YNU CPC 2021 VC 021

略解

qwop

1 ARC025 A - ゴールドラッシュ

$$\sum_{i}^{7} \max\left(D_i, J_i\right)$$

を計算すればよいです。

解答例

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

int main(){
    int ans = 0;
    vector<int> D(7);
    vector<int> J(7);
    for (int i = 0; i < 7; i++){
        cin >> D[i];
    }
    for (int i = 0; i < 7; i++){
        cin >> J[i];
    }
    for (int i = 0; i < 7; i++){
        ans += max(D[i], J[i]);
    }
    cout << ans << endl;

    return 0;
}
```

2 ABC041 B - 直方体

単純に以下のように実装すると誤答となります。(入力例 4 で確認できます。)

誤答例

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

int main(){
    long long A, B, C, X;
    cin >> A >> B >> C;
    X = (A * B * C) % 1000000007;
    cout << X << endl;
    return 0;
}
```

long long 型では

$$9{,}223{,}372{,}036{,}854{,}775{,}807 \fallingdotseq 9 \times 10^{18}$$

まで扱うことができます。今回は、$X = A \times B \times C \leq 1 \times 10^{27}$より、long long 型で$X$を正しく扱うことができません。ここで、以下の剰余の性質を利用します。

$$(A \times B)\%M = \{(A\%M) \times (B\%M)\}\%M$$

$A \times B \leq 1 \times 10^{18}$ならば long long 型で正しく扱うことができるので、以下のように実装するとよいです。

解答例

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

int main(){
    long long A, B, C, X;
    cin >> A >> B >> C;
    X = ((A * B) % 1000000007 * C) % 1000000007;
    cout << X << endl;
    return 0;
}
```

3 AGC003 A - Wanna go back home

東西の移動について考えてみます。東に進む日数をe、西に進む日数をwとして、

$$ex = wy$$

を満たす、x, y(ただし、$1 \leq x, y$)の組み合わせが存在すれば最終日に家に帰れます。

・$1 \leq e, w$の時、$(x, y) = (w, e)$が条件を満たします。

・$1 \leq e, w = 0$の時、$ex = 0$を満たすxが存在しません。

・$1 \leq w, e = 0$の時、$wy = 0$を満たすyが存在しません。

・$w = 0, e = 0$の時、明らかに等式が成り立ちます。

南北についても同様に考えれば、以下のように実装すればよいです。

分岐	出力
文字列に W が含まれていて E が含まれていない 文字列に E が含まれていて W が含まれていない 文字列に N が含まれていて S が含まれていない 文字列に S が含まれていて N が含まれていない	No
上記以外	Yes

解答例

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

int main(){
    string S;
    cin >> S;
    int n, w, s, e;
    n = count(S.begin(), S.end(), 'N');
    w = count(S.begin(), S.end(), 'W');
    s = count(S.begin(), S.end(), 'S');
    e = count(S.begin(), S.end(), 'E');
    if ((n * s == 0 && n + s > 0) || (w * e == 0 && w + e > 0)){
        cout << "No" << endl;
    }
    else{
        cout << "Yes" << endl;
    }
    return 0;
}
```

4 ARC107 B – Quadruple

条件を満たす(a,b,c,d)を四重ループで全パターン確かめると、計算量$O(N^4)$となるので間に合いません。$(a+b)$と$-(c+d)$に分けて考えます。

ここで、$(a+b)=k\ (2\le k\le 2N)$となる(a,b)の組み合わせの数をa_kとすると、

$$a_k=\min(k-1\ ,2N-k+1)$$

と表すことができます。

$$(a+b)-(c+d)=K$$

となる場合を考えれば、

$$(c+d)=k-K$$

である必要があります。よって、上記の式を満たすのはa_{k-K}通りあるので、

$$\sum a_k a_{k-K}$$

を$2\le k\le 2N$かつ、$2\le k-K\le 2N$の範囲で考えればよいです。

解答例では$k=i+K$として総和を求めています。

解答例、

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

int main(){
    long long N, K, ans = 0;
    cin >> N >> K;
    vector<long long> a(2 * N);
    for (long long i = 2; i <= 2 * N; i++){
        a[i] = min(i - 1, 2 * N - i + 1);
    }
    for (long long i = max(1ll, -K + 1); i <= 2 * N && i + K <= 2 * N; i++){
        ans += (a[i + K] * a[i]);
    }
    cout << ans << endl;

    return 0;
}
```

5 ARC017 C - 無駄なものが嫌いな人

ナップサックの大きさと、品物の大きさが大きくなるので、これらの値を用いた DP は無理です。$N = 32$と品物の数が非常に小さいことを利用します。ここで、全ての品物を入れる入れないの 2 パターンの組み合わせを考えると$N = 32$の時、およそ$2^{32} \fallingdotseq 4 \times 10^9$通りなので、全探索は無理です。そのため、品物を半分ずつ 2 つのグループに分けて、グループごとに品物を入れる入れないの組み合わせを考え、最後に 2 つのグループを組み合わせることを考えます。

これを踏まえ、以下の手順で探索を行います。

1. $\left\lfloor \frac{N+1}{2} \right\rfloor$個の品物の組み合わせを考え、それぞれの組み合わせにおける大きさの総和を格納した配列aを作る。
2. 残りの$\left\lfloor \frac{N}{2} \right\rfloor$個の品物の組み合わせを考え、それぞれの組み合わせにおける大きさの総和を格納した配列bを作る。
3. 配列bをソートする。
4. 配列aの各要素a_iについて、配列bの中から$X - a_i$となる要素の数を、二分探索を利用して数える。

1,では$2^{\left\lfloor \frac{N+1}{2} \right\rfloor}$通り、2,では$2^{\left\lfloor \frac{N}{2} \right\rfloor}$通りの組み合わせがあり$N = 32$の時、65536 通り、であるので、bit 全探索の要領ですべての組み合わせを列挙できます。

3,では、$2^{\left\lfloor \frac{N}{2} \right\rfloor}$要素の配列をソートするので、STL の sort()を使用すれば、計算量$O\left(N2^{\frac{N}{2}}\right)$でソートできます。

4 では、$2^{\left\lfloor \frac{N+1}{2} \right\rfloor}$回、$2^{\left\lfloor \frac{N}{2} \right\rfloor}$要素の配列で二分探索を行うので計算量$O\left(N2^{\frac{N}{2}}\right)$で探索できます。実装時には upper_bound, lower_bound を使用すると便利です。

解答例

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

int main(){
    long long N, X, ans = 0, q, r, l;
    cin >> N >> X;
    vector<long long> w(N);
    vector<long long> a(0);
    vector<long long> b(0);
    for (long long i = 0; i < N; i++){
        cin >> w[i];
    }
    for (long long i = 0; i < (1 << ((N + 1) / 2)); i++){
        q = 0;
        for (long long j = 0; j < (N + 1) / 2; j++){
            if ((i & (1 << j)) != 0){
                q += w[j];
            }
        }
        a.push_back(q);
    }
    for (long long i = 0; i < (1 << (N / 2)); i++){
        q = 0;
        for (long long j = 0; j < N / 2; j++){
            if ((i & (1 << j)) != 0){
                q += w[j + (N + 1) / 2];
            }
        }
        b.push_back(q);
    }
    sort(b.begin(), b.end());
    for (long long i = 0; i < a.size(); i++){
        r = upper_bound(b.begin(), b.end(), (X - a[i])) - b.begin();
        l = lower_bound(b.begin(), b.end(), (X - a[i])) - b.begin();
        ans += r - l;
    }
    cout << ans << endl;
    return 0;
}
```

6 ABC184 F - Programming Contest

ほぼ前の問題と同じなので解説は省略します。

解答例

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

int main(){
    long long N, T, ans = 0, q, r;
    cin >> N >> T;
    vector<long long> A(N);
    vector<long long> a(0);
    vector<long long> b(0);
    for (long long i = 0; i < N; i++){
        cin >> A[i];
    }
    for (long long i = 0; i < (1 << ((N + 1) / 2)); i++){
        q = 0;
        for (long long j = 0; j < (N + 1) / 2; j++){
            if ((i & (1 << j)) != 0){
                q += A[j];
            }
        }
        a.push_back(q);
    }
    for (long long i = 0; i < (1 << (N / 2)); i++){
        q = 0;
        for (long long j = 0; j < N / 2; j++){
            if ((i & (1 << j)) != 0){
                q += A[j + (N + 1) / 2];
            }
        }
        b.push_back(q);
    }
    sort(a.begin(), a.end());
    sort(b.begin(), b.end());
    for (long long i = 0; i < a.size() && a[i] <= T; i++){
        r = upper_bound(b.begin(), b.end(), (T - a[i])) - b.begin();
        if (a[i] + b[r - 1] > ans){
            ans = a[i] + b[r - 1];
        }
    }
    cout << ans << endl;
    return 0;
}
```

7 AGC026 C - String Coloring

色の塗分け方法が2^{2N}通りあるので、文字列を前半、後半のN文字ずつに分けて考えます。

色の塗分け方法は2^N通りずつありますが、

1. 文字列の前半で赤く塗られた文字を左から右へ読んだものと，文字列の後半で青く塗られた文字を右から左へ読んだものが同じ
2. 文字列の前半で青く塗られた文字を右から左へ読んだものと，文字列の後半で赤く塗られた文字を左から右へ読んだものが同じ

という 2 つの条件を満たせば問題の条件を満たします。また、後半の文字列を反転させ、塗る色を逆にして考えると

1. 文字列の前半で赤く塗られた文字を左から右へ読んだものと，文字列の後半で赤く塗られた文字を左から右へ読んだものが同じ
2. 文字列の前半で青く塗られた文字を右から左へ読んだものと，文字列の後半で青く塗られた文字を右から左へ読んだものが同じ

という条件になり、赤と青で塗られた文字が同じ文字列になるかどうかを考えればよくなります。

これらを踏まえると、塗り分ける文字列のペアを key、そのペアが何通りあるかを value とした map を作成することで問題を解くことができます。

解答例

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

int main(){
    long long N, ans = 0;
    string S, r, b;
    cin >> N >> S;
    map<pair<string, string>, long long> l;
    reverse(S.begin() + N, S.end());
    for (long long i = 0; i < (1 << N); i++){
        r = "";
        b = "";
        for (long long j = 0; j < N; j++){
            if ((i & (1 << j)) != 0){
                r += S[j];
            }
            else{
                b += S[j];
            }
        }
        l[make_pair(r, b)]++;
    }
    for (long long i = 0; i < (1 << N); i++){
        r = "";
        b = "";
        for (long long j = 0; j < N; j++){
            if ((i & (1 << j)) != 0){
                r += S[j + N];
            }
            else{
                b += S[j + N];
            }
        }
        ans += l[make_pair(r, b)];
    }
    cout << ans << endl;
    return 0;
}
```