P3613 【深基15.例2】寄包柜

题目描述

超市里有 n($n \leqslant 10^5$) 个寄包柜。每个寄包柜格子数量不一，第 i个寄包柜有a_i($a_i \leqslant 10^5$) 个格子，不过我们并不知道各个a_i的值。对于每个寄包柜，格子编号从 1 开始，一直到a_i。现在有q($q \leqslant 10^5$) 次操作：

1 i j k：在第i个柜子的第 j 个格子存入物品 k($0 \leqslant k \leqslant 10^9$)。当 k=0 时说明清空该格子。

2 i j：查询第i个柜子的第 j 个格子中的物品是什么，保证查询的柜子有存过东西。

已知超市里共计不会超过 10^7个寄包格子，a_i是确定然而未知的，但是保证一定不小于该柜子存物品请求的格子编号的最大值。当然也有可能某些寄包柜中一个格子都没有。

输入格式

第一行 2 个整数 n 和 q，寄包柜个数和询问次数。

接下来 q 个整数，表示一次操作。

输出格式

对于查询操作时，输出答案。

输入样例

```
5 4
1 3 10000 114514
1 1 1 1
2 3 10000
2 1 1
```

输出样例

```
114514
1
```

解析

我们先来思考一下，如果用普通的数组该怎么做？

我们首先需要计算一下数组的上限。最多有10^5个寄包柜，那么我们就需要使用一维来存储寄包柜的索引。接下来，每个寄包柜最多还拥有10^5个格子，那么我们就还需要开辟出来一维数组用来存储格子的索引，只有这样，我们才能通过**array[寄包柜索引][格子索引]**找到指定位置的物品。

综上，我们只需要开辟一个array[10^5][10^5]的二维数组即可。

到这里看似一切正常，但是事实真的如此吗？我们来计算一下：

已知10^5 = 100000，那么两个维度的总和就是 100000 * 100000 = 10000000000 = 10^{10}

当前是int数组，每个数字占4字节，因此当前数组所占的总字节数为4 x 10^{10}字节(b)。

看到这个数字还是没有概念吧？我们继续转化，b到Kb的进制是2^{10}，约为1024，我们直接就当它是1000好了，直接用总字节数除以1000，答案为4 x 10^7Kb。

继续转化，得到4 x 10^4Mb = 40Gb！怎么样，恐怖吧？单单这个二维数组所占的空间就是40Gb，要知道一个windows操作系统所占的空间也不过如此，更何况考试时总共给你的空间不过65Mb！很明显我们无法使用普通的二维数组，因此vector则是一个不错的选择。

另外，还需要注意的一点是：题目中明确的给了寄包柜的数量n，但是每个寄包柜有多少个格子我们是不知道的。由于题目中可能会随机操作某个寄包柜的某个格子，这对于我们来说便是可能会读写一个超过当前数组上限的索引，这时该怎么办呢？不要慌，难道你忘记resize方法了吗？

我们可以先用size()判断一下当前数组的长度，如果小于目标值，则只需要调用resize方法修改一下数组的大小即可。

编码

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  //n个存包柜,q次询问
    int n, q;
  cin >> n >> q;
  //寄存柜
    //二维vector
  //n+1的本质，是我们用1号索引来存储第一个柜子
    //第一维：存储柜子号
```

```
    //第二维：格子号
    //动态数组
    vector<vector<int>> boxes(n + 1);
  //只要满足括号内的条件，就持续执行
    //只要q是大于0的就一直执行
    //0 代表的含义是假
    //大于0，代表的含义是真
    while (q--) {  //3 2 1 0
    int opt, i, j, k;
    cin >> opt;
    //1号操作是存储
        if (opt == 1) {
      //将k物品存储第i个柜子的j号小格，j号存在吗？
            cin >> i >> j >> k; //3 10000
      //关键操作，判断格子数
            int sizeValue = boxes[i].size();
      //当前的寄存柜空间不足
            if (sizeValue < j + 1) {
        //调整存储柜大小
                boxes[i].resize(j + 1);
      }
      //将数据进行存储
            boxes[i][j] = k;
    }
      //2号操作是读取
        else {
      cin >> i >> j;
      cout << boxes[i][j] << endl;
    }
  }
  return 0;
}
```