奖牌线	题数
铜牌	4题
银牌	5题手速/6题
金牌	7题

本次比赛四题选自2025 ICPC Shaanxi National Invitational的L，G，J，M四题，除去A题的大模拟外，其他均按照通过率排序，个人认为的难度应该为A<B=D<J，A签到，直接纯暴力即可，B和D为模板题，学过基本就很容易想到如何解决，J的结论性很强，主要难在证明，如果靠暴力打表可能更容易看出规律。

题解报告

A. 衣服打包

注意到 n 很小，直接暴力所有衣服的组合，依次验证即可。

```cpp
#include <bits/stdc++.h>
using namespace std;
vector<vector<int>> que{{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}};
void solve()
{
    vector<int> a(4);
    int n, m;
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    cin >> m;
    vector<int> t(105), x(105);
    for (int i = 1; i <= m; i++)
        cin >> t[i] >> x[i];
    for (auto &it : que)
    {
        for (auto &jt : it)
        {
            if (jt > n)
                goto next;
        }
        for (int i = 1; i <= m; i++)
        {
            if (t[i] == 1)
            {
                int res = 0;
                for (auto &jt : it)
                    res += a[jt];
                if (res < x[i])
                    goto next;
            }
            else
            {
                for (auto &jt : it)
                {
                    if (a[jt] <= x[i])
                        goto nextloop;
                }
                goto next;
            nextloop:;
            }
        }
        cout << it.size() << '\n';
        return;
    next:;
    }
```

```
        cout << -1 << '\n';
        return;
    }
    signed main()
    {
        cin.tie(nullptr)->sync_with_stdio(false);
        solve();
    }
```

B. 内卷

对于最终的 $n \times m$ 的序列，考虑 m 的每一位在序列上分别匹配0和1的个数，取较大值作为答案的贡献即可。匹配过程可以用倍增优化。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define IOS                         \
    ios_base::sync_with_stdio(0); \
    cin.tie(0);                   \
    cout.tie(0);
#define int long long
typedef vector<vector<int>> mat;
using pii = pair<int, int>;
using pdd = pair<double, double>;
constexpr int N = 1e5 + 5;
int f[N][20], g[N][20];
using ll = long long;
void solve()
{
    ll n, m;
    cin >> n >> m;
    string s;
    cin >> s;
    for (int i = 0; i < n; i++)
    {
        f[i][0] = s[(i + m) % n] == '1';
        g[i][0] = s[(i + m) % n] == '0';
    }
    for (int k = 1; k < 20; k++)
    {
        for (int i = 0; i < n; i++)
        {
            f[i][k] = f[i][k - 1] + f[(i + (1ll << (k - 1)) * m) % n][k - 1];
            g[i][k] = g[i][k - 1] + g[(i + (1ll << (k - 1)) * m) % n][k - 1];
        }
    }
    ll ans = 0;
    for (int i = 0; i < m; i++)
    {
        ll p = i;
        ll cnt0 = s[i % n] == '0', cnt1 = s[i % n] == '1';
        for (int k = 19; k >= 0; k--)
        {
            if (p + (1ll << k) * m >= n * m)
                continue;
            cnt1 += f[p % n][k];
            cnt0 += g[p % n][k];
            p += (1ll << k) * m;
        }
        ans += max(cnt0, cnt1);
```

```
    }
    cout << ans << '\n';
}
signed main()
{
    IOS;
    int _ = 1;
    // cin >> _;
    while (_--)
    {
        solve();
    }
    return 0;
}
```

Bonus

本题有 $O(n)$ 的做法，可以自行思考。

C. 信息传递

本题最简单的方法是暴力打表猜结论。

结论为：

- 若 n 为奇数，3的数量对6的余数为2或者3为 `No` 否则 `Yes` 。
- 若 n 为偶数，3的数量对6的余数为0或者5为 `No` ，否则 `Yes` 。

给出简单证明：

考虑结构 `23 (x) (y)` 不难发现，Y这个位置一定是不可达的。因此，**2和3的交汇处只能出现一次**，因此构造方案为所有的2和3全部摆在一起，形成 `222...23333` 这样的结构，由于 `Y` 这个位置一定无法到达，因此其一定是起点，以该位置作为起点模拟一遍，检查是否能到达所有数即可。

最后也能得出上述结论。

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int ar[N];
void solve()
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> ar[i];
    if (n == 1)
    {
        cout << "Yes\n";
        return;
    }
    int t3 = 0;
    for (int i = 1; i <= n; i++)
    {
        if (ar[i] == 3)
            ++t3;
    }
    if (n % 2 == 1)
    {
        t3 %= 6;
        if (t3 == 2 || t3 == 3)
        {
            cout << "No\n";
        }
        else
            cout << "Yes\n";
    }
    else
    {
        t3 %= 6;
        if (t3 == 0 || t3 == 5)
        {
            cout << "No\n";
        }
        else
            cout << "Yes\n";
    }
    return;
}
signed main()
{
    cin.tie(nullptr)->sync_with_stdio(false);
```

```
    int t = 1;
    cin >> t;
    while (t--)
        solve();
}
```

D. 跨国旅行

由于每个点都会被距离它最近的“首都”染色，且相同颜色的点一定都在一起，因此可以考虑把所有的首都加入到队列中做一次bfs，求出所有点的颜色后，将相同颜色的点缩成一个点，重新建图（新图一定也是一棵树），询问就变成了新图上两点之前的简单路径的点数了，即简单路径长度+1，用lca求出即可。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define IOS                         \
    ios_base::sync_with_stdio(0); \
    cin.tie(0);                   \
    cout.tie(0);
constexpr int N = 2e5 + 5;

vector<int> edge[N], new_edge[N];

void solve()
{
    int n;
    cin >> n;
    for (int i = 1; i < n; i++)
    {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    queue<int> q;
    int color_idx = 0;
    vector<int> color(n + 1);
    for (int i = 1; i <= n; i++)
    {
        if (edge[i].size() > 2)
            q.push(i), color[i] = ++color_idx;
    }
    while (q.size())
    {
        auto u = q.front();
        q.pop();
        for (auto v : edge[u])
        {
            if (!color[v])
            {
                color[v] = color[u];
                q.push(v);
            }
        }
    }
    for (int u = 1; u <= n; u++)
    {
        for (auto v : edge[u])
        {
```

```
            if (color[u] != color[v])
            {
                new_edge[color[u]].push_back(color[v]);
                new_edge[color[v]].push_back(color[u]);
            }
        }
    }
    vector<vector<int>> f(color_idx + 1, vector<int>(20));
    vector<int> dep(color_idx + 1, 1e9);
    auto bfs = [&](int root = 1)
    {
        dep[root] = 1, dep[0] = 0;
        queue<int> q;
        q.push(root);
        while (q.size())
        {
            auto u = q.front();
            q.pop();
            for (auto v : new_edge[u])
            {
                if (dep[v] > dep[u] + 1)
                {
                    dep[v] = dep[u] + 1;
                    q.push(v);
                    f[v][0] = u;
                    for (int k = 1; k < 20; k++)
                    {
                        f[v][k] = f[f[v][k - 1]][k - 1];
                    }
                }
            }
        }
    };
    bfs();
    auto lca = [&](int a, int b)
    {
        if (dep[a] < dep[b])
        {
            swap(a, b);
        }
        for (int i = 19; i >= 0; i--)
        {
            if (dep[f[a][i]] >= dep[b])
                a = f[a][i];
        }
        if (a == b)
```

```
            return a;
        for (int i = 19; i >= 0; i--)
        {
            if (f[a][i] != f[b][i])
            {
                a = f[a][i];
                b = f[b][i];
            }
        }
        return f[a][0];
    };
    int Q;
    cin >> Q;
    while (Q--)
    {
        int u, v;
        cin >> u >> v;
        u = color[u], v = color[v];
        cout << dep[u] + dep[v] - 2 * dep[lca(u, v)] + 1 << '\n';
    }
}
signed main()
{
    IOS;
    int _ = 1;
    // cin >> _;
    while (_--)
    {
        solve();
    }
    return 0;
}
```