一、代码及注释

1.石子合并

```cpp
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
using ll = long long;

const int N = 205;

int f[N][N];    // f[i][j]: 代表合并区间[i, j]石子的最小花费
int a[N], p[N]; // p为前缀和，快速查询区间和

void solve() {
    int n; cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        p[i] = p[i - 1] + a[i];
    }
    // 由于是环形放置，可以构造两倍长数组，遍历f[i][i+n-1] (1<=i<=n) 取最小即可
    for (int i = n + 1; i <= 2 * n; i++) {
        p[i] = p[i - 1] + a[i - n];
    }

    memset(f, 0x3f, sizeof f);
    // 初始化长度为0的区间费用为0，其他为无穷大
    for (int i = 1; i <= 2 * n; i++) f[i][i] = 0;

    for (int i = 2; i <= n; i++) {  // 第一维枚举长度，长度由小增大，保证递推的正确性
        for (int j = 1; j <= 2 * n - i + 1; j++) {  // 枚举左边界
            int r = i + j - 1;                      // 计算右边界
            for (int k = j; k < r; k++) {           // 枚举分割线（当前状态可能的转移）
                f[j][r] = min(f[j][r], f[j][k] + f[k + 1][r] + p[r] - p[j - 1]);
            }
        }
    }

    int ans = 0x3f3f3f3f;
    for (int i = 1; i <= n; i++) {
        ans = min(ans, f[i][i + n - 1]);
    }
    cout << ans << '\n';

    // 最大值类同
    memset(f, 0, sizeof f);

    for (int i = 2; i <= n; i++) {
        for (int j = 1; j <= 2 * n - i + 1; j++) {  // 枚举左端点
            int r = i + j - 1;
            for (int k = j; k < r; k++) {
                f[j][r] = max(f[j][r], f[j][k] + f[k + 1][r] + p[r] - p[j - 1]);
            }
```

```
            }
        }
        ans = 0;
        for (int i = 1; i <= n; i++) {
            ans = max(ans, f[i][i + n - 1]);
        }
        cout << ans << '\n';
    }

    int main() {
        ios::sync_with_stdio(false);
        cin.tie(nullptr);
        solve();
        return 0;
    }
```

2.方格填充

```cpp
#include<iostream>
#include<algorithm>
using namespace std;
using ll = long long;

ll f[15][(1 << 15)];
// f[i][j]: 前i行且第i行的状态为j时的方案数
// 注意i * w为奇数时没有实际意义，最后一行不全部为0
int h, w;

// 检查是否连续的‘0’均为偶数个
bool chk(int a) {
    int cnt = 0;
    for (int i = 0; i < w; i++) {
        if (a & (1 << i)) {
            if (cnt & 1) return false;
            cnt = 0;
        }
        else cnt++;
    }
    if (cnt & 1) return false;
    return true;
}

void solve() {

    cin >> h >> w;

    // 只有奇数个方格显然无解
    if (h * w % 2) {
        cout << 0 << '\n';
        return;
    }

    // 假设第0行的状态为0，即所有长方形水平放置
    f[0][0] = 1;
    for (int i = 1; i <= h; i++) {
        for (int j = 0; j < (1 << w); j++) {          // 枚举当前行状态
            for (int k = 0; k < (1 << w); k++) {      // 枚举上一行状态
                if (j & k) continue;                  // 出现上1下1的情况不合法
                if (!chk(j | k)) continue;            // 有连续奇数'0'状态不合法
                f[i][j] += f[i - 1][k];
            }
        }
    }

    cout << f[h][0] << '\n';
}

int main() {
```

```
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
    return 0;
}
```