

ABSを解く！

triC 2024/01/06

0

テーマの色と内容



黒

一般的なプログラミング
の内容



赤

競プロ特化の内容

1. PracticeA – Welcome to AtCoder

1 PracticeA – Welcome to AtCoder

この問題のポイント

- ・ 入出力ができるようになる(標準入出力)

1.1

入出力ができるようになる(C++)

入力: 変数を定義して入力の数だけcinで受け取る

出力:cout、空白区切りは空白を出力する

解答例

```
#include <iostream>
using namespace std;
int main(){
    int a,b,c;
    cin >> a >> b >> c;
    string s;
    cin >> s;
    cout << a + b + c << " " << s << endl;
}
```

1.2

入出力ができるようになる(Python)

入力: `input()`で一行ずつ受け取る、入力は文字列で受け取られる

出力:基本は`print(a)`とかでいい。複数を出力する際は文字列にする

参考

<https://qiita.com/scythercas/items/5e08dffffb49468dd1176>

<https://qiita.com/Koichiro-Kanaya/items/4f46fe2c98a415681210#Python%E3%81%A7%E3%81%AE%E5%85%A5%E5%87%BA%E5%8A%9B>

解答例1

```
a = int(input())
b,c = map(int,input().split())
s = input()
ans = str(a + b + c) + ' ' + s
print(ans)
```

解答例2

```
a = int(input())
b, c = map(int, input().split())
s = input()
print("{} {}".format(a+b+c, s))
```

1.3

標準出力を使う

標準出力とは…画面に出力すること(シェルに出力すること)

デバッグの際に便利

2. ABC086A-Product

2 ABC086A-Product

解き方

- ・ 偶数か奇数か 2 で割った余りで判定する

C++

```
using namespace std;
int main(){
    int a,b;cin >> a >> b;
    if((a * b) % 2 == 1){
        cout << "Odd" << endl;
    }else{
        cout << "Even" << endl;
    }
}
```

Python

```
a,b = map(int,input().split())
if (a * b) % 2 == 1:
    print('Odd')
else:
    print('Even')
```

2.1 ABC086A-Product

この問題のポイント

なし

3. ABC081A - Placing Marbles

3 ABC081A - Placing Marbles

解き方

1の箇所を全探索する

```
s = input()
cnt = 0
if s[0] == '1':
    cnt+=1
if s[1] == '1':
    cnt+=1
if s[2] == '1':
    cnt+=1
print(cnt)
```

Python

```
#include <iostream>
using namespace std;
int main(){
    string s;cin >> s;
    int cnt = 0;
    if(s[0] == '1'){
        cnt++;
    }
    if(s[1] == '1'){
        cnt++;
    }
    if(s[2] == '1'){
        cnt++;
    }
    cout << cnt << endl;
}
```

C++

3.1 ABC081A - Placing Marbles

この問題のポイント

- ・ 全探索
- ・ 文字列の扱い方

3.2

全探索

全探索とは…

あり得る全てのパターンをしらみつぶしに調べるアルゴリズム

例) 4桁の暗証番号を0000から9999までひとつずつ調べる

0 0 0 0

~

9 9 9 9

3.3

文字列(string)の扱い方

文字列の変数s,tについて

- 文字列のi文字目 : `s[i]`

- 文字列の連結: `s + t`

C++では $s = s + t : O(|s| + |t|)$
 $s += t : O(|t|)$

文字列の長さの取得:

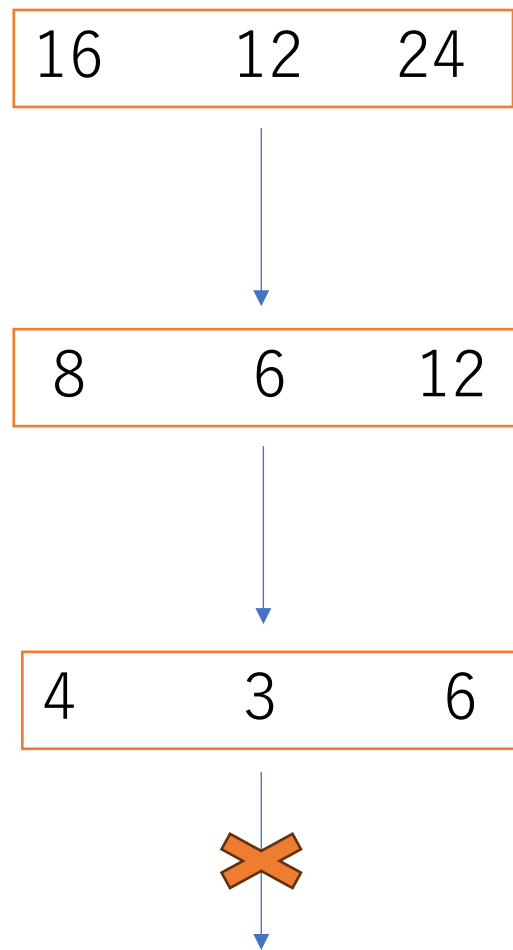
C++ : `s.size()` または `s.length()`
Python : `len(s)`

4. ABC081B - Shift only

4 ABC081B - Shift only

解き方

- ・ 操作を行えなくなるまで実際に行う
↳ while文をうまく使う



4.1 ABC081B - Shift only

python

```
n = int(input())
a = list(map(int, input().split()))

cnt = 0

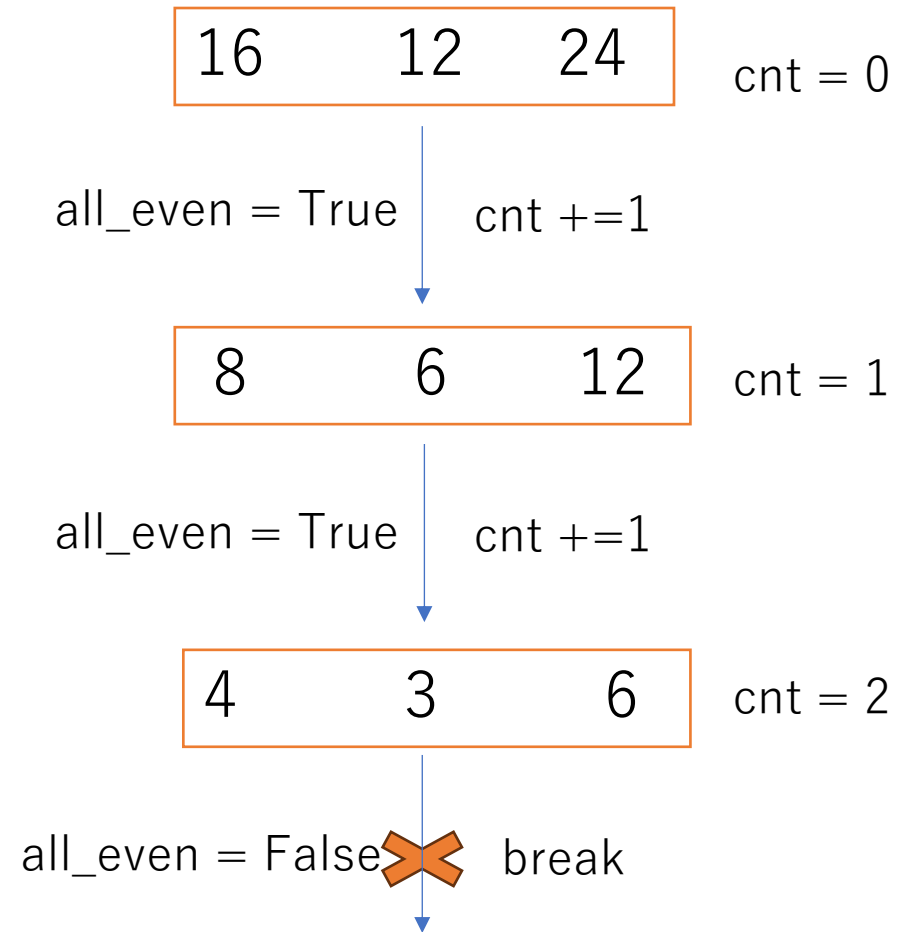
while True:
    all_even = True
    for i in range(len(a)):
        if a[i] % 2 != 0:
            all_even = False

    if all_even == False:
        break

    for i in range(len(a)):
        a[i] /= 2

    cnt += 1

print(cnt)
```



4.2 ABC081B - Shift only

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int n;cin >> n;
    vector<int> a(n);
    for(int i = 0;i < n;i++){
        cin >> a[i];
    }
    int cnt = 0;
```

C++

```
while(true){
    bool all_even = true;
    for(int i = 0;i < n;i++){
        if(a[i] % 2 != 0){
            all_even = false;
        }
    }
    if(all_even == false){
        break;
    }
    for(int i = 0;i < n;i++){
        a[i]/= 2;
    }
    cnt++;
}
cout <<cnt << endl;
}
```

4.3 ABC081B - Shift only

この問題のポイント

- while文とfor文の使い分け
- `vector(C++)`

4.4

while文とfor文の使い分け

ループの回数が決まっている

yes

for文

no

while文

4.5

C++の配列はvectorを使う

C++の生配列はちょっと扱いづらいのでvectorを使うのがおすすめ

(少なくとも競プロでは)

良い資料

APG4bで勉強がおすすめ

https://atcoder.jp/contests/apg4b/tasks/APG4b_n

リファレンス

<https://cpprefjp.github.io/reference/vector/vector.html>

5. ABC087B - Coins

5

ABC087B - Coins

解き方

・ 全探索

500 円玉が 0 枚 ～ A 枚の場合 ($A + 1$ 通り)

100 円玉が 0 枚 ～ B 枚の場合 ($B + 1$ 通り)

50 円玉 が 0 枚 ～ C 枚の場合 ($C + 1$ 通り) をすべて調べる

※0枚の場合もあるので注意

5.1 ABC087B - Coins

C++

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int A,B,C,X;
    cin >> A >> B >> C >> X;
    int ans = 0;
    for (int a = 0; a <= A; a++) {
        for (int b = 0; b <= B; b++) {
            for (int c = 0; c <= C; c++) {
                int sum = 500*a + 100*b + 50*c;
                if (sum == X) {
                    ans++;
                }
            }
        }
    }
    cout << ans << endl;
}
```

Python

```
A = int(input())
B = int(input())
C = int(input())
X = int(input())

ans = 0

for a in range(A+1):
    for b in range(B+1):
        for c in range(C+1):
            sum = 500*a + 100*b + 50*c
            if sum == X:
                ans += 1

print(ans)
```

5.2 ABC087B - Coins

この問題のポイント

- ・ 多重ループ

5.3

多重ループに慣れる

多重ループには慣れも必要

添字を i, j, k, \dots \Rightarrow 分かりやすい添字

にする工夫もある

例)

```
for (int a = 0; a <= A; ++a) {  
    for (int b = 0; b <= B; ++b) {  
        for (int c = 0; c <= C; ++c) {  
            //略  
        }  
    }  
}
```

6. ABC083B - Some Sums

解き方

1~Nまでの数それぞれについて各桁の和を調べる

①余りを使って求めるやり方

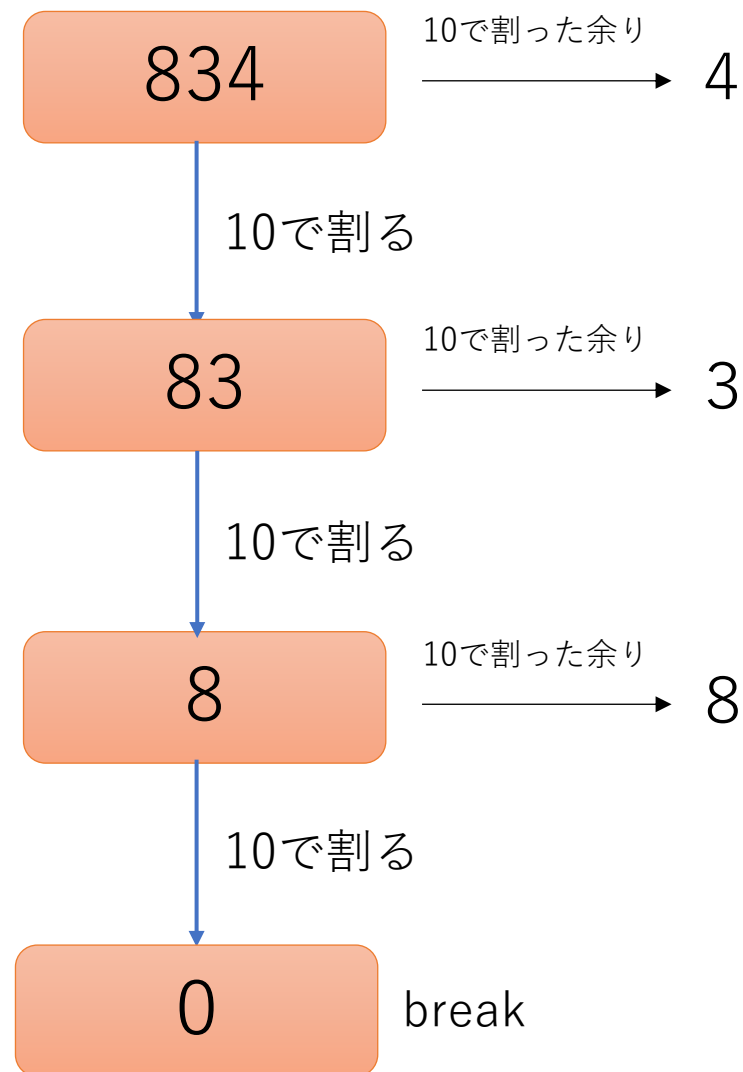
②文字列にして求めるやり方

6.1 ABC083B - Some Sums

①余りを使って求めるやり方

10で割った余りを足す、元の数に10で割るを繰り返す

例) 834の各桁の和を求めるとき



6.2 ABC083B - Some Sums

①余りを使って求めるやり方

```
#include <iostream>
using namespace std;
int main(){
    int N,A,B;
    cin >> N >> A >> B;
    int ans = 0;
    for(int i = 0;i <= N;i++){
        int digit_sum = 0;
        int val = i;
        while(val > 0){
            digit_sum += val % 10;
            val /= 10;
        }
        if(A <= digit_sum && digit_sum <= B){
            ans += i;
        }
    }
    cout << ans << endl;
}
```

C++

Python

```
N, A, B = map(int, input().split())

ans = 0

for i in range(1,N+1):
    val = i
    digit_sum = 0
    while val > 0:
        digit_sum += val % 10
        val //= 10
    if A <= digit_sum and digit_sum <= B:
        ans+= i
print(ans)
```

6.3 ABC083B - Some Sums

②文字列にして求めるやり方

C++

```
#include <iostream>
using namespace std;
int main(){
    int N,A,B;
    cin >> N >> A >> B;
    int ans = 0;
    for(int i = 1;i <= N;i++){
        string s = to_string(i); //iを文字列にする
        int digit_sum = 0;
        for(int j = 0;j < s.size();j++){
            digit_sum += s[j] - '0';
        }
        if(A <= digit_sum && digit_sum <= B){
            ans += i;
        }
    }
    cout << ans << endl;
}
```

Python

```
N, A, B = map(int, input().split())

ans = 0

for i in range(1,N+1):
    s = str(i) #iを文字列にする
    digit_sum = 0
    for c in s:
        digit_sum += int(c) # 数字にする
    if A <= digit_sum and digit_sum <= B:
        ans += i
print(ans)
```


6.4 ABC083B - Some Sums

この問題のポイント

- ・ 整数の剰余

6.5

整数の剰余で気をつける点

Pythonの注意点

`/` は割り算(小数部分も計算する)

`//` は整数除算(割り算の商が帰ってくる)

```
print(10 / 3)  
# 3.3333333333333335
```

```
print(0.1 / 0.03)  
# 3.3333333333333335
```

```
print(10 // 3)  
# 3
```

```
print(0.1 // 0.03)  
# 3.0
```

6.6

整数の剰余で気をつける点

C++の注意点

整数除算は0に近い方に丸め込まれる！

```
cout << 10/3 << endl;  
// 3
```

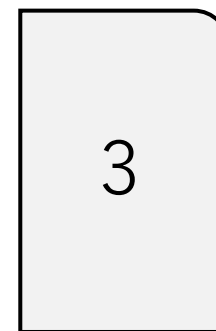
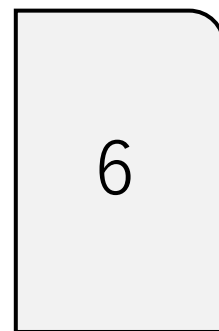
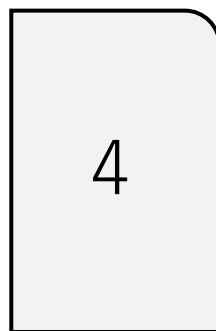
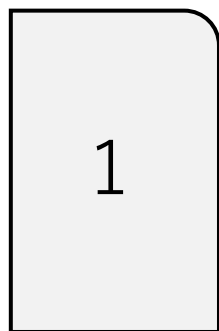
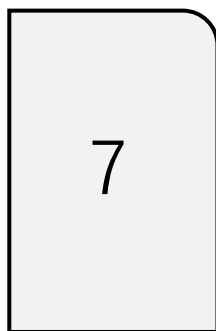
```
cout << -10/3 << endl;  
//-3
```

7. ABC088B - Card Game for Two

7 ABC088B - Card Game for Two

解き方

最適な戦略とは...?



7.1 ABC088B - Card Game for Two

解き方

7	1	4	6	3
---	---	---	---	---

Alice

7	+	4	+	1	=	12
---	---	---	---	---	---	----

Bob

6	+	3	=	9
---	---	---	---	---

最適な戦略 =
取れる中で一番大きいものを取る!

7.2 ABC088B - Card Game for Two

解き方

最適な戦略 = 取れる中で一番大きいものを取る

-> 大きいものから並べて交互に取れば良い

(降順ソートして前から見る)

Python

```
N = int(input())
a = list(map(int, input().split()))

a.sort(reverse=True) #降順ソート

Alice = 0
Bob = 0

for i in range(N):
    if i % 2 == 0: #Aliceのターン
        Alice += a[i]
    else:          #Bobのターン
        Bob += a[i]

print(Alice - Bob)
```

7.3 ABC088B - Card Game for Two

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main(){
    int N;
    cin >> N;
    vector<int> a(N);
    for(int i = 0; i < N ; i++){
        cin >> a[i];
    }
    sort(a.begin(), a.end()); //昇順にソート
    reverse(a.begin(), a.end()); //降順にする
```

C++

```
int Alice = 0;
int Bob = 0;
for (int i = 0; i < N; i++) {
    if (i % 2 == 0) { // Alice のターン
        Alice += a[i];
    }
    else { // Bob のターン
        Bob += a[i];
    }
}
cout << Alice - Bob << endl;
}
```

※ sortは algorithmに入っている

7.4 ABC088B - Card Game for Two

この問題のポイント

- ・ 貪欲法
- ・ 入出力例をみて考える

7.5

貪欲法

貪欲法とは

後のことは考えずその場面での最善を選んでいく手法

例)

1円玉,5円玉,10円玉がいっぱいあるときに、X円ちょうどをできるだけ少ない枚数で払う方法

貪欲法だとできない例)

1円玉,7円玉,10円玉がいっぱいあるときに、X円ちょうどをできるだけ少ない枚数で払う方法

7.6

入出力例を見て考える

問題文を見ても一見分からない問題でも

入出力例を使って実際に試したりすると分かることはよくある

コーナーケースに早く気づくためにも問題を読むときに入出力例も見ておくの良い

8. ABC085B - Kagami Mochi

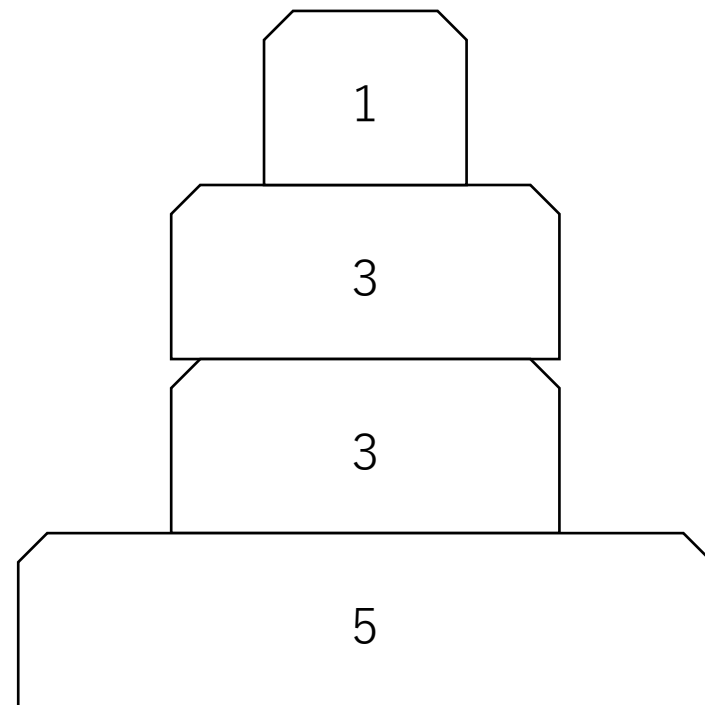
8 ABC085B - Kagami Mochi

解き方

値の種類の数を数える！

①バケット法を使う

②集合を管理できるデータ構造を使う(set)



8.1 ABC085B - Kagami Mochi

①バケット法

バケット法とは

取りうる値に対応するデータの置き場（バケツ）を用意しておき、
整列したい値を順に対応するバケツに入れていく方式

配列numを用意して、

$\text{num}[i] := \text{値 } i \text{ が何個あるか}$ を記録すればよい

8.2 ABC085B - Kagami Mochi

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int N;
    cin >> N;
    vector<int> d(N);
    for(int i = 0;i < N;i++){
        cin >> d[i];
    }
    vector<int> num(101);
    for(int i = 0;i < N;i++){
        num[d[i]]++;
    }
    int ans = 0;
    for(int i = 1;i <= 100;i++){
        if(num[i] >= 1){//少なくとも1個 i があれば
            ans++;
        }
    }
    cout << ans << endl;
}
```

C++

Python

```
N = int(input())
d = [int(input()) for _ in range(N)]

num = [0]*101
for v in d:
    num[v] += 1

ans = 0
for i in range(1,101):
    if num[i] >= 1: #少なくとも1個 i があれば
        ans += 1
print(ans)
```

8.3 ABC085B - Kagami Mochi

②データ構造(set)を用いる

set : 重複の無いデータのまとまりを扱うためのデータ型

配列の中に出現する要素数とか集合内の要素の最大値、最小値の取得などに使われる

setに関して

C++ https://atcoder.jp/contests/APG4b/tasks/APG4b_aa
 <https://cpprefjp.github.io/reference/set/set.html>

Python <https://note.nkmk.me/python-set/>

8.4 ABC085B - Kagami Mochi

C++

```
#include <iostream>
#include <vector>
#include <set>
using namespace std;
int main(){
    int N;
    cin >> N;
    vector<int> d(N);
    for(int i = 0; i < N; i++){
        cin >> d[i];
    }
    set<int> values;
    for(int i = 0; i < N; i++){
        values.insert(d[i]);
    }
    cout << values.size() << endl;
}
```

Python

```
N = int(input())

d = [int(input()) for _ in range(N)]

print(len(set(d)))
```

8.5 ABC085B - Kagami Mochi

この問題のポイント

- ・ 様々なデータ構造を使いこなせるようになる

8.6

様々なデータ構造を使いこなす

競プロでは使えると便利な(使えないと解けない)データ構造がいくつもある

例) set map queue stack priority_queue ...

内部実装について詳しく知っている必要はほぼないので、たくさん使って覚える

C++使う人は一通り目を通くのを推奨します

https://atcoder.jp/contests/apg4b/tasks/APG4b_aa

Pythonの人は?

9. ABC085C - Otoshidama

解く前に...

1秒間で処理できるfor文ループの回数は 10^8 回程度

9.1 ABC085C - Otoshidama

解き方

$$(1\text{万円札の枚数}) + (5\text{千円札の枚数}) + (\text{千円札の枚数}) = N$$

$$\Leftrightarrow (\text{千円札の枚数}) = N - (1\text{万円札の枚数}) - (5\text{千円札の枚数})$$

を利用してループの数を減らす

9.2 ABC085C - Otoshidama

解き方

1万円札の枚数と5千円札
の枚数を全探索する

Python

千円札の枚数は一意に決まる

```
N, Y = map(int, input().split())

ans10000 = -1
ans5000 = -1
ans1000 = -1

for i in range(N+1): #10000円の枚数を 0 ~ N で調べる
    for j in range(N+1 - i): #5000円の枚数を 0 ~ N-i で調べる
        k = N - i - j
        if 10000 * i + 5000 * j + 1000 * k == Y:
            ans10000 = i
            ans5000 = j
            ans1000 = k

print(ans10000, ans5000, ans1000)
```

9.3 ABC085C - Otoshidama

C++

```
#include<iostream>
using namespace std;
int main(){
    int N,Y;cin >> N >> Y;
    int ans100000 = -1,ans5000 = -1,ans1000 = -1;
    for(int i = 0; i <= N;i++){
        for(int j = 0;i + j <= N;j++){
            int k = N - i - j;
            int sum = 10000 * i + 5000 * j + 1000 * k;
            if(sum == Y){
                ans100000 = i;
                ans5000 = j;
                ans1000 = k;
            }
        }
    }
    cout << ans100000 << " " << ans5000 << " " << ans1000 << endl;
}
```


9.4 ABC085C - Otoshidama

この問題のポイント

- ・ 計算量のオーダーを意識する

計算量のオーダー

- ・ どのくらいの計算時間がかかるのかを判断するために使う
- ・ プログラムへの入力サイズを n として、プログラムの実行時間が n に応じてどう変化するか考える
- ・ $O(n)$ とか $O(n^2)$ といったように書く

すごく分かりやすい => <https://qiita.com/drken/items/872ebc3a2b5caaa4a0d0>

- ・ ループの回数だと考えると分かりやすい


10. ABC049C - 白昼夢

解き方


- 4つの文字列が取り除けるかを後ろから貪欲に試す

前からだと

dream dreamer



erase eraser



完全に含まれてしまう文字列がある

後ろからだと

maerd

remaerd

esare

resare

完全に含まれてしまう文字列がない！

10.1 ABC049C - 白昼夢

完全に含まれる文字列があるとどんな問題があるか

dream

dreamer

erase

eraser

例えば

dreamer.... という文字列が与えられたとき、

dream er.... と dream で区切るべきか

dreamer と dreamer で一括りとするべきかが分からない

10.2 ABC049C - 白昼夢

後ろから見ると

maerd remaerd esare resare

完全に他の文字列に含まれる文字列がないから簡単に区切りを入れられる!

maerdesare => maerd esare

※入力の文字列も反転させる必要がある

10.3 ABC049C - 白昼夢

```
S = input()
#後ろから見るためにすべての文字列を反転する
S = S[::-1]
words = ["dream", "dreamer", "erase", "eraser"]
words = [word[::-1] for word in words]
able = True
i = 0
while i < len(S):
    dividable = False # 4 個の文字列どれかで divide できるか
    for word in words:
        if S[i:i + len(word)] == word:#divide できたら i を進める
            dividable = True
            i += len(word)
            break
    if dividable == False:#divide できなかった
        able = False
        break

if able:
    print("YES")
else :
    print("NO")
```

Python

10.4 ABC049C - 白昼夢

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
    string S;
    cin >> S;
    // 後ろから見るためにすべての文字列を反転する
    reverse(S.begin(),S.end());
    vector<string> words = {"dream","dreamer","erase","eraser"};
    for(int i = 0;i < 4;i++){
        reverse(words[i].begin(),words[i].end());
    }
```

C++

```
bool able = true;
for(int i = 0;i < S.size();){
    bool dividable = false;// 4 個の文字列どれかで divide できるか
    for(int j = 0;j < 4;j++){
        string word = words[j];
        if(S.substr(i,word.size()) == word){// divide できたら i を進める
            dividable = true;
            i += word.size();
            break;
        }
    }
    if(!dividable){// divide できなかった
        able = false;
        break;
    }
}
if(able){
    cout << "YES" << endl;
} else{
    cout << "NO" << endl;
}
}
```


10.4 ABC049C - 白昼夢

この問題のポイント

- ・ 後ろから考える

10.5

後ろから考える

今回のように後ろやゴールから考えると簡単になる問題はよくある

問題例

A41 - Tile Coloring https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_ao

B41 - Reverse of Euclid https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_dn

11. ABC086C - Traveling

11 ABC086C - Traveling

解き方

- ・ 時間内に次の目的地にたどり着けるか & ちょうど次の目的地にいられるかを判定

11.1 ABC086C - Traveling

時間内に次の目的地にたどり着けるか

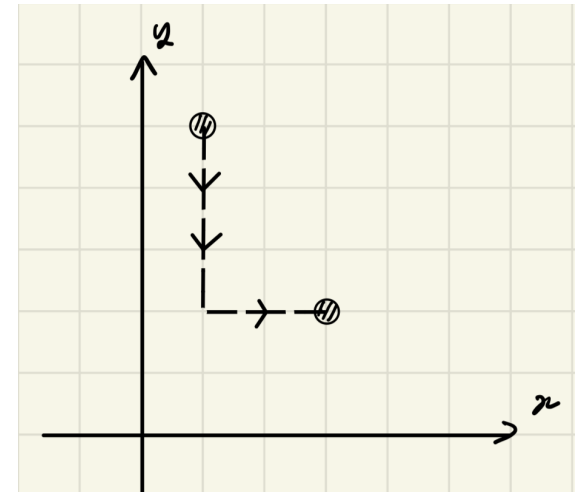
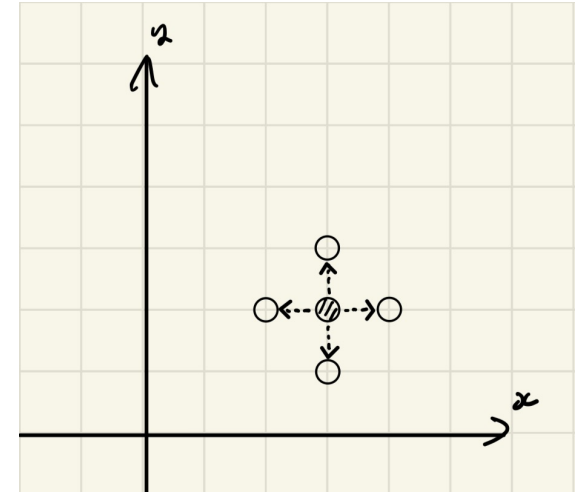
- 1秒でx軸方向に 1 またはy軸方向に 1 移動できる
- T 秒で最大で距離 T だけ移動できる

$$dt = t_{i+1} - t_i$$

$$\text{dist} = \text{abs}(x_{i+1} - x_i) + \text{abs}(y_{i+1} - y_i)$$

とすると

$\text{dist} \leq dt$ なら目的地にたどり着ける



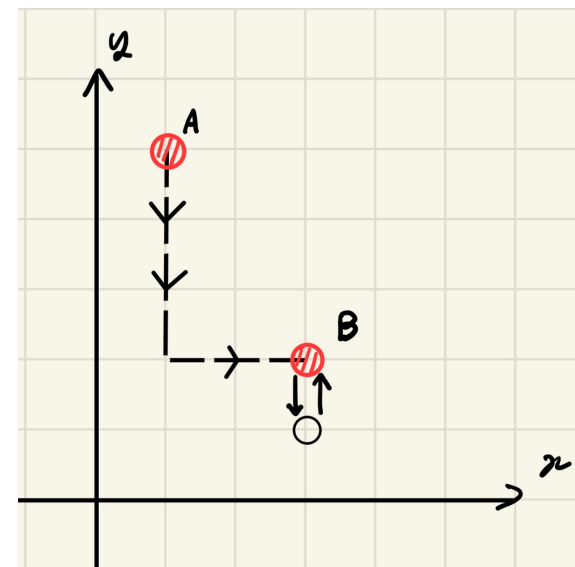
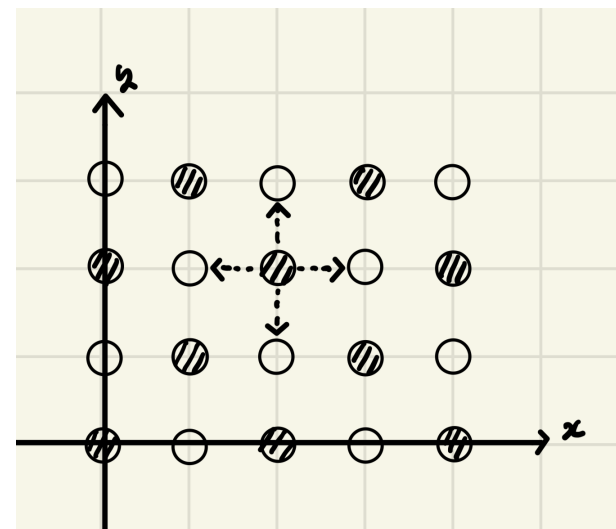
11.2 ABC086C - Traveling

ちょうど次の目的地にいられるか

- ・ 毎回の操作で $x_i + y_i$ の偶奇が反転する

つまり、時間ぴったりで次の目的地にいるためには

$\text{dist} \leq \text{dt}$ かつ dist と dt の偶奇の一致 が必要十分条件



11.3 ABC086C - Traveling

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int N;
    cin >> N;
    vector<int> t(N+1),x(N+1),y(N+1);
    for(int i = 0;i < N;i++){
        cin >> t[i+1] >> x[i+1] >> y[i+1];
    }
}
```

C++

```
bool able = true;
for(int i = 0;i < N;i++){
    int dt = t[i+1] - t[i];
    int dist = abs(x[i+1] - x[i]) + abs(y[i+1] - y[i]);
    if(dt < dist){
        able = false;
    }
    if(dist % 2 != dt % 2){
        able = false;
    }
}

if(able){
    cout << "Yes" << endl;
}else{
    cout << "No" << endl;
}
}
```

11.4 ABC086C - Traveling

```
N = int(input())
prev_t, prev_x, prev_y = 0, 0, 0

able = True

for _ in range(N): # 入力を受け取りながら処理をする
    t, x, y = map(int, input().split())
    dist = abs(x - prev_x) + abs(y - prev_y)
    dt = t - prev_t

    if dist > dt :
        able = False
    if dist % 2 != dt % 2:
        able = False
    prev_t, prev_x, prev_y = t, x, y

if able == True:
    print("Yes")
else:
    print("No")
```

Python

11.5 ABC086C - Traveling

この問題のポイント

- ・ 偶奇性に注目

11.6

偶奇性に注目する

偶奇性(パリティ)に着目する問題も多い

AGC010A – Addition https://atcoder.jp/contests/agc010/tasks/agc010_a

AGC020A – Move and Win https://atcoder.jp/contests/agc020/tasks/agc020_a

終わり

参考

<https://qiita.com/drken/items/fd4e5e3630d0f5859067#5-%E9%81%8E%E5%8E%BB%E5%95%8F%E7%B2%BE%E9%81%B8-10-%E5%95%8F>

<https://qiita.com/drken/items/872ebc3a2b5caaa4a0d0>

<https://tysonblog-whitelabel.com/atcoder-beginners-selection>

https://note.com/keisuke_funabiki/n/nec9df628f77c