YNU CPC 2021 VC 024 A 略解

qwop

1.AtCoderBeginnerContest216 C - Many Balls

魔法 B が現在の数を 1 ビット左シフトさせることに対応しています。これにより、Nを二進法で表した時の数列を$0 \to B, 1 \to AB$と置き換えた文字列から最後のBを除いた文字列を出力すればよいです。

表 1 C - Many Balls 入力例

入力例	N	$N_{(2)}$	S
1	5	101	AB B A
2	14	1110	AB AB AB

実装するときは後に使う魔法を先に決定すると楽だと思います。
以下の操作をNが0になるまで行い、出来た文字列を反転させたものを出力すればよいです。

- N偶数の時、Nを2で割り、文字列に B を追加。
- N奇数の時、Nから1を引き、文字列に A を追加。

実装例

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

int main(){
    long long N;
    string S;
    cin >> N;
    while (N > 0){
        if (N % 2 == 0){
            N /= 2;
            S += 'B';
        }
        else{
            N--;
            S += 'A';
        }
    }
    reverse(S.begin(), S.end());
    cout << S << endl;
    return 0;
}
```

2. AtCoderGrandContest 037,A - Dividing a String

文字が 3 文字以上に区切る必要はないです。このことから、Sのi文字目s_iまでにa_i個に区切れるとすると、以下の漸化式が成り立ちます。

$$a_i = \begin{cases} a_{i-1} + 1 \ (s_{i-1} \neq s_i) \\ a_{i-3} + 2 \ (s_{i-1} = s_i) \end{cases} (i \geq 3)$$

ただし、

$$a_0 = 0, a_1 = 1, a_2 = \begin{cases} 1 \ (s_1 \neq s_2) \\ 2 \ (s_1 = s_2) \end{cases}$$

証明は省略します。

実装例

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

int main(){
    string S;
    cin >> S;
    vector<long long> a(S.size() + 2);
    a[0] = 1;
    if (S[0] == S[1]){
        a[1] = 1;
    }
    else{
        a[1] = 2;
    }
    if (S[1] == S[2]){
        a[2] = 2;
    }
    else{
        a[2] = a[1] + 1;
    }
    for (int i = 3; i < S.size(); i++){
        if (S[i] != S[i - 1]){
            a[i] = a[i - 1] + 1;
        }
        else{
            a[i] = a[i - 3] + 2;
        }
    }
    cout << a[S.size() - 1] << endl;
    return 0;
}
```

3. AtCoder Beginner Contest 137,C - Green Bin

ソートした文字列が一致するものがアナグラムになります。ソートした文字列がn種ありi種類目の文字列がa_i個存在するとき、

$$\sum_{i=1}^{n} \binom{a_i}{2}$$

を求めればよいです。

実装例

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

int main(){
    long long N, ans = 0;
    cin >> N;
    vector<string> s(N);
    vector<long long> q(N);
    map<string, long long> a;
    for (int i = 0; i < N; i++){
        cin >> s[i];
        sort(s[i].begin(), s[i].end());
        a[s[i]]++;
        if (a[s[i]] > 1){
            q[i] = 1;
        }
    }
    for (int i = 0; i < N; i++){
        if (q[i] == 0){
            ans += ((a[s[i]] - 1) * a[s[i]] / 2);
        }
    }
    cout << ans << endl;
    return 0;
}
```

4.AtCoder Beginner Contest 209, D – Collision

高橋君と青木君のいる二つの街の間の道の数が偶数本なら街、奇数本なら道で出会います。制約条件から、どの街同士の経路も 1 通りしかありません。これにより、幅優先探索を用いて、街 1 から他の街まで何本の道を経由すればたどり着けるかを求めた後、街 1 から高橋君のいる街、街 1 から青木君のいる街の経由する道路の数の偶奇を調べればよいです。

解答例

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

int main(){
    long long N, Q, x, y;
    cin >> N >> Q;
    vector<vector<long long>> a(N, vector<long long>(0));
    vector<long long> b(N, -1);
    for (int i = 0; i < N - 1; i++){
        cin >> x >> y;
        a[x - 1].push_back(y - 1);
        a[y - 1].push_back(x - 1);
    }
    queue<long long> q;
    q.push(0);
    b[0] = 0;
    while (!q.empty()){
        long long w = q.front();
        q.pop();
        for (int i = 0; i < a[w].size(); i++){
            if (b[a[w][i]] == -1){
                b[a[w][i]] = b[w] + 1;
                q.push(a[w][i]);
            }
        }
    }
    for (int i = 0; i < Q; i++){
        cin >> x >> y;
        if (b[x - 1] % 2 == b[y - 1] % 2){
            cout << "Town" << endl;
        }
        else{
            cout << "Road" << endl;
        }
    }
    return 0;
}
```

5.AtCoderBeginnerContest216,D - Pair of Balls

筒の一番上のボールがすべて異なる場合に操作不能となります。したがって、以下のような操作をすればよいです。

1. 色iが筒の一番上に何個あるかを表す配列c_iを用意する。
2. 色iの下にあるボールの集合d_iを用意する。
3. $c_i = 2$となるiを queue に入れる。
4. queue の先頭の要素wを出し、集合d_wのj番目の要素をd_{wj}としたとき、$c_{d_{wj}}$を 1 大きくする。この時、$c_{d_{wj}} = 2$となるならば、d_{wj}を queue に入れる。
5. queue が空になるまで 4 を繰り返したときに、N回繰り返していた場合、達成可能。そうでない場合、達成不可能。

解答例

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

int main(){
    long long N, M, ans = 0;
    cin >> N >> M;
    vector<long long> k(M);
    vector<long long> c(N);
    vector<vector<long long>> d(N, vector<long long>(0));
    queue<long long> q;
    for (int i = 0; i < M; i++){
        cin >> k[i];
        vector<long long> a(k[i]);
        for (int j = 0; j < k[i]; j++){
            cin >> a[j];
            if (j == 0){
                c[a[j] - 1]++;
                if (c[a[j] - 1] == 2){
                    q.push(a[j] - 1);
                }
            }
            else{
                d[a[j - 1] - 1].push_back(a[j] - 1);
            }
        }
    }
    while (!q.empty()){
        long long w = q.front();
        q.pop();
        ans++;
        for (int i = 0; i < d[w].size(); i++){
            c[d[w][i]]++;
            if (c[d[w][i]] == 2){
                q.push(d[w][i]);
            }
        }
    }
    if (ans < N){
        cout << "No" << endl;
    }
    else{
        cout << "Yes" << endl;
    }
    return 0;
}
```

6. AtCoderGrandContest032, B - Balanced Neighbors

Nが偶数の時、以下のようにノードを要素の和が$N+1$になるような集合に分けます。

$$\{1,N\},\{2,N-1\},\dots,\{i,N+1-i\}$$

この時、同じ集合に属さないノード同士をすべてエッジで結べば、$S=(N+1)((N/2)-1)$となり、条件を満たします。

Nが奇数の時、以下のようにノードを要素の和がNになるような集合に分けます。

$$\{1,N-1\},\{2,N-2\},\dots,\{i,N-i\},\dots,\{N\}$$

この時、同じ集合に属さないノード同士をすべてエッジで結べば、$S=N(N-1)/2$となり、条件を満たします。

解答例

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

int main(){
    long long N;
    cin >> N;
    vector<pair<long long, long long>> a(0);
    if (N % 2 == 1){
        for (int i = 0; i < N - 1; i++){
            for (int j = i + 1; j < N; j++){
                if (i + j + 2 != N){
                    a.push_back(make_pair(i + 1, j + 1));
                }
            }
        }
    }
    else{
        for (int i = 0; i < N - 1; i++){
            for (int j = i + 1; j < N; j++){
                if (i + j + 2 != N + 1){
                    a.push_back(make_pair(i + 1, j + 1));
                }
            }
        }
    }
    cout << a.size() << endl;
    for (int i = 0; i < a.size(); i++){
        cout << a[i].first << ' ' << a[i].second << endl;
    }
    return 0;
}
```

7.AtCoderGrandContest053,B - Taking the middle

青木君のカードの価値の総和を最小にすることを考えます。i回目の操作では、中央値になりうるカードは$N+1-i$枚目から、$N+i$枚目までのものです。
よって、青木君は$1 \leq j \leq N$を満たす、すべての整数jに対して、$N+1-j$枚目から、$N+j$枚目までのカードをj枚以上もつことがわかります。
したがって、以下の操作をN回繰り返せば、青木君のカードの価値の総和の最小値を求めることができます。

- 繰り返しk回目で、$N+1-k$枚目から、$N+k$枚目の内、未選択のカードの中から価値が最小のものを選ぶ。

解答例では優先度付きキューを使って、価値が最小のカードを見つけています。

解答例

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

int main(){
    long long N, ans = 0, sum = 0;
    cin >> N;
    vector<long long> v(2 * N);
    priority_queue<long long, vector<long long>, greater<long long>> c;
    priority_queue<long long, vector<long long>, greater<long long>> d;
    for (int i = 0; i < 2 * N; i++){
        cin >> v[i];
        sum += v[i];
    }
    for (int i = 0; i < N; i++){
        c.push(v[N - 1 - i]);
        d.push(v[N + i]);
        if (c.top() > d.top()){
            ans += d.top();
            d.pop();
        }
        else{
            ans += c.top();
            c.pop();
        }
    }
    ans = sum - ans;
    cout << ans << endl;
    return 0;
}
```

8. AtCoder Regular Contest 107, D - Number of Multisets

i個の数字でjを作ることを考え、その総数を$a_{i,j}$とします。i個の数字でjを作るときに、1を使うか使わないかで場合分けをします。

- つかうとき

1 を一つ使うとした場合、残りの$i-1$個の数字で$j-1$を作ることになります。その総数は$a_{i-1,j-1}$です。

- つかわないとき

$\frac{1}{2},\frac{1}{4},\frac{1}{8}$…の中から$i$個の数字で$j$を作ることになります。ここで、要素をすべて 2 倍にして考えると、$1,\frac{1}{2},\frac{1}{4},$…の中から$i$個の数字で$2j$を作ることになります。その総数は$a_{i,2j}$です。

これらより、漸化式

$$a_{i,j}=a_{i-1,j-1}+a_{i,2j}$$

が成り立ちます。ただし$a_{0,0}=1$、$i<j$の時、$a_{i,j}=0$です。このことから、$a_{N,K}$を求めればよいです。計算量は$1\le i\le N$、$1\le j\le 2N$を満たす範囲の$a_{i,j}$を求めればよいので、

$O(N^2)$です。

解答例

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

int main(){
    long long N, K, X, MOD = 998244353;
    cin >> N >> K;
    vector<vector<long long>> dp(N + 1, vector<long long>(2 * N + 1));
    dp[0][0] = 1;
    for (int i = 1; i < N + 1; i++){
        for (int j = 2 * N; j > 0; j--){
            if (i < j){
                dp[i][j] = 0;
            }
            else{
                dp[i][j] += dp[i - 1][j - 1];
                if (i >= 2 * j){
                    dp[i][j] += dp[i][2 * j];
                }
                dp[i][j] %= MOD;
            }
        }
    }
    cout << dp[N][K] << endl;
    return 0;
}
```