C. Experience

2 seconds

64 megabytes

standard input

standard output

In a novel online game, the players fight the monsters and get the experience, as usual. To fight monsters players join together in raid teams. After the destruction of the monster, all the players of the team get the same amount of experience points. The special feature of the game is that a team cannot be split up and no one can leave a team. The only supported operation is to join two teams together.

Since there are already a lot of people playing the game, you are asked to maintain the experience points of the players.

**Input**

The first line of the input contains two integers *n* and *m* (1 ≤ *n*, *m* ≤ 2·105) — the number of players and the number of queries.

Next *m* lines contain the description of queries, one per line. A query can be of three types:

* join *X* *Y* — join two teams to which players *X* and *Y* belong to (if they are already in the same team, nothing changes).
* add *X* *V* — add *V* (1 ≤ *V* ≤ 100) experience points to each player in a team to which player *X* belongs to.
* get *X* — output the current number of experience points of player *X*.

Initially, each player has 0 experience points and each of the player is in its own team of size one.

**Output**

For each query get *X* output the current number of experience points of player *X* on a separate line.

**Example**

**input**

**Copy**

3 6  
add 1 100  
join 1 3  
add 1 50  
get 1  
get 2  
get 3

**output**

**Copy**

150  
0  
50

#include<bits/stdc++.h>

#define int long long int

#define pb push\_back

#define pp pop\_back

#define pii pair<int,int>

#define vec vector<int>

#define mp make\_pair // DISJOINT SET UNION(DSU)

#define inf 1e10

#define F first

#define S second

using namespace std;

int parent[300001],rak[300001],add[300001];

void make\_set(int v) // making a new set

{

parent[v]=v;

rak[v]=0;

}

int find\_set(int v) // finding the leader of the set containing v

{

if(parent[v]==v)

return v;

else

return find\_set(parent[v]);

}

int get(int v)

{

if(v==parent[v])

return add[v];

return add[v]+get(parent[v]);

}

void union\_set(int u,int v) // DOING UNION BY RANK

{

u=find\_set(u); // finding leader of the set containing u

v=find\_set(v); // finding leader of the set containing v

if(u!=v)

{

if(rak[u]<rak[v])

swap(u,v);

parent[v]=u;

add[v]-=add[u];

if(rak[u]==rak[v])

rak[u]++;

}

}

int32\_t main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

int tt=1;

//cin>>tt;

while(tt--)

{

int n,m;

cin>>n>>m;

for(int i=1;i<=n;i++)

make\_set(i);

for(int i=1;i<=m;i++)

{

string s;

cin>>s;

if(s=="add")

{

int a,b;

cin>>a>>b;

a=find\_set(a);

add[a]+=b;

}

else if(s=="join")

{

int a,b;

cin>>a>>b;

union\_set(a,b);

}

else

{

int x;

cin>>x;

cout<<get(x)<<"\n";

}

}

}

}

1. **#include**<bits/stdc++.h>
2. **#define** **int** **long** **long** **int**
3. **#define** pb push\_back
4. **#define** pp pop\_back
5. **#define** pii pair<**int**,**int**>
6. **#define** vec **vector<int>**
7. **#define** mp make\_pair **// DISJOINT SET UNION(DSU)**
8. **#define** inf 1e10
9. **#define** F first
10. **#define** S second
11. **using** **namespace** std;
12. **int** parent[300001],rak[300001],add[300001];
13. **void** make\_set(**int** v) **// making a new set**
14. {
15. parent[v]=v;
16. rak[v]=0;
17. }
18. **int** find\_set(**int** v) **// finding the leader of the set containing v**
19. {
20. **if**(parent[v]==v)
21. **return** v;
22. **else**
23. **return** find\_set(parent[v]);
24. }
25. **int** get(**int** v)
26. {
27. **if**(v==parent[v])
28. **return** add[v];
29. **return** add[v]+get(parent[v]);
30. }
31. **void** union\_set(**int** u,**int** v) **// DOING UNION BY RANK**
32. {
33. u=find\_set(u); **// finding leader of the set containing u**
34. v=find\_set(v); **// finding leader of the set containing v**
35. **if**(u!=v)
36. {
37. **if**(rak[u]<rak[v])
38. swap(u,v);
39. parent[v]=u;
40. add[v]-=add[u];
41. **if**(rak[u]==rak[v])
42. rak[u]++;
43. }
44. }
45. **int32\_t** main()
46. {
47. ios\_base::sync\_with\_stdio(**false**);
48. cin.tie(NULL);
49. cout.tie(NULL);
50. **int** tt=1;
51. **//cin>>tt;**
52. **while**(tt--)
53. {
54. **int** n,m;
55. cin>>n>>m;
56. **for**(**int** i=1;i<=n;i++)
57. make\_set(i);
58. **for**(**int** i=1;i<=m;i++)
59. {
60. string s;
61. cin>>s;
62. **if**(s==**"add"**)
63. {
64. **int** a,b;
65. cin>>a>>b;
66. a=find\_set(a);
67. add[a]+=b;
68. }
69. **else** **if**(s==**"join"**)
70. {
71. **int** a,b;
72. cin>>a>>b;
73. union\_set(a,b);
74. }
75. **else**
76. {
77. **int** x;
78. cin>>x;
79. cout<<get(x)<<**"\n"**;
80. }
81. }
82. }
83. }