[Problem C]　ボロック予想

動的計画法(Dynamic Programming, DP)により解ける。
x を表現するのに必要な正四面体数の個数の最小値を n_tetra[x]とする。
正四面体数 1, 4, 10, 20, 35, … , x 以下の最大の正四面体数　に対して、n_tetra[x]は、{n_tetra[x − 1], n_tetra[x − 4], n_tetra[x − 10], n_tetra[x − 20], n_tetra[x − 35], …, n_tetra[x − x 以下の最大の正四面体数]}の最小値+1 で求めることができる。ここで、n_tetra[0] = 0 である。したがって、x = 1 から順に x を一つずつ増やしながら n_tetra[x]を求めていくことができる。

問題として与えられる x の最大値は 1,000,000 であり、上記のアルゴリズムでは、一度 n_tetra[x]を計算すると、1 ≦ y ＜ x に対して n_tetra[y]も求まっている。したがって最初に準備として百万未満の全ての整数 x に対して n_tetra[x]を求めておけば、与えられた数値に対する答えは n_tetra[与えられた数値]を見るだけで得られる。

x を表現するのに必要な奇数の正四面体数の個数の最小値も同様の考え方で求められる。すなわち、求める値を n_odd_t[x]とすると、奇数の正四面体数 1, 35, …, x 以下の奇数の正四面体数に対して、n_odd_t[x] は{n_odd_t[x – 1], n_odd_t[x – 35], …, n_odd_t[x – x 以下の奇数の正四面体数]}の最小値＋１で求めることができる。

整数 a が奇数かどうかの判定は、a % 2 あるいは a & 1 が 1 となるかどうかで判定できる。

プログラム例

```
// ACM-ICPC 2010 Japan Online Contest Problem C
// http://icpc2010.honiden.nii.ac.jp/domestic-contest/problems#section_C
//      ファイル名: c1.c
//      コンパイル方法: cc pc1.c
//      実行方法: ./a.out < C0 > C0.result など
//      チェック方法: diff C0.ans C0.result など
//
// アルゴリズム概要
// x を表すために必要な正四面体数の個数の最小値を n_tetra[x]とすると
// 正四面体数 1, 4, 10, ... に対して、
// n_tetra[x] = min(n_tetra[x-1], n_tetra[x-4], n_tetra[x-10], ...) + 1
// となる。即ち n_tetra[x] の値は y<x に対しする n_tetra[y] を用いて計算できる。
// x は 1000000 未満であるので、x=1 から 999999 まで順に n_tetra[x]を求める。
// 一方、x を表すために必要な奇数の正四面体数の個数の最小値を n_odd_t[x]とすると
// 奇数の正四面体数 1, 35, ... に対して
// n_odd_t[x] = min(n_odd_t[x-1], n_odd_t[x-35], ...)+1
// となるので、n_tetra[x] の場合と同様に
// x=1 から 999999 まで順に n_odd_t[x]を求めることができる
// このような解法は動的計画法(dynamic programming)と呼ばれる

#include <stdio.h>
```

```
#define N 1000000          // 与えられるデータは 1000000 より小さい

int n_tetra[N];            // 正四面体数の最小個数を入れる配列
int n_odd_t[N];            // 奇数の正四面体数の最小個数を入れる配列

// 1 <= x < N に対して n_tetra[x] と n_odd_t[x] を全て求める
void prepare()
{
  int x,j,tetra;
  n_tetra[0] = n_odd_t[0] = 0;

  for(x=1; x<N; x++){
    n_tetra[x] = n_odd_t[x] = x;      // 正四面体数 1 を x 個用いれば x を表現できる
    for(j=1; ; j++){                  // 各正四面体数に対して小さい方から順にチェック
      tetra = j*(j+1)*(j+2)/6;        // tetra は j 番目の正四面体数
      if(tetra > x) break;            // x より大きくなれば、
                                      // これ以降の正四面体数は用いられない
      // この正四面体数を用いることにより、必要な正四面体数の個数が少なくなるなら、
      // その個数を採用
      if(n_tetra[x] > n_tetra[x - tetra] + 1)
        n_tetra[x] = n_tetra[x - tetra] + 1;
      if(tetra & 1){                          // 正四面体数が奇数の場合も同様にチェック
        if(n_odd_t[x] >   n_odd_t[x-tetra]+1)
          n_odd_t[x] = n_odd_t[x-tetra]+1;
      }
    }
  }
}

int main()
{
  int x;
  // まず最初に 1 <= x < N に対して n_tetra[x] と n_odd_t[x] を全て求めておく
  prepare();

  while(1){
    scanf("%d", &x);          // x を入力
    if(x==0) break;           // x == 0 なら終了
    printf("%d %d¥n", n_tetra[x], n_odd_t[x]);          // n_tetra[x]と n_odd_t[x]を出力
  }
}
```