Multi-University #5 C "History Repeat Itself" Solution

write by Gestalti Lur
2012-09-16
題目鏈接：http://acm.hdu.edu.cn/showproblem.php?pid=4342

題目大意

輸出第 N 个非完全平方數 M 以及⌊sqrt(i)⌋1<=i<=M 的和。

算法分析

考慮如果已知 M 的話那麼前面的完全平方數爲⌊sqrt(M)⌋個[1]，那麼顯然有 M-⌊sqrt(M)⌋=N。所以可以在[N,2^31]這個區間二分 M。二分實現的時候注意精度問題。

后面一問可以首先根據前面的完全平方數規律找出計算的方法：可以觀察出⌊sqrt(i)⌋爲 1 的有 3 項，爲 2 的有 5 項，爲 3 的有 7 項……推知設 a(i)爲⌊sqrt(i)⌋的項數，即有 a(i)=i*2+1。那麼⌊sqrt(i)⌋1<=i<=M 顯然是要計算 1..⌊sqrt(M)⌋的 a(i)之和加上 M-⌊sqrt(M)⌋^2 多出來的部分。這個多出來的部分是不完整的 a(⌊sqrt(M)⌋+1),所以和爲((⌊sqrt(M)⌋ + 1)*2+1)*(M - ⌊sqrt(M)⌋^2 + 1)。對於前面完整的 a(i),它們的和爲(i*2+1)*i,前⌊sqrt(M)⌋的(i*2+1)*i 之和可以用平方和公式推出。計算這兩部分的和即可。

參考代碼

```
/*
  MULTI UNVIERSIY #5 C
  2012-09-13
  ACCEPTED
*/
#include<iostream>
#include<cstdio>
#include<cmath>
#define INF ( long long )1 << 31
#define lint unsigned long long
using std::cin;
using std::cout;
using std::sqrt;

bool sync_with_stdio( bool sync = false );
lint m , cnt , tot;

int find_nth()
{
  lint l = m , r = INF , mid;
  while( 1 )
    {
```

1　直觀的規律，還沒有考慮證明。

```
    mid = l + r >> 1;
    if( l == mid ) break;
    if( mid - ( lint )(sqrt( double( mid ) ) ) < m ) l = mid;
    else r = mid;
   }
  return mid + 1;
}

lint count()
{
 lint n , n0 = ( lint )( sqrt( double( cnt ) ) );
 n = n0 - 1;
 return ( n * ( n + 1 ) * ( 2 * n + 1 ) / 3 + n * ( n + 1 ) / 2 + ( n0 * ( cnt - n0 * n0
+ 1 ) ) );
}

int main()
{
 cin>>tot;
 while( tot -- )
   {
    cin>>m;
    cnt = find_nth();
    cout<<cnt<<" "<<count()<<"\n";
   }
 return 0;
}
```