# Aim :- Assignment of intervals to each user in most optimal way.

# Backtracking Algorithm:-Exploring every possible assignment.

# Intuition :- Trying every possible combination and if total assignments are more total assignments are more previous assignments then update the solution vector.

# Description of algorithm :-

# Take one array for Users denoting Interval to which it is assigned or -1 if not assigned any.

# Take one array for interval denoting User to which it is assigned or -1 if not assigned any.

# For each user either we can assign it any interval from the favourable intervals or left it unassigned and do the same for each user recursively.

# And backtrack when solution is explored to explore other solutions.

# Try every possible assignment and when solution is obtained update the solution vector depending on the number of assignments .

# Code in Cpp:-

|  |
| --- |
| #include <bits/stdc++.h>  **using** **namespace** std;  **const** **int** N=**1e2**+**5**;  vector<vector<**int**>> FeasibleInterVals(N);  vector<**int**> UserAssigned(N,-**1**);  vector<**int**> InterValAssigned(N,-**1**);  vector<vector<pair<**int**,**int**>>> Solution;  **void** **assignInterVals**(**int** userId, **int** noUser)  { *//termination condition arrived and store the solution*  **if**(userId==noUser+**1**)  {  vector<pair<**int**,**int**>> PossibleSolution;  **for**(**int** user=**1**;user<=noUser;user++)  {  **if**(UserAssigned[user]>**0**)  PossibleSolution.push\_back(make\_pair(user,UserAssigned[user]));  }  Solution.push\_back(PossibleSolution);  *//deleting smaller solutions*  **for**(**auto** oneSolution =Solution.begin();oneSolution!=Solution.end();oneSolution++)  {  **if**(oneSolution->size()<PossibleSolution.size())  {  Solution.erase(oneSolution);  oneSolution--;  }  }  **return**;  }  *//excluding any assignment*  assignInterVals(userId+**1**,noUser);  **for**(**int** interval : FeasibleInterVals[userId])  { *//already assigend hence continue*  **if**(InterValAssigned[interval]!=-**1**) **continue**;  UserAssigned[userId]=interval;  InterValAssigned[interval]=userId;  assignInterVals(userId+**1**,noUser);  *//backtracking*  UserAssigned[userId]=-**1**;  InterValAssigned[interval]=-**1**;  }  }  **bool** **operator**<(vector<pair<**int**,**int**> >& p, vector<pair<**int**,**int**> > & q)  {  **return** (p.size()<q.size());  }  **void** PrintSolution()  { *//removing duplicate Solutions*  **auto** iterAtor=unique(Solution.begin(),Solution.end());  Solution.resize(iterAtor-Solution.begin());  *//Printing the Optimal Solutions*  **for**(**int** ithSolution=**0**;ithSolution <min(**4**,(**int**)Solution.size());ithSolution++)  {  vector<pair<**int**,**int**>> oneSolution =Solution[ithSolution];  sort(oneSolution.begin(),oneSolution.begin());    **for**(**int** user=**0**;user<oneSolution.size();user++)  {  cout<<oneSolution[user].first<<"th user is assigned "<<oneSolution[user].second<<"th interval"<<endl;  }  cout<<endl;  }  }  **int** main()  {    **time\_t** start, end;  time(&start);    #ifndef ONLINE\_JUDGE    freopen("input.txt", "r", stdin);    freopen("output.txt", "w", stdout);    #endif  **int** noUser;  *//enter the number of users*  cin >> noUser;  **int** noInterval;  *//enter the number of intervals*  cin >> noInterval;  **for**(**int** user=**1**;user<=noUser;user++)  {    *//enter the number of favaroble intervals of ith user*  **int** noFeasibleIntervals; cin >> noFeasibleIntervals;  **while**(noFeasibleIntervals--)  {  *//enter the interval for ith user*  **int** ithFavourableInterval;  cin >> ithFavourableInterval;  FeasibleInterVals[user].push\_back(ithFavourableInterval);  }  }  assignInterVals(**1**,noUser);    PrintSolution();  time(&end);  **double** time\_taken = **double**(end - start);  cout << "Time taken by program is : " << fixed << time\_taken << setprecision(**5**);  cout << " sec " << endl;  **return** **0**;  } |

# Tracing the Algorithm:-

# Example:-

# U1-I1,I2,I3

# U2-I1,I2

# U1 left unassigned->U2 left unassigned and base condition is reached and total assignments is 0 and solution vector is empty.

# 2)U1 is assigned I1 and U2 is Left unassigned and base case is reached and obtained solution is (u1-I1) and new solution vector is ((u1,-I1)).

# 3) U1 is assigned I1 and U2 is assigned I2and obtained solution is(u1-I1,U2,-I2) and solution vector is updated as ((u1-I1,U2-I2)).

# 4) U1 is assigned I2 and U2 is left unassigned and obtained solution is (u1-I2) and solution vector is ((u1-I1,U2-I2)).

# 5)U1 is assigned I2 and U2 is assigned I1 and solution obtained (U1-I2,U2-I1) and the solution vector is ((u1-I1,U2-I2), (u1-I2,U2-I1)).

# 6)U1 is assigned I3 and U2 is left unassigned and the solution generated is U1-I3 and solution vector is ((u1-I1,U2-I2), (u1-I2,U2-I1)).

# 7)U1 is assigned I3 and U2 is Assigned I1 and the solution generated is ((U1-I3),(U2-I1)) and solution vector is ((u1-I1,U2-I2), (u1-I2,U2-I1)),(U1-I3,U2-I1)).

# 8)U1 is assigned I3 and U2 is Assigned I2 and solution generated is (U1-I3,U2-I2) and solution vector is ((U1-I1,U2-I2), (U1-I2,U2-I1)),(U1-I3,U2-I1),(U1-I3,U2-I2)).

# Output of algorithm :-

# Solution 1 is U1-I1,U2-I2.

# Solution 2 is U1-I2,U2-I1.

# Solution 3is U1-I3,U2-I1.

# Solution 4 is U1-I3,U2-I2.

# Pros of Algorithm:-

# 1)Exlpores every possible assignments .

# 2)Best Solution for smaller inputs <=5 users and <=Intervals.

# Cons of Algorithm:-

# 1) Time complexity of solution is very bad o(noUsers!).

# 2)Solution won`t work for larger inputs >=10