状压dp练习2

人员

左子毅、朱奕鸣、杨洋、刘子淇、赵清航、周子航、于迦浩 到课

作业检查

左子毅 完成

朱奕鸣 未完成

杨洋 完成

刘子淇 完成

赵清航 未完成

周子航 完成

于迦浩 完成

作业

https://www.luogu.com.cn/contest/170764

A、B 2道题目必做

课堂表现

同学们代码实现能力都存在一定的问题，课下一定要好好练习刷题题单

课堂内容

Minimax Problem

```
题目要求最小值最大，一眼二分，二分最终最小的答案
因为m非常小，check时可以把m列的状态通过状态压缩的方式映射成一个数
开一个f数组，记录一个状态对应的行
之后利用256*256的时间复杂度统计是否有合法解即可
```

```cpp
#include <bits/stdc++.h>

using namespace std;

const int maxn = 3e5 + 5, N = 8;
int w[maxn][N];

bool check(int n, int m, int mid, int& ansL, int& ansR) {
  int f[1<<N];
  memset(f, 0, sizeof(f));
  for (int i = 1; i <= n; ++i) {
    int res = 0;
    for (int j = 0; j < m; ++j) {
      if (w[i][j] >= mid) res |= (1<<j);
    }
    f[res] = i;
  }

  for (int i = 0; i < (1<<m); ++i) {
    for (int j = 0; j < (1<<m); ++j) {
      if ((i|j)==(1<<m)-1 && f[i] && f[j]) {
        ansL = f[i], ansR = f[j];
        return true;
      }
    }
  }
  return false;
}

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) {
    for (int j = 0; j < m; ++j) cin >> w[i][j];
  }

  int l = 0, r = 1e9;
  int ansL = 1, ansR = 1;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (check(n, m, mid, ansL, ansR)) l = mid+1;
    else r = mid-1;
  }
  cout << ansL << " " << ansR << endl;
  return 0;
}
```

Petya and Spiders

定义状态 f[i][j][k]:
第 i 行状态为 j，第 i-1 行状态为 k 时，前 i-1 行最多有多少空白格
转移过程：f[i][j][k] <- f[i-1][k][l] + sum(k)
sum(k): k 这个状态代表第 i-1 行有几个空白格
转移条件：对于 k 每个为 0 的二进制位，要求它的 左边、右边、上边、下边 至少存在一个 1

```cpp
#include <bits/stdc++.h>

using namespace std;

const int N = 40 + 5, M = 6;
int f[N][1<<M][1<<M];
// f[i, j, k]:  i ~ j,  i-1 ~ k

bool check(int n, int m, int i, int j, int k, int l) {
  // i: j  i-1: k  i-2: l
  if (i-2==0 && l) return false;
  if (i==n+1 && j) return false;
  for (int p = 0; p < m; ++p) {
    if (!((k>>p) & 1)) {
      if ((j>>p) & 1) continue;
      if ((l>>p) & 1) continue;
      if (p>0 && ((k>>(p-1)) & 1)) continue;
      if (p<m && ((k>>(p+1)) & 1)) continue;
      return false;
    }
  }
  return true;
}

int sum(int k, int m) {
  int res = 0;
  for (int i = 0; i < m; ++i) {
    if (!((k>>i) & 1)) ++res;
  }
  return res;
}

int main()
{
  int n, m; cin >> n >> m;
  if (n < m) swap(n, m);

  int res = 0;
  for (int i = 2; i <= n+1; ++i) {
    for (int j = 0; j < (1<<m); ++j) {
      for (int k = 0; k < (1<<m); ++k) {
        for (int l = 0; l < (1<<m); ++l) {
          // i: j  i-1: k  i-2: l
          if (check(n, m, i, j, k, l)) {
            f[i][j][k] = max(f[i][j][k], f[i-1][k][l] + sum(k, m));
            if (i == n+1) res = max(res, f[i][j][k]);
          }
        }
      }
    }
```

```
    }

    cout << res << endl;
    return 0;
}
```