# Yandex Algorithms 2011 R1 D "Sum Of Medians" Solution[1]


write by Gestalti Lur
2012-09-17
題目鏈接：http://www.codeforces.com/problemset/problem/85/D

## 題目大意

對於一個元素序列{a}有三種操作：add x(0<x<=10^9) ,在序列後加入一個元素 x,del x 刪除集合中的一個元素 x，sum 输出 sum( a_i | i mod 5≡3)的值。操作數量少於 10^5 個。

## 算法分析

可以用線段樹來處理區間和的操作。如果將一個[1,T]的線段樹的每個節點上記錄了區間裏元素下標 mod 5 ≡ x 的和記作[l,r].s[x]，那麼對於每個詢問操作，只要輸出[1,T].s[3]即可。維護這些值的時候可以想到對於一個區間[l,r]，他的左孩子範圍的元素下標 mod 5 之後的結果在[l,r]上也是一樣的，右孩子[mid+1,r]的下標和在[l,r]的情況則是左孩子裏的元素個數加上右孩子當前節點的下標，因此可以推出對於[l,r].s[x]在其右孩子的部分是[mid+1,r].s[ ( x - [l,mid].cnt ) mod 5 ](cnt 爲區間內元素的個數)。

所以顯然有[l,r].s[x]=[l,mid].s[x]+[mid+1,r].s[ (( x - [l,mid].cnt ) mod 5 + 5 )mod 5 ](具體實現的時候因爲 x-[l,mid].cnt 會有負數的情況所以需要加成正數取模)

其次因爲元素大小至多會有 10^9,不能直接建立線段樹。所以可以採用“離散化”的處理方法：首先讀入所有的操作，可以預處理出所有的 add 操作一共有多少種不同的元素以及每個元素的大小，設這個值爲 T，那麼線段樹中就至多會有 T 個元素。先建立一個[1,T]的線段樹，然後對於進行操作的節點 x，可以用二分或者 std::lower_bound()找出它在線段樹當中的位置對其進行操作。

## 參考代碼

```
/*
  CF 85 D
  gestapolur
  ACCEPTED
  ICPC 2012 Chengdu Online A were same as this prob
*/
#include<cstdio>
#include<algorithm>
#include<cstring>
#define MAXN 100005
#define lint long long
struct treenode
{
  int l , r , cnt;
  lint sum[ 5 ];
};
```

1 ICPC 2012 Chengdu Online A 題與此題相同。

```
treenode node[ MAXN << 4 ];
int a[ MAXN ] , q[ MAXN ];
int n , tot;
void create( int s , int l , int r )
{
 memset( node[ s ].sum , 0 , sizeof( node[ 0 ].sum ) );
 node[ s ].l = l;
 node[ s ].r = r;
 node[ s ].cnt = 0;
 if( l < r )
   {
    int mid = l + r >> 1;
    create( s << 1 , l , mid );
    create( s << 1 | 1 , mid + 1 , r );
   }
 return ;
}
void upload( int s )
{
 for( int i = 0 ; i < 5 ; ++ i )
   node[ s ].sum[ i ] = node[ s << 1 ].sum[ i ] + node[ s << 1 | 1 ].sum[ ( ( i -
node[ s << 1 ].cnt ) % 5 + 5 ) % 5 ];
 return ;
}
void update( int s , int pos , int val , int k )
{
 k ? ++ node[ s ].cnt : -- node[ s ].cnt;
 if( node[ s ].l == node[ s ].r )
   {
    node[ s ].sum[ 0 ] = k ? val : 0;
    return ;
   }
 int mid = node[ s ].l + node[ s ].r >> 1;
 if( pos <= mid ) update( ( s << 1 ) , pos , val , k);
 else update( ( s << 1 | 1 ) , pos , val , k );
 upload( s );
 return ;
}

void init()
{
 int i;
 char ch[ 3 ];
 tot = 0;
 for( i = 1 ; i <= n ; ++ i )
   {
    scanf( "%s" , ch );
    switch( ch[ 0 ] )
```

```
    {
    case 'a':
      scanf( "%d" , &q[ i ]);
      a[ tot ++ ] = q[ i ];
      break;
    case 's':
      q[ i ] = 0;
      break;
    case 'd':
      scanf( "%d" , &q[ i ] );
      q[ i ] = -q[ i ];
      break;
    }
   }
  std::sort( a , a + tot );
  tot = std::unique( a , a + tot ) - a;
  memset( node[ 1 ].sum , 0 , sizeof( node[ 1 ].sum ) );
  if( tot ) create( 1 , 1 , tot + 1 );
  return ;
}

int bipart( int p )
{
  int l , r , mid;
  for( l = 1 , r = tot , mid = l + r >> 1 ; l != mid ; mid = l + r >> 1 )
    if( p >= a[ mid ] ) l = mid;
    else r = mid;
  return r;
}

void solve()
{
  int pos;
  for( int i = 1 ; i <= n ; ++ i )
    {
     //pos = ( q[ i ] == 0 ? 0 : std::lower_bound( a , a + tot , ( q[ i ] > 0 ? q[ i ] : -q[
i ] ) ) - a );
     pos = ( q[ i ] ? bipart( q[ i ] > 0 ? q[ i ] : -q[ i ] ) : 0 );
     if( q[ i ] )
     update( 1 , pos , ( q[ i ] > 0 ? q[ i ] : -q[ i ] ) , ( q[ i ] > 0 ? 1 : 0 ) );
     else
     printf( "%I64d\n" , node[ 1 ].sum[ 2 ] );
    }
  return ;
}

int main()
{
```

```
  while( scanf( "%d" , &n ) != EOF )
    {
     init();
     solve();
    }
  return 0;
}
```