Bitmask DP

Assignment Problem

Your task will be to calculate number of different assignments of n different topics to n students such that everybody gets exactly one topic he likes.

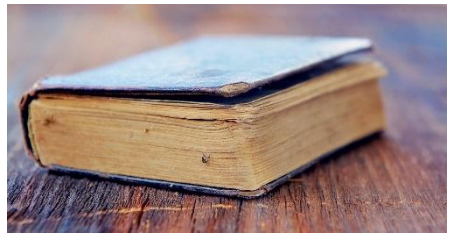
Input is like grid

At any i th cell value 0 means that student does not likes i th topic.

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

At any i th cell value 1 means that student likes i th topic

Think of backtracking solution first

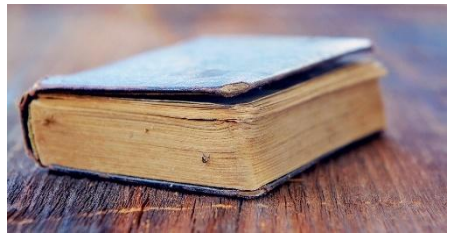


Topic 1

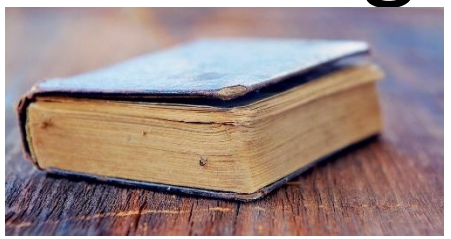
Student 1

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first



Topic 1



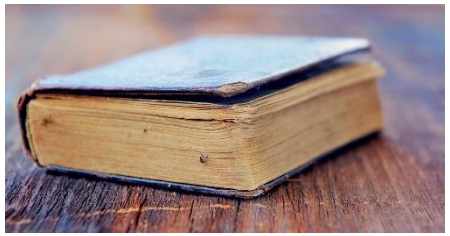
Topic 2

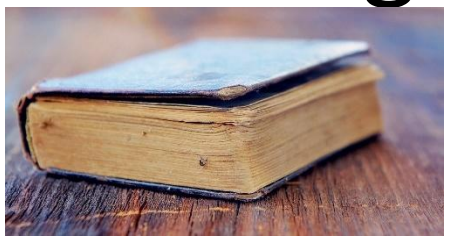
Student 1

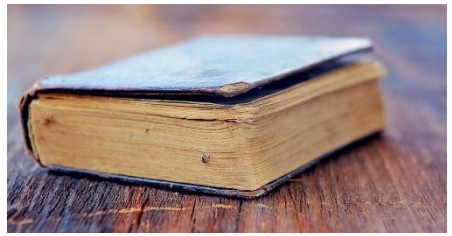
Student 2

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first







Condition Satisfied

Topic 1

Topic 2

Topic 3

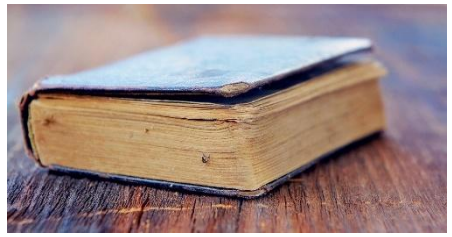
Student 1

Student 2

Student 3

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first

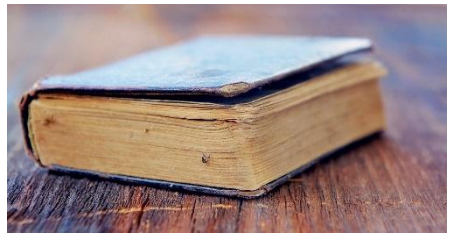


Topic 1

Student 2

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first

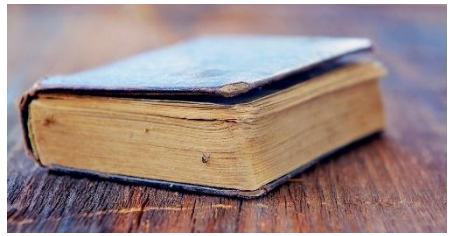


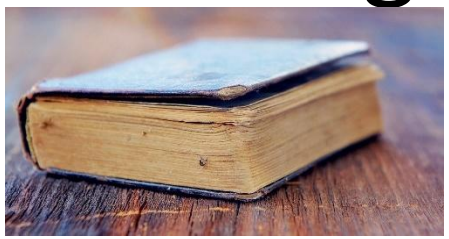
Topic 1

Student 3

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first





Topic 1

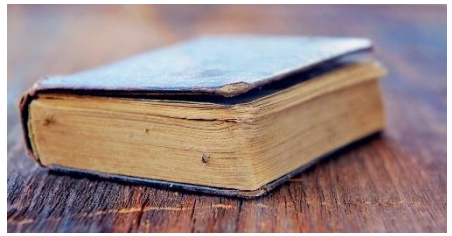
Topic 2

Student 3

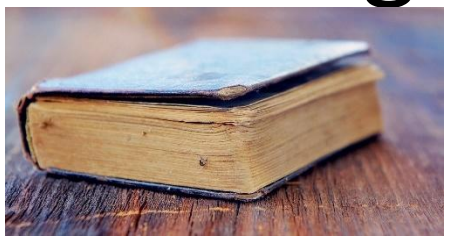
Student 1

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first



Topic 1



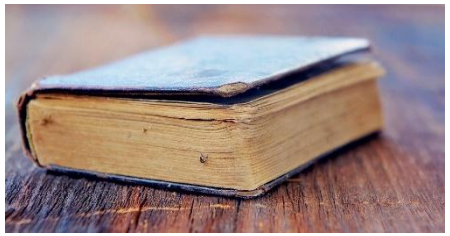
Topic 2

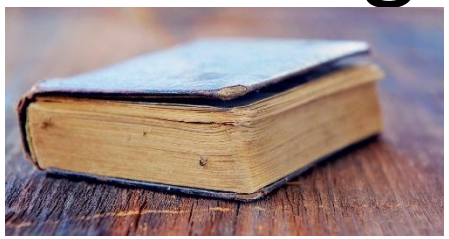
Student 3

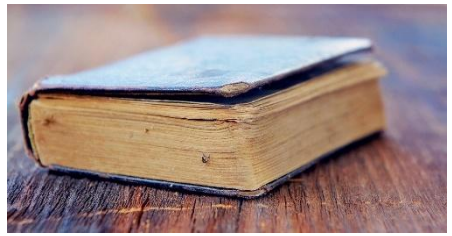
Student 2

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first







Condition Satisfied

Topic 1

Topic 2

Topic 3

Student 3

Student 2

Student 1

Topic ID → / Student ID ↓	1	2	3
1	1	0	1
2	0	1	1
3	1	1	1

Think of backtracking solution first

```
// student vector will be initialised  by 0 initialy
// student[k] = 0 means kth student has not been assigned some topic
// student[k] = 1 means kth student has been assigned some topic
int total_topics,total_students;
int solve(int i,vector<int> &student,vector<vector<int> > & likes)
{
    if(i==total_topics+1)
        return 1;
    int ans=0;
    for(int k=1;k<=total_students;k++)
    {
        if(student[k]==0&&likes[k][i]==1)
        {
            student[k]=1;
            ans+=solve(i+1,student,likes);
            student[k]=0;
        }
    }
    return ans;
}
```

Working of mask is same as working of student array in previous code

Total number of set bits in mask variable denotes number of topics assigned

If i th bit in mask is set (means =1) then it means that i th student has been assigned the task.

__builtin_popcount(mask)
This function returns total number of set bits in mask. This is an inbuilt function of C++.

```
ll total_topics,likes[21][21];

ll solve(ll mask)
{
    ll current_topic_number = __builtin_popcount(mask);
    if(current_topic_number==total_topics) return 1;

    ll count=0;
    for(ll j=1;j<=n;j++)
    {
        if(!(mask&(1LL<<j))&&likes[j][current_topic_number]==1)
        {
            ll temp=mask;
            mask=mask|(1LL<<j);
            count+=solve(mask);
            mask=temp;
        }
    }

    return count;
}
```

```cpp
ll total_topics,likes[21][21];
ll dp[1100000]; //---------------
ll solve(ll mask)
{
    ll current_topic_number = __builtin_popcount(mask);
    if(current_topic_number==total_topics) return 1;
    if(dp[mask]!=-1) return dp[mask]; // --------------------
    ll count=0;
    for(ll j=1;j<=n;j++)
    {
        if(!(mask&(1LL<<j))&&likes[j][current_topic_number]==1)
        {
            ll temp=mask;
            mask=mask|(1LL<<j);
            count+=solve(mask);
            mask=temp;
        }
    }
    dp[mask]=count; // ------------------
    return count;
}
```