二分，再二分！

——从 Mobiles(IOI 2001)一题看多重二分

【引言】

二分，作为一种非常重要的思想及方法，在许多方面都有着不可替代的作用。信息学的发展一日千里，使“二分”思想中产生新的思路。本文想就一种新颖的二分思想——多重二分谈谈作者的一些粗浅看法。

【问题】

假设 Tampere 地区被划分成 N*N 个单元，形成一个 N*N 的表格，行、列坐标均为 0 到 N－1，每个单元中有一个移动电话信号发射基地，这些基地不时地报告所在单元的移动电话数的变化情况。写一个程序，接收这些报告并回答一些询问，询问内容是某些矩形区域内目前的移动电话总数。

输入有若干行，每行包含一个整数，代表某种操作，

后跟若干个整数，是此操作的参数，具体如下表：

操作	参数	含义
0	*N*	初始化表格，大小为 N*N，所有单元的值均为 0。此操作只出现一次，且一定是第一条出现的操作。
1	*X Y A*	将 (*X*, *Y*)单元的移动电话数增加 *A*。*A* 可能为正或负。
2	*L T R B*	询问从左上角(*L*,*T*)到右下角(*R*,*B*)的矩形区域内的移动电话总数。
3	无	结束程序，此操作只出现一次，且一定是最后一条出现的操作。

对每个 2 号操作，你要输出一行，包含单个整数，代表当时“询问”操作的答案。

范围限制：

表格大小	$N\times N$	$1\times 1 \le N\times N \le 1024\times 1024$
任意时刻单元的值 *V*	*V*	$0 \le V \le 2^{15}-1$ (= 32767)
每次增加的个数	*A*	$-2^{15} \le A \le 2^{15}-1$ (= 32767)
输入的操作总数	*M*	$3 \le M \le 60002$
所有单元的值之和	*S*	$S \le 2^{30}$

【初步分析】

看到此题，首先会想到一种最简单的算法：开一个 1024*1024 的数组，记录每个单元的移动电话数，开始全部

清零。顺次读入指令，进行操作，“更改”操作只需要直接改动数组上的相应单元，“询问”操作则必须察看从左上角到右下角的所有单元，把其中的移动电话数累加起来并输出。

这种算法的优点是简单自然，便于实现。缺点是时间复杂度过高，每条“更改”操作的处理时间是常数级，但每条“询问”操作的处理时间却高达 $O(n^2)$级。在最坏情况下，这种算法的时间复杂度为 $O(m* n^2)=60000*1024*1024$，显然无法在规定时间内出解。

【二分方法】

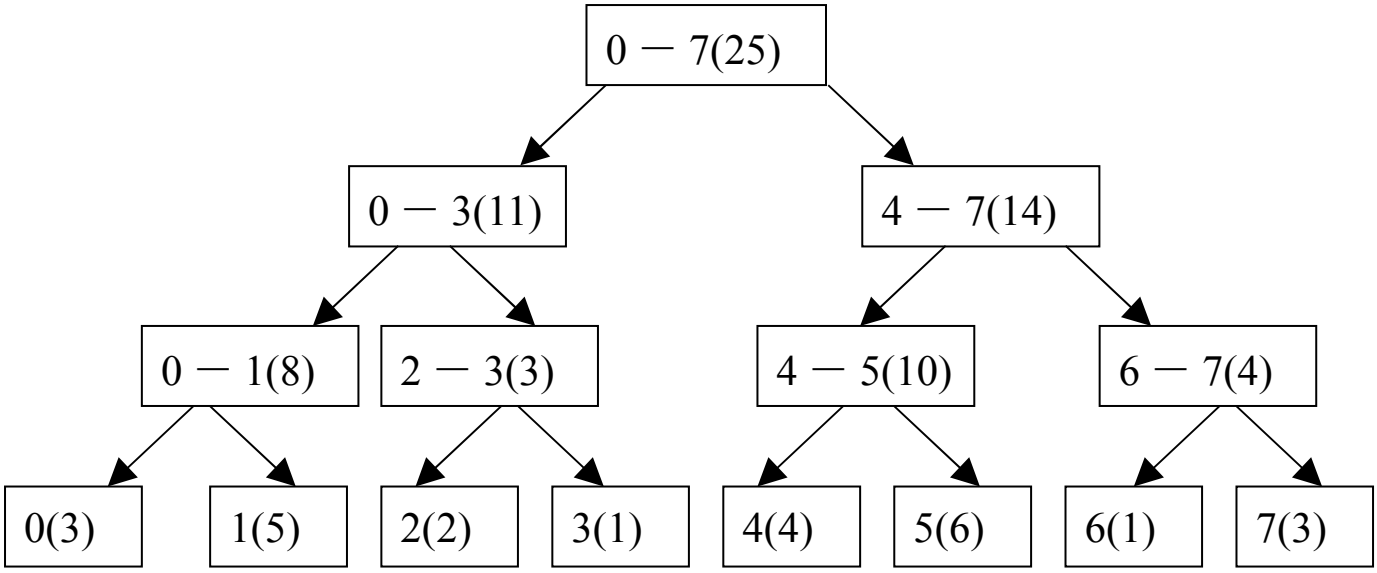
为了便于分析，我们把问题简化一下，降一维来看：只考虑表格中的一行，“更改”操作改动此行中某单元的移动电话数，“询问”操作查询从 L 单元至 R 单元共有多少移动电话。

如果用前面所说的简单算法，则“更改”操作时间复杂度为 O(1)，“询问”操作时间复杂度为 O(n)，总体时间复杂度为 O(m*n)，是很差的。

为了改进，很自然地，我们得到一种**二分**的思想：将1至n的n个单元划分成两段，再对每段继续二分下去，并记录每一段移动电话数的总和，形成一种**树状结构**。

例如：下面一行单元对应如图所示的分法，括号中的数字代表每一段的移动电话总数：

坐标	0	1	2	3	4	5	6	7
移动电话数	3	5	2	1	4	6	1	3



二分之后，“更改”操作需要修改从“根”到“叶”的路径上的各段的移动电话数，因此其时间复杂度为$O(\log_2 n)$。“询问”操作如何实现呢？

注意，(L－R)这一段中的移动电话总数，等于(0－R)段的移动电话总数减去(0－(L－1))段的移动电话总数。而(0－X)段的移动电话总数，可以按照如下方法得到：Total初始为0，从“根”出发，向下查找X，若某一步是向右走，

则将左子树整段的移动电话数加入 Total，最后找到 X，再将“叶”节点上记录的移动电话数加入 Total，此时的 Total 值即为(0－X) 段的移动电话总数。

这样，“询问”操作的时间复杂度也为 $O(\log_2 n)$，整个算法时间复杂度仅为 $O(m*\log_2 n)$，是很优的。

【多重二分】

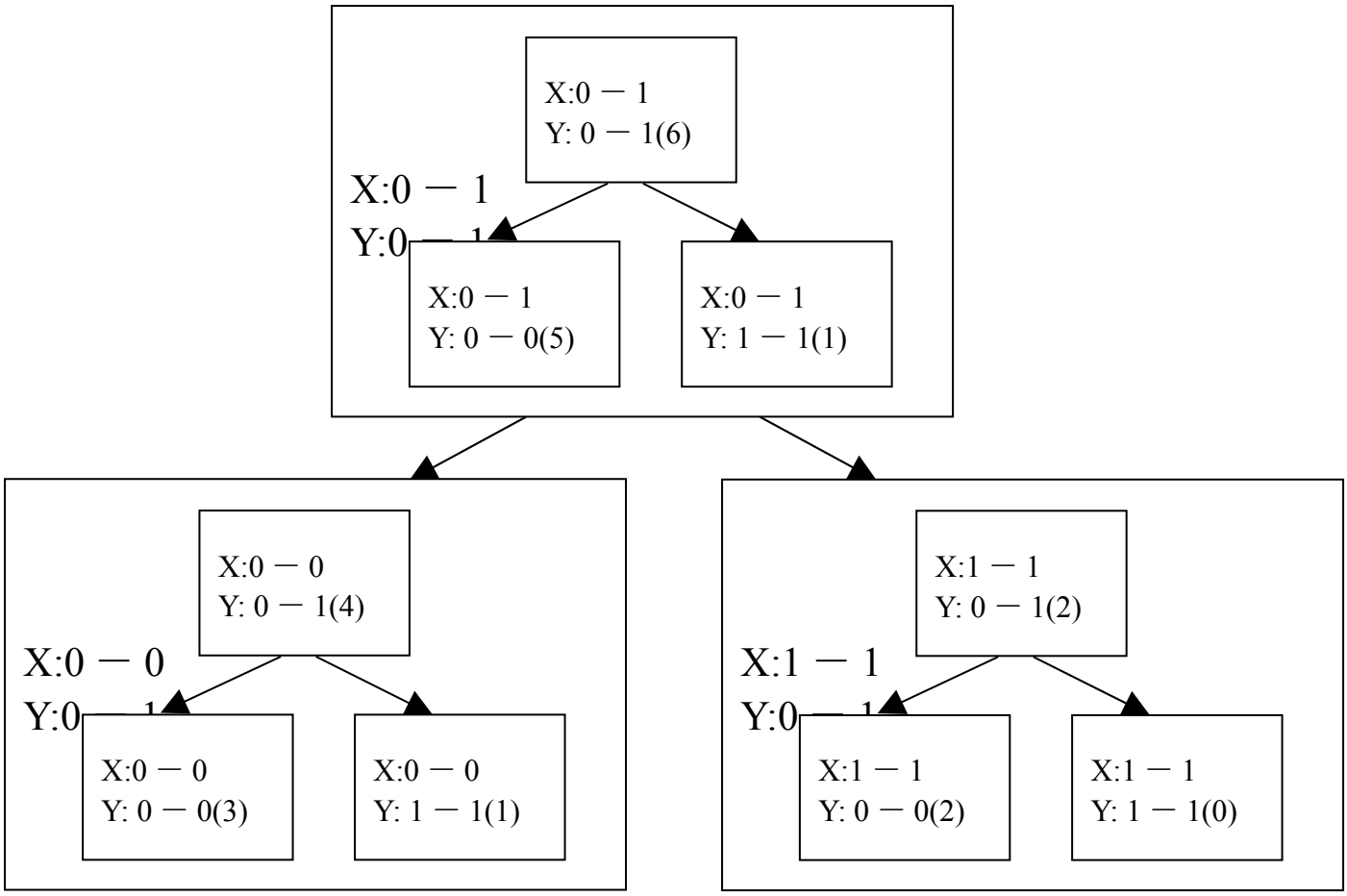
上面分析的是一维的情况，以二分思想为指导，得出了一种比较优的算法。这种算法能否推广到二维的情况呢？可以的。

思考一下，一维情况下，“二分”的对象是“段”，或者可以看成“线”，“分”的标准是 X 坐标。如果推广到二维，那么“二分”的对象理应是“面”，也即表格，“分”的标准应该是 X、Y 坐标同时考虑，此时“二分”涉及两个不同的标准，因此我们称之为“**多重二分**”。

如何分呢？首先，我们按照 X 坐标，把整个表格分成部分，并对每个部分按照 X 坐标继续二分下去，同时，我们将分得的每个部分再按 Y 坐标进行二分，并记下最终分得的每个部分的移动电话总数。

例如，下面的表格可以分成右图的情况，括号中的数字代表每一部分的移动电话总数：

X Y	0	1
0	3	2
1	1	0



“多重二分”的结果，实际上类似于一维情况下二分的结果，也是形成一种“树”状的结构，只不过由于是多重二分，所以这棵“树”的每个节点仍然是一棵“树”。

在这棵“二分树”上进行“改动”操作，只要按照 X 坐标从“根”到“叶”，处理路径上每个节点，由于这些节点也是树，所以要按照 Y 坐标从“根”到“叶”，依次

修改路径上的每个节点的数据。这样，处理每个节点的时间复杂度为 $O(\log_2 n)$，每次修改需要处理的节点数为 $\log_2 n$，所以每次“改动”操作的时间复杂度为 $O(\log_2 n * \log_2 n)$。

“询问”操作的实现，可以模仿一维情况下的算法。首先从左上角（L,T）到右下角（R,B）的矩形区域内的移动电话总数等于(0,0)到(R,B)的移动电话数减(0,0)到(L－1,B)的移动电话数，再减(0,0)到(R,T－1)的移动电话数，再加(0,0)到(L－1, T－1)的移动电话数。

0,0						
					R,T－1	
		L,T				
	L－1,B				R,B	

$$[(L,T),(R,B)] = [(0,0),(R,B)] - [(0,0),(R,T-1)] - [(0,0),(R,T-1)] + [(0,0),(R-1,T-1)]$$

而从(0,0)到(X,Y)的矩形区域内的移动电话总数，也可以仿照一维情况下的算法，Total 初始为 0，在“二分树”上从“根”到“叶”查找 X，如果某步是向右子树走，则须处理相应的左子树的根节点，直到找到 X，再处理此叶节点。

以 X 坐标为标准的“二分树”上的每个节点同时也是一棵以 Y 坐标为标准的“二分树”，对其处理的方法为：从“根”到“叶”查找 Y，如果某步是向右子树走，则将相应左子树整个部分的移动电话总数加入 Total，最后找到 Y，将叶节点上的数值加入 Total。

经过这样的处理之后，得到的 Total 值就是从(0,0)到(X,Y)的矩形区域内的移动电话总数，在 X 坐标为标准的“二分树”上处理每个节点的时间复杂度为 $O(\log_2 n)$，每次这样的统计需要处理 $\log_2 n$ 个这种节点，而每次“询问”操作需要进行四次这种统计，所以“询问”操作的时间复杂度为 $O(\log_2 n * \log_2 n)$。

因此，整个算法的时间复杂度为 $O(m*\log_2 n * \log_2 n)$，是很快的，足以在时限内出解。

【总结】

通过对 Mobiles 这题解法的探讨，我们对“二分”这种思想和方法有了更深刻的认识。一般来说，“二分”方法往往构造出一棵“二分树”。如果“分”的标准只有一个，那么这棵“二分树”的节点就是与待处理的对象直接相关的数据；当处理对象涉及到几种不同的标准时，我们往往可以用“多重二分”，这时，“二分树”的节点就应该是一棵次级“二分树”，可以设想，对象涉及到 Z 个不同的二分标准，就有 Z 级二分树，对对象进行操作的时间复杂度就是 $O[(\log_2 n)^Z]$，由“多重二分”思想得出的算法，应该是很优秀的。

可见，当我们面临的问题适于用二分解决，而“分”的标准又有多个的时候，我们应该想到多重二分，它是两

种重要思想——“二分”思想和“降维”思想结合的产物可以有效地降低时间复杂度。不要犹豫了，二分，再二分！

【附】

- Mobiles 原题如下，本文为讨论方便，翻译时有改动：

Mobile phones

PROBLEM

Suppose that the fourth generation mobile phone base stations in the Tampere area operate as follows. The area is divided into squares. The squares form an $S \times S$ matrix with the rows and columns numbered from 0 to S-1. Each square contains a base station. The number of active mobile phones inside a square can change because a phone is moved from a square to another or a phone is switched on or off. At times, each base station reports the change in the number of active phones to the main base station along with the row and the column of the matrix.

Write a program, which receives these reports and answers queries about the current total number of active mobile phones in any rectangle-shaped area.

INPUT AND OUTPUT

The input is read from standard input as integers and the answers to the queries are written to standard output as integers. The input is encoded as follows. Each input comes on a separate line, and consists of one instruction integer and a number of parameter integers according to the following table.

Instruction	Parameters	Meaning
0	S	Initialize the matrix size to $S \times S$ containing all zeros. This instruction is given only once and it will be the first instruction.
1	$X\ Y\ A$	Add A to the number of active phones in table square (X, Y). A may be positive or negative.
2	$L\ B\ R\ T$	Query the current sum of numbers of active mobile phones in squares (X,Y), where $L \le X \le R$, $B \le Y \le T$
3		Terminate program. This instruction is given only once and it will be the last instruction.

The values will always be in range, so there is no need to check them. In particular, if *A* is negative, it can be assumed that it will not reduce the square value below zero. **The** indexing starts at 0, e.g. for a table of size 4× 4, we have $0 \leq X \leq 3$ and $0 \leq Y \leq 3$.

Your program should not answer anything to lines with an instruction other than 2. If the instruction is 2, then your program is expected to answer the query by writing the answer as a single line containing a single integer to standard output.

PROGRAMMING INSTRUCTIONS

In the examples below, the integer `last` is the last one to be read from a line, and `answer` is the integer variable containing your answer.

If you program in C++ and use `iostreams`, you should use the following implementation for reading standard input and writing to standard output:

```
cin>>last;
cout<<answer<<endl<<flush;
```

If you program in C or C++ and use scanf and printf, you should use the following implementation for reading standard input and writing to standard output:

```
scanf ("%d", &last);
printf("%d\n",answer); fflush (stdout);
```

If you program in Pascal, you should use the following implementation of reading standard input and writing to standard output:

```
Read(last); ... Readln;
Writeln(answer);
```

EXAMPLE

stdin	stdout	explanation
`0 4`		Initialize table size to 4× 4.
`1 1 2 3`		Update table at (1,2) with +3.
`2 0 0 2 2`		Query sum of rectangle $0 \leq X \leq 2$, $0 \leq Y \leq 2$.
	`3`	Answer the query.

```
1  1  1  2          Update table at (1, 1) with +2.
1  1  2  -1            Update table at (1, 2) with -1.

2  1  1  2  3          Query sum of rectangle 1 ≤ X ≤ 2,  1 ≤ Y ≤  3.

           4       Answer the query.
3                  Terminate program.
```

CONSTRAINTS

Table size	$S \times S$	$1 \times 1 \le S \times S \le 1024 \times 1024$
Cell value V at any time	V	$0 \le V \le 2^{15}-1$ (= 32767)
Update amount	A	$-2^{15} \le A \le 2^{15}-1$ (= 32767)
No of instructions in input	U	$3 \le U \le 60002$
Maximum number of phones in the whole table	M	$M = 2^{30}$

Out of the 20 inputs, 16 are such that the table size is at most 512× 512.

NOTE: The web test facility feeds your input file to your program's standard input.

- Mobiles.pas 程序如下：

```
var
   index        :array[0..1023,0..15] of integer;
   s            :array[0..1023,0..1023] of longint;
   n,cmd        :integer;


procedure init;
var
   m,low,up,mid :integer;
begin
    read(m);
    n:=1;
    while n<m do n:=n shl 1;
    dec(n);
    fillchar(index,sizeof(index),0);
    for m:=0 to n do
    begin
        low:=0;up:=n;
```

```
        while low<up do
        begin
            if up=m then break;
            mid:=(low+up) shr 1;
            if mid<m then begin inc(index[m,0]);index[m,index[m,0]]:=mid;low:=mid+1;
                         end
              else begin inc(index[m,0]);index[m,index[m,0]]:=up;up:=mid;
                   end;
        end;
        inc(index[m,0]);index[m,index[m,0]]:=m;
    end;
end;

procedure update;
var
   x,y,d,i,j    :integer;
begin
    read(x,y,d);
    for i:=1 to index[x,0] do
     if index[x,i]>=x then
      for j:=1 to index[y,0] do
        if index[y,j]>=y then inc(s[index[x,i],index[y,j]],d);
end;

function rect(x,y:integer):longint;
var
   i,j          :integer;
   total        :longint;
begin
    if (x<0) or (y<0) then begin rect:=0;exit;
                           end;
    total:=0;
    for i:=1 to index[x,0] do
      if index[x,i]<=x then
        for j:=1 to index[y,0] do
          if index[y,j]<=y then inc(total,s[index[x,i],index[y,j]]);
    rect:=total;
end;

procedure query;
var
   left,bottom,right,top:integer;
begin
    read(left,top,right,bottom);
```

```
    writeln(rect(right,bottom)-rect(left-1,bottom)-rect(right,top-1)+rect(left-1,top-1));
end;

BEGIN
    repeat
       read(cmd);
       case cmd of
           0:init;
           1:update;
           2:query;
       end;
    until cmd=3;
END.
```