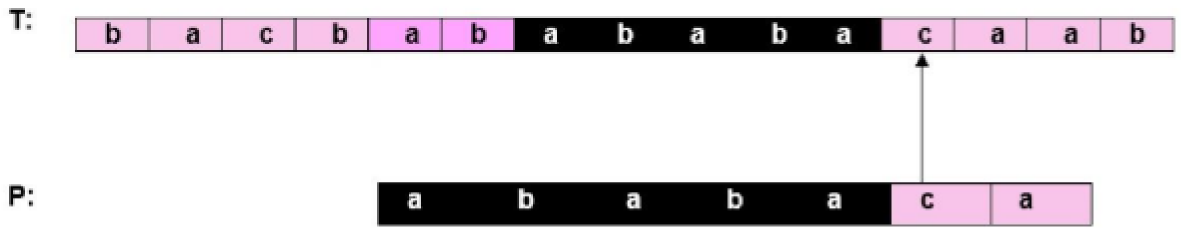
Step11: i = 11, q =4

Comparing P [5] with T [11] P [5] match with T [11]



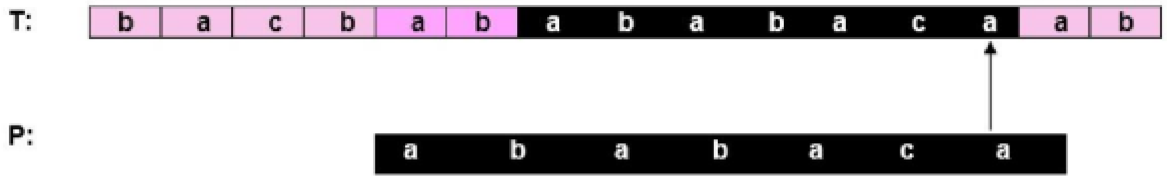
Step12: i = 12, q = 5

Comparing P [6] with T [12] P [6] matches with T [12]



Step13: i = 3, q = 6

Comparing P [7] with T [13] P [7] matches with T [13]

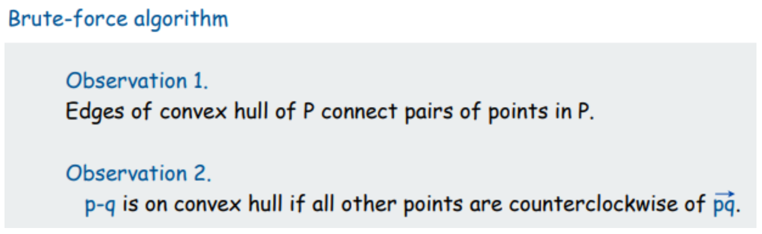


- Pattern 'P' has been found to complexity occur in a string 'T.' The total number of shifts that took place for the match to be found is i-m = 13 - 7 = 6 shifts.

<u>Question # 11</u> **[1+3 =4 marks]**

a. Given the line segment $\overrightarrow{(a,b)}$ in below figure, design a brute force algorithm to determine whether this line will be the part of convex hull. Hint: use counterclockwise turn technique.

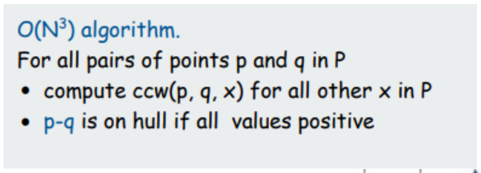
Solution:

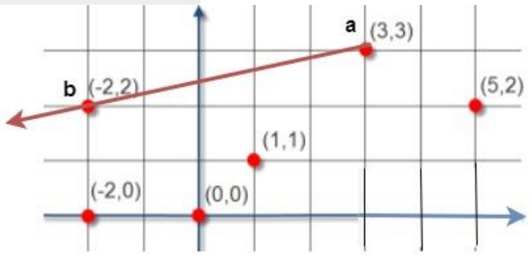


Brute-force algorithm

Observation 1.
Edges of convex hull of P connect pairs of points in P.

Observation 2.
p-q is on convex hull if all other points are counterclockwise of $\overrightarrow{pq}$.


$O(N^3)$ algorithm.
For all pairs of points p and q in P
- compute ccw(p, q, x) for all other x in P
- p-q is on hull if all values positive



Dry run

Compute ccw of line segment $\overrightarrow{(a,b)}$ to all other points and check whether it is positive or not

```
1. a =  [3, 3] ,  b= [-2, 2]  , c= [-2, 0]

     ( -2 - 3 ) * ( 0 - 3 ) - ( -2 - 3 ) * ( 2 - 3 )
     Ans=  10
     its counter clockwise turn


2. a =  [3, 3] ,  b= [-2, 2]  , c= [0, 0]

     ( -2 - 3 ) * ( 0 - 3 ) - ( 0 - 3 ) * ( 2 - 3 )
     Ans=  12
     its counter clockwise turn

3. a =  [3, 3] ,  b= [-2, 2]  , c= [1, 1]

     ( -2 - 3 ) * ( 1 - 3 ) - ( 1 - 3 ) * ( 2 - 3 )
     Ans=  8
     its counter closckwise turn

4. a =  [3, 3] ,  b= [-2, 2]  , c= [5, 2]
     ( -2 - 3 ) * ( 2 - 3 ) - ( 5 - 3 ) * ( 2 - 3 )
  Ans=  7
its counter closckwise turn
```

<u>Question # 12</u> **[2+2 =4 marks]**

Compute time complexity :

(a)

This is O($\boldsymbol{Sqrt(n)}$)

Although there is only one loop but at each iteration of while loop , "i" is incremented and then added to previous value of "s" which makes it **O($\boldsymbol{Sqrt(n)}$)**

(b)

This is *O*(n*)*

Only first for loop will run (n/3) times so ***O(n)***

For each iteration of first for loop, second for loop will run for one time only (as j will be equal to n in second for loop in first iteration and j<n condition will become false) and second for loop will terminate after its every first iteration.

It will also not go in the inner most while loop as while condition (j<=1) will never be true .