**Aim :- Solution of 1-1Assignment Problem.**

Algorithm 1:-

Backtracking Solution

Explores each Solution and If obtained Solution has Better assignment then update the final solution.

Either We can assign a user an Interval from its favourable intervals or left it unassigned and repeat the same for Next User.

This can be achieved either by Iteratively or recursively.

Recursive solution is easier to implement and backtrack.

Description of Code:-

We have one array UserAssigned to keep track of interval assigned to that user and is set as –1 if not assigned any interval.

We have another array IntervalAssigned to keep track of user assigned to that interval and is set as –1 if not assigned any user.

We have vector of vector of pair of integers (User and assigned Interval) to store the various obtained.

Whenever we arrive at base case of recurssion , we have a solution and we pop previous solution having less assignment than the current solution.

After the algorithm terminates we obtained various solution.

We sort the solution vectot on the basis of assignments (Size of Solution).

We print top 4 solutions.

Tracing the Algorithm :-

Consider the simple case

A-I1,I2

B-I1,I2

First A wil be left unassigned

->B is left unassigned and obtained solution would be empty.

->B will be assigned I1 and solution would be (B-I1).

-> B will be assigned I2 and solution would be (B-I2).

Now A will be Assigned I1

->B will left unassigned and obtained solution is (A-I1).

->B will assigned I2 and solution obtained is (A-I1,B-I2).

Now A will be assigned to I2

->B will be left unassigned and obtained solution is (A-I2).

-> B will be assigned to I1 and the obtained solution is (A-I2,B-I1).

Last step is to print the top 4 solution based on solution size.

Result :-

(A-I1,B-I2) (A-I2,B-I1) (A-I1) (A-I2)

Efficiency of Algorithm :-

Consider the worst case

Nouser=Nointervals=n.

Every interval is favourable for every User

Time Complexity:-

O(n!)

Space Complexity

O(n!)

Pros of Algorithm:-

1) Explores every possible assignement.

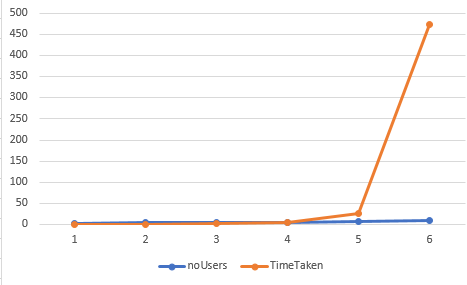
2) Best results for nouser<=4 and noInterval<=4.

Cons of Algorithm :-

1) Time complexity is very poor and for larger inputs very slow solutions.

2)Solution won’t work nouser>9 as n! Will cross the range of array and run time error will generated.

Graph of Time Taken vs Input(noUsers=noInterval=n).



Algorithm 2:-

This Problem can be efficiently by Graph flow Algorithm (Ford Fulkerson Algorithm).

We can construct a graph having one source node ,one destination node , noUsers number of nodes each represent one user and noIntervals number of nodes representing each interval.

We can construct unidirectional edges having weight 1 from source node to every node representing a user, from every node representing interval to destination node and from every node representing a user to nodes representing its favourable intervals.

Note :-Weight 1 refers that atmost 1 intervaL can be assigned to any user.

Now we can Use Ford Fulkerson Algorithm to find maximum assignment.

Description of ALgorithm:-

Construct a residual list as same as of adjacency list.

Try to find a path from source vertex to destination vertex having non zero flow(minimum weight in the path).

If we find non zero flow then from all the forward edges in path substract the flow and to all the backward edges in path add flow value.

This process is called Augmentation and new forward weights are Residual capacity.

Backward edges represent how much flow we can undo .

Repeat the same untill zero flow is obtained.

Implimetation :-

Edmond Karp Algorithm:-

This implimenation uses a breath first search to find the path from source to destination and parent of each node is stored in a array in the discovered path.

Whenever we find non zero path augment it.

The Total flow (assignments) can be obtained as sum of flows in every iteration.

Constructing Solution ?

All those edges having forward weight 1 initially and now weight 0 are assignments.

Thus We got one optimal Solution.

How to more Solutions ?

We can change the graph in following way

1)By reversing the order of Users.

2)By reversing the order of Intervals.

3)By reversing the order of Users and Intervals both.

We will 4 solutions but there may repitition.

Remove Duplicate solutions if any .

If number of solution are less than 4 now

We can now create another destination

Weight from initial destination to final destination should be initialised with a very high value.

After every solution we should change the weight to noAssignments-1 as we are interested in finding the solution having assignments less than previous by 1.

As soon we have 4 solutions we can terminate the algorithm.

Tracing the Algorithm:-

Consider the example

U1-I1,I2

U2-I2

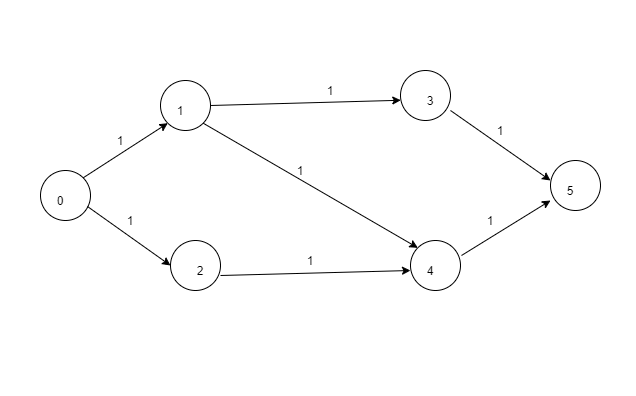


Fig1

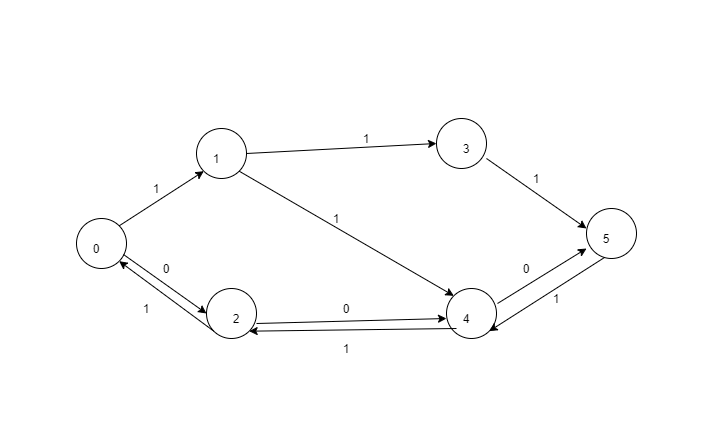
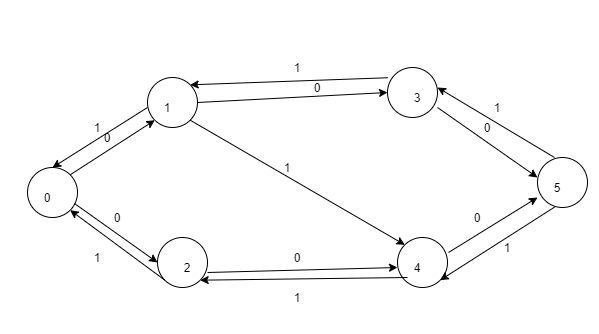


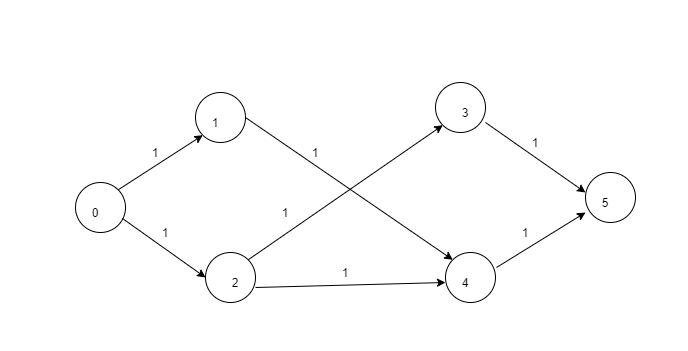
Fig 2



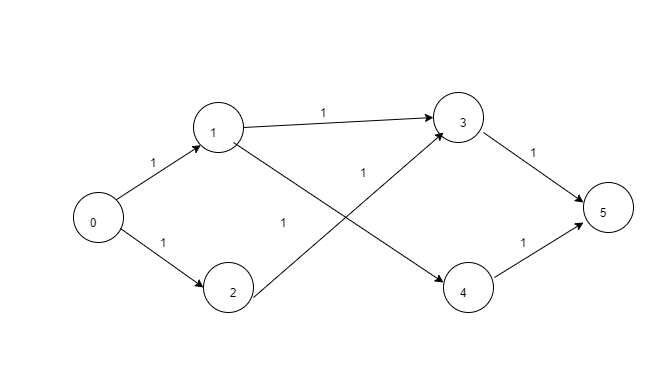
**Fig 3**

Graph can be changed by

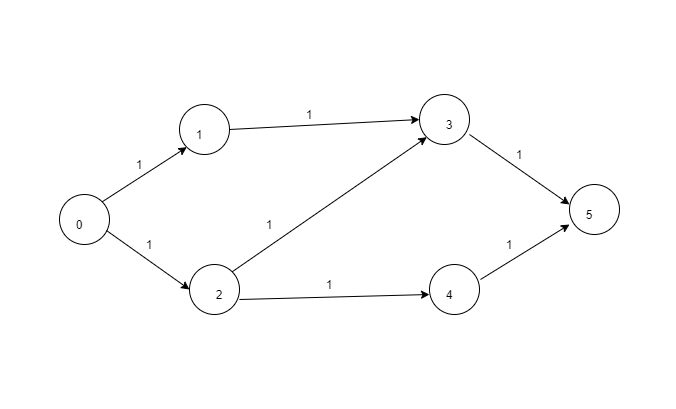
1)User order reversing:-



2)Interval reversing :-



3)Both User and Interval Reversing:-



Efficiency of Algorithm :-

Let the noUsers=n1 noInterval=n2

Time Complexity:-

Adjacency Matrix Implimentation O(n1\*n2^3).

Adjacency List Implimentation O(n1\*n2^2).

Space Complexity:-

Adjacency Matrix implimentation O((n1+n2)^2).

Adjacency List implimentation O(n1\*n2).

Pros of Algo :-

1)Solution is very efficient for user <=500 solution can be obtanied within 1sec.

2)Space is optimised using Adjacency list implimentation.

Cons of Algorithm:-

1)For finding more than 1 solution we have to change the Graph.

2)Doesn’t cover all possiblity only 4 are covered which may produce duplicate solutions.

Graph of Time Taken vs no of Users:-

