Matching

There are N men and N women, both numbered $1,2,\ldots,N$.

For each
i,j $(1 \leq i, j \leq N)$, the compatibility of Man i and Woman j is given as an integer $A_{i,j}$.
If $A_{i,j} = 1$, Man i and Woman j are compatible;
if $A_{i,j} = 0$, they are not.

Taro is trying to make N pairs,
each consisting of a man and a woman who are compatible.

Here, each man and each woman must belong to exactly one pair.

Find the number of ways in which Taro can make N pairs, modulo $10^9 + 7$.

Constraints

All values in input are integers.
$1 \leq N \leq 21$
$A_{i,j}$ is 0 or 1.

Example

Sample Input 1

```
3
0 1 1
1 0 1
1 1 1
```

Sample Output 1

```
3
```

There are three ways to make pairs, as follows.
(i,j) denotes a pair of Man i and Woman j:

(1,2),(2,1),(3,3)
(1,2),(2,3),(3,1)
(1,3),(2,1),(3,2)

Problem Breakdown

Lets understand the problem first

We have:

- N men: 1..N
- N women: 1..N
- Compatibility matrix $a[i][j]$:
  - $a[i][j] = 1$ → man i can be matched with woman j
  - $a[i][j] = 0$ → they cannot

We want to count the number of perfect matchings:

- Every man is matched with exactly one woman.
- Every woman is matched with exactly one man.
- Each pair in the matching must be compatible.

We need the count modulo $10^9 + 7$.

This is basically:

In how many ways can we assign each man a compatible woman, such that no woman is used twice?

This is identical in shape to the assignment DP we discussed, except instead of minimizing cost we are counting ways.

Bitmask design

We'll:

- Work with 0-based indices in code:
  - Men: 0..N-1
  - Women: 0..N-1
- Use an integer mask of N bits to represent which women are already chosen.

Interpretation of mask:

- Bit j is 1 → Woman j is already matched.
- Bit j is 0 → Woman j is still free.

Example for $N = 3$:

- mask = 0b101 (5 in decimal)
  - bit 0 = 1 → woman 0 used
  - bit 1 = 0 → woman 1 free
  - bit 2 = 1 → woman 2 used
  - So, used women = {0, 2}.

DP state definition

We use the classic pattern:

Let dp[mask] = number of ways to match the first k men (men 0..k-1) with exactly the women in mask, where k = popcount(mask).

k = __builtin_popcount(mask) = number of set bits in mask = number of women already assigned.

So it's natural to say:
- We've already matched men 0,1,...,k-1 to those k women.

So:

dp[0] = 1.
- no women used = no men matched = 1 "empty" way.

Our answer will be:

dp[(1 << N) - 1]
(mask where all N bits are 1, i.e., all women are used → all N men matched).

Transition: how do we move from one state to another?

From a given mask:

1. Let i = popcount(mask) → this is the index of the next man to match.
   - We have already matched men $0..i-1$.
   - Next, we want to match man i.
2. For this man i, we look at all women j such that:
   - $a[i][j] == 1$ (compatible), and
   - bit j is 0 in mask (woman j is free).
3. For each such j:
   - We create a new mask: newMask = mask | (1 << j) (mark woman j as used).
   - We add dp[mask] ways to dp[newMask].

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MOD = 1e9 + 7;

int addmod(int a, int b) {
    a += b;
    if (a >= MOD) a -= MOD;
    return a;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int N;
    cin >> N;

    // a[i][j]: compatibility of man i and woman j
    // using 0-based indexing
    vector<vector<int>> a(N, vector<int>(N));
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cin >> a[i][j];
        }
    }

    int totalMasks = 1 << N;
    vector<int> dp(totalMasks, 0);

    // Base case: no men matched, no women used
    dp[0] = 1;

    // Iterate over all masks
    for (int mask = 0; mask < totalMasks; mask++) {
        // How many men are already matched?
        int i = __builtin_popcount(mask);

        // If i == N, we've already matched all men, no need to continue
        if (i >= N) continue;

        // If there are 0 ways to reach this mask, skip
        if (dp[mask] == 0) continue;

        // Try to match man i with some compatible woman j
        for (int j = 0; j < N; j++) {
            // If woman j is not used yet AND compatible with man i
            if (!(mask & (1 << j)) && a[i][j] == 1) {
                int newMask = mask | (1 << j);
                dp[newMask] = addmod(dp[newMask], dp[mask]);
            }
        }
    }

    // Answer: all N men matched with all N women (all bits set)
    int fullMask = (1 << N) - 1;
    cout << dp[fullMask] << '\n';

    return 0;
}
```