CSES Problem Set

Planets and Kingdoms

Submission details

Task:	Planets and Kingdoms
Sender:	seleneal1996
Submission time:	2021-11-26 20:25:54
Language:	C++17
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.01 s	»
#5	ACCEPTED	0.01 s	»
#6	ACCEPTED	0.15 s	»
#7	ACCEPTED	0.15 s	»
#8	ACCEPTED	0.15 s	»
#9	ACCEPTED	0.15 s	»
#10	ACCEPTED	0.15 s	»

Code

```
//https://cses.fi/problemset/task/1683/
#include <bits/stdc++.h>
class P7{
  public:
  int visit[100005];
  std::stack<int>A;
  std::vector<int> part[100005], sequence[100
  std::vector<std::vector<int>> adj, adj_rev;
  int k=0;
  void first_dfs(int u)
  {
    if (visit[u])
      return;
    visit[u]=1;
    for(auto var:part[u])
      first_dfs(var);
    A.push(u);

  }
```


Graph Algorithms

...
Road Reparation -
Road Construction -
Flight Routes Check -
Planets and Kingdoms ✓
Giant Pizza -
Coin Collector -
Mail Delivery -
De Bruijn Sequence -
...

Your submissions

2021-11-26 20:25:54 ✓
2021-11-26 19:58:35 ✗
2021-11-26 19:55:44 ✗
2021-11-26 18:38:09 ✓

```
  void second_dfs(int u)
  {
    if(visit[u])
      return;
    visit[u]=k;
    for(auto var:sequence[u])
      second_dfs(var);
  }

  void final_solution(){
    int n,m;
    std::cin>>n>>m;
    for (int i=0; i<m; i++){
      int a,b;
      std::cin>>a>>b;
      part[a].push_back(b);
      sequence[b].push_back(a);
    }
    for (int i=1; i<n+1; i++){
      if(!visit[i])
        first_dfs(i);
    }
    memset(visit,0,sizeof visit);
    while(!A.empty())
    {
        int x = A.top();
        A.pop();
        if (!visit[x])
        {
          k++;
          second_dfs(x);
        }
    }
    std::cout << k <<"\n";
    for (int i=1; i<n+1; i++)
      std::cout <<visit[i]<<"\n";
  }

};

int main(){
    std::ios::sync_with_stdio(0);
    std::cin.tie(0);
    P7 S1= P7();
    int t=1;
    while(t--){
      S1.final_solution();
    }
    return 0;
}
```

Share code to others

Test details

Test 1

Verdict: ACCEPTED

input
10 20 4 5

```
10 7
6 1
6 5
...
```

correct output

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

user output

```
4
4
3
1
1
...
```

Test 2

Verdict: ACCEPTED

input

```
10 20
5 6
1 2
7 9
7 3
...
```

correct output

```
2
1 1 1 2 1 1 1 1 1 1
```

user output

```
2
1
1
1
2
...
```

Test 3

Verdict: ACCEPTED

input

```
10 20
1 6
10 1
1 10
9 10
...
```

correct output

```
3
3 2 1 3 3 3 3 3 3 3
```

user output

```
3
3
2
1
3
...
```

Test 4

Verdict: ACCEPTED

input

```
10 20
4 7
10 6
10 5
10 7
...
```

correct output

```
7
2 4 3 1 5 7 5 6 5 5
```

user output

```
7
2
4
3
1
...
```

Test 5

Verdict: ACCEPTED

input

```
10 20
8 1
7 6
6 4
3 1
...
```

correct output

```
3
3 3 3 3 3 3 3 3 1 2
```

user output

```
3
3
3
3
3
...
```

Test 6

Verdict: ACCEPTED

input

```
100000 200000
32402 49159
41650 12290
90019 96038
12320 82053
...
```

correct output

```
43755
26333 26333 40691 34351 26331 ...
```

user output

```
43755
26333
26333
40691
34351
...
```

Test 7

Verdict: ACCEPTED

input

```
100000 200000
98891 58773
74281 97370
25400 8211
25600 1357
...
```

correct output

```
43634
26308 26307 26306 43439 26305 ...
```

user output

```
43634
26308
26307
26306
43439
...
```

Test 8

Verdict: ACCEPTED

input

```
100000 200000
30442 46106
83330 48942
25229 50273
33345 72005
...
```

correct output

```
43498
43496 26032 26031 32415 26033 ...
```

user output

```
43498
43496
26032
26031
32415
...
```

Test 9

Verdict: ACCEPTED

input

```
100000 200000
45422 77478
6800 88602
9724 59882
20954 36466
...
```

correct output

```
44087
26407 26407 26407 26404 26407 ...
```

user output

```
44087
```

```
26407
26407
26407
26404
...
```

Test 10

Verdict: ACCEPTED

input

```
100000 200000
77767 73183
30807 58373
23969 94613
27280 75565
...
```

correct output

```
43903
43903 43902 26255 26256 29892 ...
```

user output

```
43903
43903
43902
26255
26256
...
```