分組大作戰 (Teams)

問題敘述

專題分組戰局開，吾亦尋盟盼功成。奈何誤入庸才窟，終日喧嚷論歌聲。

旁人竊喜圖摸魚，我組空談進度稀。忍無可忍拋虛語，拍案擲筆憤然離。

轉投小古求新路，幸蒙接納免飄零。他人尚在亂如麻，不知何處是歸營。

我輩今朝圖表滿，舊時廢組似落花。莫笑昨夜尚同組，且看期末定乾坤。

- 涂納 作

下課鐘聲一響，教室裡的氣氛忽然活絡起來。

這不是尋常的下課閒聊，而是一場關於「專題組員」的微妙戰局。

同學們開始呼朋引伴，三三兩兩地湊在一起，沒多久就形成了一個個小團體。

然而，真正的故事才正要展開。

事件 1：合併

「欸，我覺得我們題目跟你們差不多耶，要不要乾脆一起做？」

「好啊，反正人多一點比較不怕做不出來。」

於是，**兩組人很快就決定合併成一個更龐大的團隊**。

群組裡傳來一連串「叮咚」聲，大家互相加好友，打算之後一起討論程式、分配工作。

有人開心地笑著說：「這樣期末報告一定穩了！」

也有人心裡偷偷盤算：「人多我就可以划水了，嘻嘻！」

就這樣，一次又一次的「合併」，讓班上的小組有的變成巨型團隊，有的卻仍然維持著三五人的小規模。

事件 2：跳槽

可惜，並不是所有人都對自己的組滿意。

「……我真的受不了了。」

某天晚上，涂納在群組裡留下一段話。

他的組裡整天吵著要去唱歌，專題進度卻空空如也。忍到現在，他已經快炸開了。

隔天，他在自習室啪地闔上筆電，一臉不耐：

「你們要玩就去玩吧，反正我不想再跟你們混了。」

說完，他乾脆俐落地提著背包，走到另一邊，拉開椅子，直接坐到另一組的桌旁。

「小古，你們還缺人嗎？我想加入你們。」

小古愣了一下，隨即露出有點驚訝又有點尷尬的笑容。

「呃……當然可以啊……」他低聲地說，臉微微泛紅，「歡迎加入～」

被他拋下的組員面面相覷，有人嘆了口氣，有人低聲咒罵，但沒有人敢把他叫回去。

就這樣，涂納正式「跳槽」，**從原本的組員變成另一個團隊的新血**。

這種事情並不只發生一次。有人因為理念不合離開，也有人因為嫌原本組太廢，轉而投靠別組。漸漸地，「跳槽」這件事在這學期變得很普遍。

事件 3：查詢

隨著同學不斷重組，班上的分組情況變得越來越混亂。

「欸欸，你知道涂納現在到底在哪一組嗎？」

「不清楚耶，我記得他上週才剛跳去小古那邊啊？」

「才怪啦，昨天有人說他又跑去跟哈魯同一組了。」

「不對吧，我明明聽說他在小碩那組。」

有時候，甚至連組員自己都一臉迷茫：

「……等等，我現在到底是屬於哪一組？」

「咦？**我們這組還剩多少人啊？**」

分組大戰白熱化

隨著時間推進，這場分組大戰逐漸白熱化。

有的團隊越做越強大，桌上散落著滿滿的筆記、圖表與進度表；

有的團隊卻在不斷的跳槽中，始終維持著搖搖欲墜的狀態。

或許在最後，所有人會組成一個超級大團體；

又或許，會有人孤身一人，成為全班「最自由的靈魂」。

這是一場屬於青春的「分組大作戰」。

輸入說明

第一行輸入兩個整數 $N,\ Q$，表示班上共有 N 位同學，並且接下來會發生 Q 個事件。

同學的編號為 $1 \sim N$，最初每位同學各自單獨一組。

接下來的 Q 行中，每一行代表一個事件，格式如下：

- `1 x y`
  編號為 x 的同學所在小組，與編號為 y 的同學所在小組 $\boxed{\text{合併成同一組}}$。
  1. 由於小組成員經常變動，導致有人不清楚自己所在的小組。
  ⇒ 可能 x 跟 y 在合併前就在同個小組。
  2. 保證 $x \neq y$
- `2 x y`
  編號為 x 的同學 $\boxed{\text{離開原本的小組}}$，並且 $\boxed{\text{單獨加入 } y \text{ 所在的小組}}$。
  1. 當 x 退出後，x 覺得還是原本的好，回到原來的小組。
  ⇒ 可能 x 跟 y 原本在同個小組，x 離開後又加回來原本小組。
  2. 保證 $x \neq y$
- `3 x`
  查詢編號為 x 的同學所在小組的 $\boxed{\text{成員數量}}$。

測資限制

- $1 \leq N,\ Q \leq 200000$
- $1 \leq x,\ y \leq N$

輸出說明

對於每一個事件 `3 x`，輸出一個整數，表示編號為 x 的同學所在小組的 $\boxed{\text{成員數量}}$。

每個查詢結果各自輸出在一行。

範例測資

範例輸入 1

```
4 6
1 1 2
1 3 4
3 1
2 2 3
3 1
3 2
```

範例輸出 1

```
2
1
3
```

恭喜你看完這道題目！

為了表達祝賀，最後附上輸入輸出模板供大家使用 > <

連結：https://reurl.cc/Qaalzq

只須針對有 TODO: 的地方做修改即可！

提示

~~請自行通靈~~

如果對併查集 (Disjoint Set) 有一些基礎認識，

就會發現事件 1 和事件 3 根本小菜一碟，可以輕鬆搞定。

那事件 2 該怎麼做呢？

其實事件 2 可以理解為「先退出原本集合，再加入新集合」。

前半步可以想像成兩個動作：

1. 將原本集合的大小減 1：

    因為 x 退出原本集合了！

2. 建立一個新的節點，取代原本的節點：

給 x 建立一個新節點，新節點是在併查集上原先沒被用到的節點。

(簡單的實作方式是利用併查集 (Disjoint Set) 陣列後方的空節點，再搭配一個新的陣列 `id[]` 來記錄每個元素目前在併查集 (Disjoint Set) 中的索引，並在需要時隨時更新。)

後半步呢？完全跟事件 1 一樣耶！

「哇，原來這麼神奇 !」

是不是有點小興奮了呢？

那就趕快動手寫出這題吧！

— — — — — — 這是分隔線 — — — — — —

如果還沒被上機考擊倒、考完試還有餘裕，甚至還對併查集有興趣的話，

不妨在網路上查查看 $\boxed{\text{持久化併查集}}$，

能幫助你對併查集有更深入的理解。

想挑戰更高難度的話，

也可以去學習 $\boxed{\text{時間線段樹 (Time Segment Tree) 結合可回溯併查集 (Rollback Disjoint Set)}}$，

並用這些方法解出這題。

至於這部分嘛……出題者自己也不是很熟（笑），

有興趣的話，不妨去請教摳疼助教！

Teams

Problem Description

When the dismissal bell rang, the classroom atmosphere suddenly came alive.

This wasn' t the usual after-class chatter, but a subtle battlefield over "project teammates."

Students began calling out to friends, clustering in twos and threes, and before long, small groups started to form.

Yet, the real story was only just beginning.

Event 1: Merging

"Hey, I think our topic is pretty similar to yours. Why don' t we just do it together?"

"Sure, with more people, it' s less likely we' ll fail."

And so, **two groups quickly decided to merge into one larger team.**

Phones buzzed with a flurry of notifications as everyone added each other as friends, planning to discuss code and divide tasks later.

Someone laughed happily: "With this, our final report is basically secured!"

Meanwhile, someone else secretly thought: With more people, I can just slack off, hehe.

Thus, with one "merge" after another, some groups in the class grew into massive teams, while others remained as small clusters of three to five.

Event 2: Switching Teams

Unfortunately, not everyone was satisfied with their group.

"⋯I really can' t take this anymore."

One night, Tuna God left a message in his group chat.

His team spent all their time talking about going to karaoke, while the project document remained completely empty. After holding it in for so long, he was about to explode.

The next day in the study room, he slammed his laptop shut with irritation:

“If you guys just want to play, go ahead. I’ m done with this.”

With that, he decisively grabbed his backpack, walked over to another side of the room, pulled out a chair, and sat down at another team’ s table.

“Percy, do you still need someone? I want to join your group.”

Percy froze for a moment, then gave a smile that was half surprised, half awkward.

“Uh···of course···you’ re welcome to join ” he said quietly, his cheeks slightly flushed.

The abandoned teammates stared at one another—some sighed, some muttered curses —but no one dared to call him back.

And so, Tuna God officially “switched teams,”

leaving his original group to become fresh blood for another.

This didn’ t just happen once. Some left because of disagreements, others because their group was “too useless,” and they wanted to join a stronger one. Gradually, switching teams became the buzzword of the semester.

Event 3: Checking

As students kept reshuffling, the class’s group situation became more and more chaotic.

“Hey, do you know which group Tuna God is in right now?”

“No idea, I thought he just switched to Percy’ s group last week?”

“No way, someone told me yesterday he went to Haru’ s group.”

“That can’ t be right—I heard he’ s in Bro Shuo’ s group.”

Sometimes, even the members themselves looked confused:

“···Wait, which group am I in right now?”

“Huh? **How many people do we even have left in our group?**”

The Group Battle Heats Up

As time went on, this group battle reached a boiling point.

Some teams grew stronger and stronger, their desks piled high with notes, diagrams, and progress charts.

Others, shaken by constant defections, remained fragile and on the verge of collapse.

In the end, perhaps everyone would merge into one massive super-team.

Or perhaps, someone would remain all alone, becoming the class' s "freest soul".

This was a youthful "Battle of the Groups."

Input Format

The first line contains two integers N, Q, representing that there are N students in the class, and that Q events will occur.

The students are numbered from 1 to N, and initially each student is in their own individual group.

The following Q lines each describe an event, in one of the following formats:

- `1 x y`
  The group of student x and the group of student y are $\boxed{\text{merged into one group}}$.
  Since group members change frequently, some students may be unsure which group they belong to.
  ⇒ It is possible that x and y are already in the same group before the merge.
  It is guaranteed that $x \neq y$.
- `2 x y`
  Student x $\boxed{\text{leaves their original group}}$ and $\boxed{\text{joins the group of student } y \text{ as an individual}}$.
  After leaving, x might feel that the original group was better and return to it.
  ⇒ It is possible that x and y were originally in the same group, and after leaving, x rejoined the same group.
  It is guaranteed that $x \neq y$.
- `3 x`
  Query the $\boxed{\text{number of members}}$ in the group that student x currently belongs to.

Constraints

- $1 \leq N,\ Q \leq 200000$
- $1 \leq x,\ y \leq N$

Output Format

For each event `3 x`, output an integer representing the $\boxed{\text{number of members}}$ in the group that student x currently belongs to.

Each query result should be printed on a separate line.

Example Test Case

Sample Input 1

```
4 6
1 1 2
1 3 4
3 1
2 2 3
3 1
3 2
```

Sample Output 1

```
2
1
3
```

Congratulations on finishing this problem!

As a token of celebration, here is an input/output template for you to use > <

Link: https://reurl.cc/Qaalzq

You only need to modify the parts marked with TODO:.

Hint

If you have some basic understanding of the Disjoint Set Union (DSU), you' ll notice that Event 1 and Event 3 are a piece of cake and can be handled easily.

But what about Event 2?

In fact, Event 2 can be understood as:

"first leave the original set, then join a new set."

The first half can be thought of as two steps:

1. Decrease the size of the original set by 1:

   Because x has left its original set!

2. Create a new node to replace the original node:

   Assign a new node for x, which is an unused node in the DSU structure.

(A simple implementation is to use the unused nodes at the back of the DSU array, together with an additional array `id[]` to record the current DSU id of each element, updating it whenever necessary.)

And the second half? It' s exactly the same as Event 1!

"Wow, that' s amazing!"

Are you feeling a bit excited now?

Then go ahead and implement this problem!

— — — — — — This is a seperate line. — — — — — —

If you haven' t been knocked out by the lab exam yet, still have some energy left afterwards, and are even interested in Disjoint Set Union,

you might want to look up $\boxed{\text{Persistent Disjoint Set Union}}$ online.

It can help you gain a deeper understanding of Disjoint Set Union.

If you want to challenge yourself with something harder,

you could also try $\boxed{\text{Segment Tree on Time with Rollback DSU}}$,

and solve this problem using those methods.

As for that part⋯well, even the problem setter isn' t very familiar with it (laughs).

If you' re interested, feel free to ask the TA Colten!