CSP 真题讲解

人员

刘锦轩、牛晓晨、隋天翼、王彦臻、史吉轩、卢新闻、战鹤文、卢炫佑、咸俊永、盛捷、彭贵晟 到课

本周作业

https://www.luogu.com.cn/contest/282038 的 A ~ F 题

https://www.luogu.com.cn/contest/282043 的 A C F 题

课堂表现

今天课上讲了 9 道题目, 讲的题比较多, 同学们课下一定要好好补补题, 要不然讲课的意义就没有了

课上如果有题目没完全掌握的, 可以课下看回放再听一听

课堂内容

P5657 [CSP-S2019] 格雷码

for 循环依次看每一位是 1 还是 0, 从最高位看到最低位, 从而确定最终结果

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

using namespace std;

typedef unsigned long long ULL;

int main()
{
  int n; ULL k; cin >> n >> k;
  for (int i = n-1; i >= 0; --i) {
    if ((k>>i) & 1) cout << 1, k = (1ULL<<i) - 1 - k%(1ULL<<i);
    else cout << 0;
  }
  cout << endl;
  return 0;
}
```

P5658 [CSP-S2019] 括号树

定义 f[i]: 以 i 这个点作为结尾点时, 一共有多少合法的括号串

f[i] 如何求, 维护一个栈, 在 dfs 的过程中, 找到点 i 对应的左括号 id, f[i] 的值即为 f[fa[id]] 的值 +1

如何求以 i 结尾时, 前面的 合法括号子串 的数量? -> i 前面所有点的 f 值加和, dfs 过程中可以维护

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

using namespace std;

typedef long long LL;
const int maxn = 5e5 + 5;
char s[maxn];
int fa[maxn];
vector<int> vec[maxn];
LL f[maxn];
stack<int> stk;
LL res = 0;

void dfs(int u, LL pNums) {
  int id = -1;

  if (s[u] == '(') stk.push(u);
  else if (!stk.empty()) {
    id = stk.top(); stk.pop(); f[u] = f[fa[id]]+1;
    pNums += f[u];
  }

  res ^= u*pNums;
  for (int i : vec[u]) dfs(i, pNums);

  if (s[u] == '(') stk.pop();
  else if (id != -1) stk.push(id);
}

int main()
{
  int n; cin >> n;
  cin >> (s+1);
  for (int i = 2; i <= n; ++i) {
    int x; cin >> x; fa[i] = x;
    vec[x].push_back(i);
  }

  dfs(1, 0);
  cout << res << endl;
  return 0;
}
```

P7075 [CSP-S2020] 儒略日

可以先暴力求得: 1600 年 1 月 1 日前的所有情况

对于 1600 年 1 月 1 日以后的情况, 可以以 400 为一个周期, 先确定在哪个周期中, 然后暴力往后循环找具体日期即可

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

using namespace std;

typedef long long LL;
const int maxn = 3e6 + 5;
struct node {
  int y, m, d;
  bool flag;
} w[maxn];

string get_string(node x) {
  string res = to_string(x.d) + " " + to_string(x.m) + " " + to_string(x.y);
  if (x.flag) res += " BC";
  return res;
}

bool isRun(int y) {
  if (y < 0) return (-y)%4==1;
  if (y < 1582) return y%4==0;
 return (y%400==0)||(y%100!=0&&y%4==0);
}

int days[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
bool check(int y, int m, int d) {
  int limit = days[m];
  if (isRun(y) && m==2) limit = 29;
  return d > limit;
}

int main()
{
  int id = 0;
  for (int y=-4713,m=1,d=1; y < 1600; ++id) {
    w[id] = {abs(y),m,d,y<0};

    ++d;
    if (check(y,m,d)) ++m, d=1;
    if (m == 13) ++y, m = 1;

    if (y == 0) y = 1;
    if (y==1582 && m==10 && d==5) d = 15;
  }

  int T; cin >> T;
  while (T -- ) {
    LL x; cin >> x;
    if (x < id) cout << get_string(w[x]) << endl;
    else {
      x -= id;
      int yy = 1600, mm = 1;

      int C = 400*365 + 97;
```

```
      int temp = x/C; yy += temp*400; x%=C;
      for (int i = yy; ; ++i) {
        int limit = (isRun(i) ? 366 : 365);
        if (x >= limit) x -= limit;
        else { yy = i; break; }
      }

      for (int i = 1; ; ++i) {
        int limit = days[i];
        if (isRun(yy) && i==2) limit = 29;
        if (x >= limit) x -= limit;
        else { mm = i; break; }
      }

      int dd = x+1;

      cout << get_string({yy,mm,dd,false}) << endl;
    }
  }
  return 0;
}
```

P7076 [CSP-S2020] 动物园

求出来哪些位可以选, 假设有 cnt 位可以选, 答案即为 2^cnt - n

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

using namespace std;

typedef unsigned long long ULL;

int main()
{
  int n, m, c, k; cin >> n >> m >> c >> k;

  ULL v1 = 0;
  for (int i = 1; i <= n; ++i) {
    ULL x; cin >> x; v1 |= x;
  }

  ULL v2 = 0;
  for (int i = 1; i <= m; ++i) {
    int p, q; cin >> p >> q; v2 |= (1ULL<<p);
  }

  int cnt = 0;
  for (int i = 0; i < k; ++i) {
    int c1 = (v1>>i)&1, c2 = (v2>>i)&1;
    if (c1 || !c2) ++cnt;
  }
```

```
  if (cnt == 64) {
    if (!n) cout << "18446744073709551616" << endl;
    else cout << 0ULL-n << endl;
  } else cout << (1ULL<<cnt) - n << endl;
  return 0;
}
```

P7913 [CSP-S 2021] 廊桥分配

先预处理只针对 国内/国际 航班, 如果给他们 i 个廊桥, 最多能有多少飞机抵达的信息

然后枚举左右廊桥的数量, 求最大值

如何预处理: 考虑给每架飞机用最小编号的廊桥, 看每架飞机需要用几号廊桥

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

using namespace std;

const int maxn = 1e5 + 5;
struct node {
  int l, r;
  bool operator < (const node& p) const { return l < p.l; }
} a[maxn], b[maxn];

struct Info {
  int id, tm;
  bool operator < (const Info& p) const { return tm < p.tm; }
  bool operator > (const Info& p) const { return tm > p.tm; }
};

int p1[maxn], p2[maxn];

void solve(int n, int m, node c[], int p[]) {
  sort(c+1, c+m+1);
  priority_queue<Info, vector<Info>, greater<Info>> q;
  set<int> s; for (int i = 1; i <= n; ++i) s.insert(i);

  for (int i = 1; i <= m; ++i) {
    int l = c[i].l, r = c[i].r;
    while (!q.empty() && q.top().tm<l) s.insert(q.top().id), q.pop();
    if (!s.empty()) {
      int id = *s.begin(); p[id]++; q.push({id, r}); s.erase(id);
    }
  }

  for (int i = 1; i <= n; ++i) p[i] += p[i-1];
}

int main()
```

```
{
  int n, m1, m2; cin >> n >> m1 >> m2;
  for (int i = 1; i <= m1; ++i) cin >> a[i].l >> a[i].r;
  for (int i = 1; i <= m2; ++i) cin >> b[i].l >> b[i].r;

  solve(n, m1, a, p1), solve(n, m2, b, p2);

  int res = 0;
  for (int i = 0; i <= n; ++i) res = max(res, p1[i]+p2[n-i]);
  cout << res << endl;
  return 0;
}
```

P7915 [CSP-S 2021] 回文

2 种情况, 一种是先选 1, 后选 pos1; 另一种是先选 2 * n, 后选 pos2 (pos1/pos2 是跟 a[1]/a[2 * n] 数值一样的位置)

可以统一考虑, 根据先选和后选的情况, 可以把数组分为两段, 第一段是 l1~l2, 第二段是 r2~r1

接下来每次选, 只有 4 种情况:

l1/r2 配对, l1/l2 配对, r1/r2 配对, r1/l2 配对

找字典序小的情况优先考虑即可

如果都不行, 输出 -1

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

using namespace std;

const int maxn = 1e6 + 5;
int w[maxn];

string solve(deque<int> dq1, deque<int> dq2, string s1, string s2) {
  while (!dq1.empty() || !dq2.empty()) {
    int l1 = -1, l2 = -2, r2 = -3, r1 = -4;
    if (!dq1.empty()) l1 = dq1.front(), l2 = dq1.back();
    if (!dq2.empty()) r2 = dq2.front(), r1 = dq2.back();

    if (l1==r2) s1 += "L", s2 += "R", dq1.pop_front(), dq2.pop_front();
    else if ((int)dq1.size()>=2 && l1==l2) s1 += "L", s2 += "L", dq1.pop_front(),
dq1.pop_back();
    else if ((int)dq2.size()>=2 && r1==r2) s1 += "R", s2 += "R", dq2.pop_back(),
dq2.pop_front();
    else if (r1==l2) s1 += "R", s2 += "L", dq2.pop_back(), dq1.pop_back();
    else return "-1";
  }

  reverse(s2.begin(), s2.end());
  return s1 + s2;
```

```
}

void solve() {
  int n; cin >> n;
  for (int i = 1; i <= 2*n; ++i) cin >> w[i];

  int pos1, pos2;
  for (int i = 1; i <= 2*n; ++i) {
    if (i!=1 && w[i]==w[1]) pos1 = i;
    if (i!=2*n && w[i]==w[2*n]) pos2 = i;
  }

  deque<int> dq1, dq2;
  for (int i = 2; i <= pos1-1; ++i) dq1.push_back(w[i]);
  for (int i = pos1+1; i <= 2*n; ++i) dq2.push_back(w[i]);
  string s = solve(dq1, dq2, "L", "L");
  if (s != "-1") { cout << s << endl; return; }

  dq1.clear(), dq2.clear();
  for (int i = 1; i <= pos2-1; ++i) dq1.push_back(w[i]);
  for (int i = pos2+1; i <= 2*n-1; ++i) dq2.push_back(w[i]);
  s = solve(dq1, dq2, "R", "L");
  if (s != "-1") { cout << s << endl; return; }

  cout << -1 << endl;
}

int main()
{
  int T; cin >> T;
  while (T -- ) solve();
  return 0;
}
```

P11362 [NOIP2024] 遗失的赋值

先对输入的数据进行 去重 和 判无解 操作

按照 c 值从小到大排序, 求每一段的合法的方案数

最前面一段的方案数是 v^2*(c[1]-1), 最后面一段的方案数是 v^2*(n-c[m])

中间 last ~ now 的方案数是 v^2*(now-last) - v^(now-last-1)*(v-1) 种方案

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
const int mod = 1e9 + 7;
struct node {
```

```
  int c, d;
  bool operator < (const node& p) const {
    if (c != p.c) return c < p.c;
    return d < p.d;
  }
  bool operator == (const node& p) const { return c==p.c && d==p.d; }
} w[maxn];

int qmod(int a, int k) {
  int res = 1;
  while (k) {
    if (k&1) res = (LL)res*a % mod;
    a = (LL)a*a % mod;
    k >>= 1;
  }
  return res;
}

void solve() {
  int n, m, v; cin >> n >> m >> v;
  for (int i = 1; i <= m; ++i) cin >> w[i].c >> w[i].d;
  sort(w+1, w+m+1);
  m = unique(w+1, w+m+1) - w - 1;

  for (int i = 2; i <= m; ++i) {
    if (w[i].c == w[i-1].c) { cout << 0 << endl; return; }
  }

  int res = 1;
  for (int i = 2; i <= m; ++i) {
    int last = w[i-1].c, now = w[i].c;
    int temp = (qmod(v,2*(now-last)) - (LL)qmod(v,now-last-1)*(v-1)%mod + mod) %
mod;
    res = (LL)res*temp % mod;
  }
  res = (LL)res * qmod(v,2*(w[1].c-1)) % mod * qmod(v, 2*(n-w[m].c)) % mod;
  cout << res << endl;
}

int main()
{
  int T; cin >> T;
  while (T -- ) solve();
  return 0;
}
```

P9869 [NOIP2023] 三值逻辑

f[i]: 先求出来 i 的值是依赖于哪个变量初值的

d[i]: 求 i 的值是跟初值一样还是跟初值相反

输入维护完之后, 按照关系进行建图

奇环中的点是 Unknown, 跟 n+2 相连的点也是 Unknown

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

using namespace std;

const int maxn = 1e5 + 5;
int f[maxn], d[maxn];
struct node {
  int to, val;
};
vector<node> vec[maxn];
int st[maxn];
int cnt; bool flag;

void dfs(int u, int c) {
  if (st[u] != -1) {
    if (c != st[u]) flag = false;
    return;
  }

  st[u] = c; ++cnt;
  for (node it : vec[u]) dfs(it.to, (c+it.val)%2);
}

void solve() {
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n+2; ++i) {
    f[i] = i, d[i] = 0; vec[i].clear(); st[i] = -1;
  }

  while (m -- ) {
    string s; int a; cin >> s >> a;
    if (s == "T") f[a] = n+1, d[a] = 0;
    else if (s == "F") f[a] = n+1, d[a] = 1;
    else if (s == "U") f[a] = n+2, d[a] = 0;
    else {
      int b; cin >> b;
      f[a] = f[b], d[a] = (d[b]+(s=="-")) % 2;
    }
  }

  for (int i = 1; i <= n; ++i) {
    vec[f[i]].push_back({i,d[i]}); vec[i].push_back({f[i],d[i]});
  }

  int res = 0;
  cnt = 0; dfs(n+2, 0); res += cnt-1;

  for (int i = 1; i <= n; ++i) {
    if (st[i] == -1) {
```

```
        cnt = 0, flag = true;
        dfs(i, 0);
        if (!flag) res += cnt;
      }
    }

    cout << res << endl;
}

int main()
{
    int id, T; cin >> id >> T;
    while (T -- ) solve();
    return 0;
}
```

P8865 [NOIP2022] 种花

rht[i][j]: 以 (i,j) 为左点时, 往右有几种情况

down[i][j]: 以 (i,j) 为上点时, 往下有几种情况

s1[i][j]: rht[i][j], rht[i+1][j], ..., rht[n][j] 的和

s2[i][j]: rht[i][j] * down[i][j], rht[i+1][j] * down[i+1][j], ..., rht[n][j] * down[n][j] 的和

预处理完这些信息后, 后续的答案可以 O(1) 求

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

using namespace std;

typedef long long LL;
const int maxn = 1000 + 5;
const int mod = 998244353;
char s[maxn][maxn];
int rht[maxn][maxn], down[maxn][maxn];
int s1[maxn][maxn], s2[maxn][maxn];

void solve() {
    memset(s, 0, sizeof(s));
    memset(rht, 0, sizeof(rht)), memset(down, 0, sizeof(down));
    memset(s1, 0, sizeof(s1)), memset(s2, 0, sizeof(s2));

    int n, m, c, f; cin >> n >> m >> c >> f;
    for (int i = 1; i <= n; ++i) cin >> (s[i]+1);

    for (int i = 1; i <= n; ++i) {
      for (int j = m; j >= 1; --j) {
        if (s[i][j]=='0' && s[i][j+1]=='0') rht[i][j] = rht[i][j+1]+1;
      }
    }
```

```
  for (int j = 1; j <= m; ++j) {
    for (int i = n; i >= 1; --i) {
      if (s[i][j]=='0' && s[i+1][j]=='0') down[i][j] = down[i+1][j]+1;
    }
  }

  for (int i = n; i >= 1; --i) {
    for (int j = 1; j <= m; ++j) {
      if (s[i][j] == '0') {
        s1[i][j] = (s1[i+1][j] + rht[i][j]) % mod;
        s2[i][j] = (s2[i+1][j] + (LL)rht[i][j]*down[i][j]%mod) % mod;
      }
    }
  }

  int resC = 0, resF = 0;
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) {
      if (s[i][j]=='0' && s[i+1][j]=='0') {
        resC = (resC + (LL)rht[i][j]*s1[i+2][j]%mod) % mod;
        resF = (resF + (LL)rht[i][j]*s2[i+2][j]%mod) % mod;
      }
    }
  }

  cout << resC*c << " " << resF*f << endl;
}

int main()
{
  int T, id; cin >> T >> id;
  while (T -- ) solve();
  return 0;
}
```