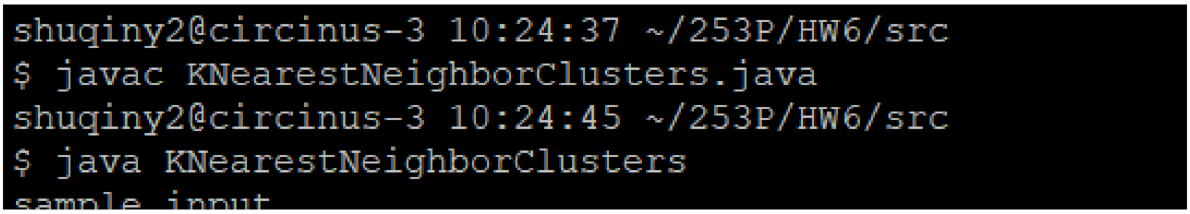
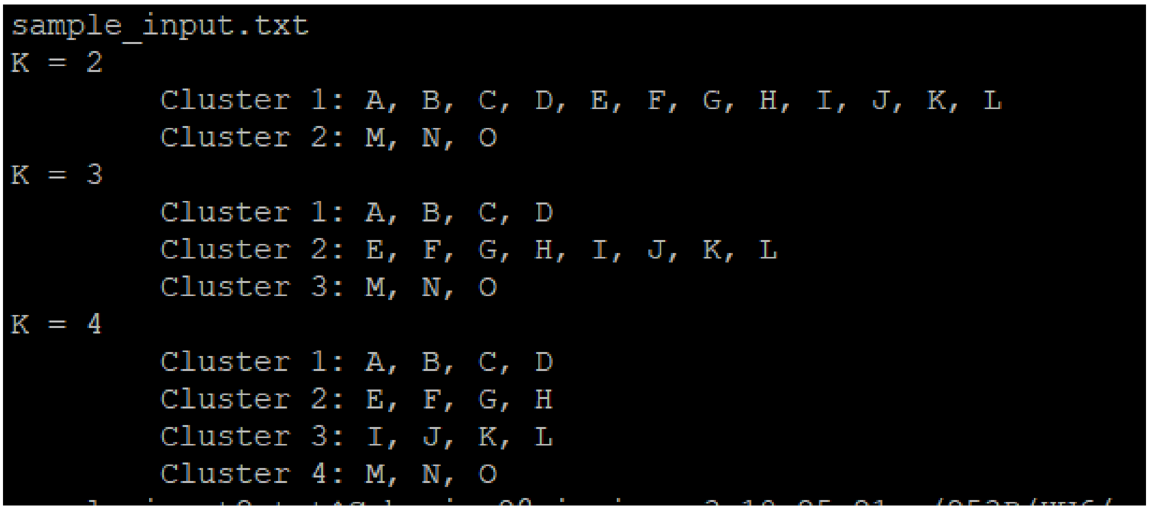
Compile



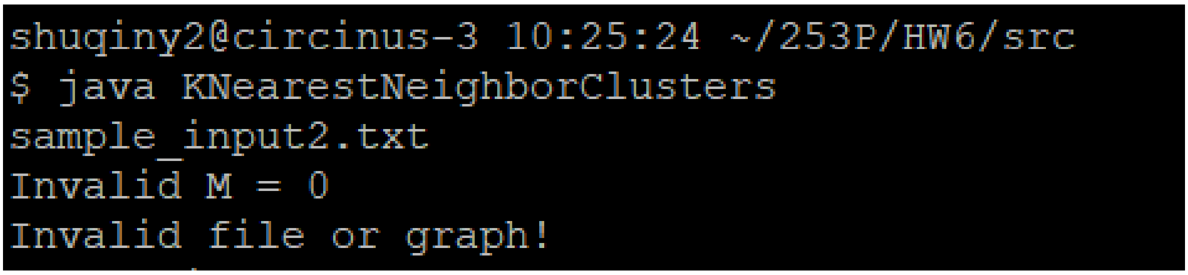
```
shuqiny2@circinus-3 10:24:37 ~/253P/HW6/src
$ javac KNearestNeighborClusters.java
shuqiny2@circinus-3 10:24:45 ~/253P/HW6/src
$ java KNearestNeighborClusters
sample input
```

Input:	output:
15 A 6 2 B 7 3 C 9 3 D 8 5 E 9 8 F 8 9 G 7 10 H 6 11 I 8 14 J 6 14 K 4 14 L 2 14 M 2 8 N 2 6 O 3 5 2 3 4	K=2 Cluster 1: A, B, C, D, E, F, G, H, I, J, K, L Cluster 2: M, N, O K=3 Cluster 1: A, B, C, D Cluster 2: E, F, G, H, I, J, K, L Cluster 3: M, N, O K=4 Cluster 1: A, B, C, D Cluster 2: E, F, G, H Cluster 3: I, J, K, L Cluster 4: M, N, O



```
sample_input.txt
K = 2
        Cluster 1: A, B, C, D, E, F, G, H, I, J, K, L
        Cluster 2: M, N, O
K = 3
        Cluster 1: A, B, C, D
        Cluster 2: E, F, G, H, I, J, K, L
        Cluster 3: M, N, O
K = 4
        Cluster 1: A, B, C, D
        Cluster 2: E, F, G, H
        Cluster 3: I, J, K, L
        Cluster 4: M, N, O
```

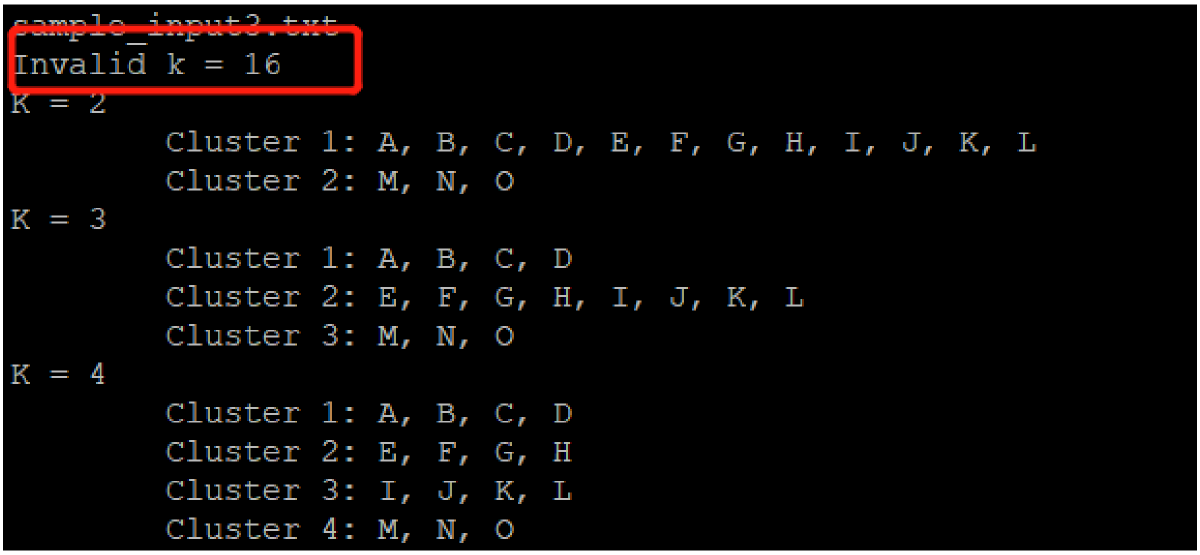
Input:	output:
0	error message



```
shuqiny2@circinus-3 10:25:24 ~/253P/HW6/src
$ java KNearestNeighborClusters
sample_input2.txt
Invalid M = 0
Invalid file or graph!
```

Input:	output:

15
 	
A 6 2
B 7 3
C 9 3
D 8 5
E 9 8
F 8 9
G 7 10
H 6 11
I 8 14
J 6 14
K 4 14
L 2 14
M 2 8
N 2 6
O 3 5
 	
2
3
4
16
	K=2
	    Cluster 1: A, B, C, D, E, F, G, H, I, J, K, L
	    Cluster 2: M, N, O
	K=3
	    Cluster 1: A, B, C, D
	    Cluster 2: E, F, G, H, I, J, K, L
	    Cluster 3: M, N, O
	K=4
	    Cluster 1: A, B, C, D
	    Cluster 2: E, F, G, H
	    Cluster 3: I, J, K, L
	    Cluster 4: M, N, O
	Error message



```
sample_input3.txt
Invalid k = 16
K = 2
        Cluster 1: A, B, C, D, E, F, G, H, I, J, K, L
        Cluster 2: M, N, O
K = 3
        Cluster 1: A, B, C, D
        Cluster 2: E, F, G, H, I, J, K, L
        Cluster 3: M, N, O
K = 4
        Cluster 1: A, B, C, D
        Cluster 2: E, F, G, H
        Cluster 3: I, J, K, L
        Cluster 4: M, N, O
```

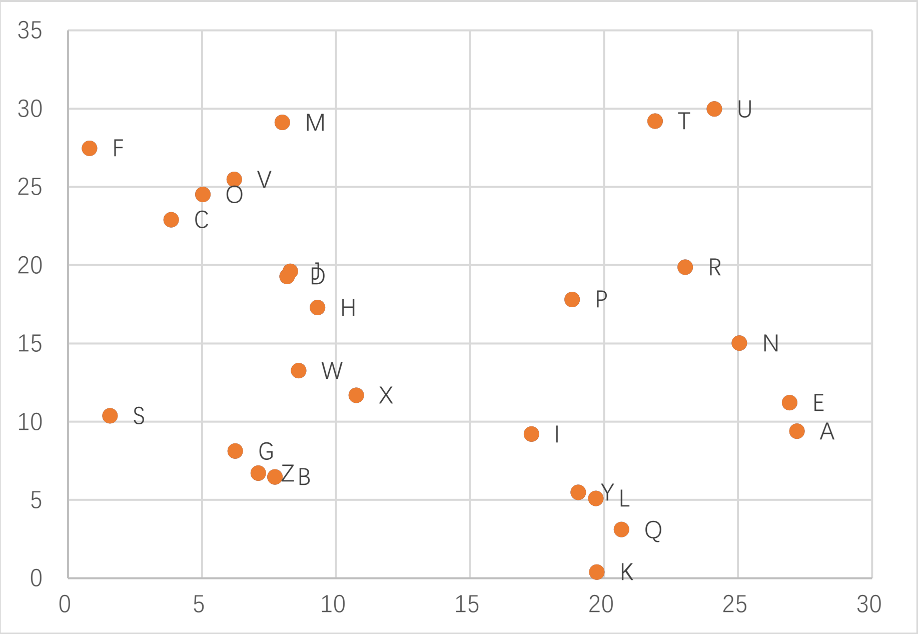
Input:

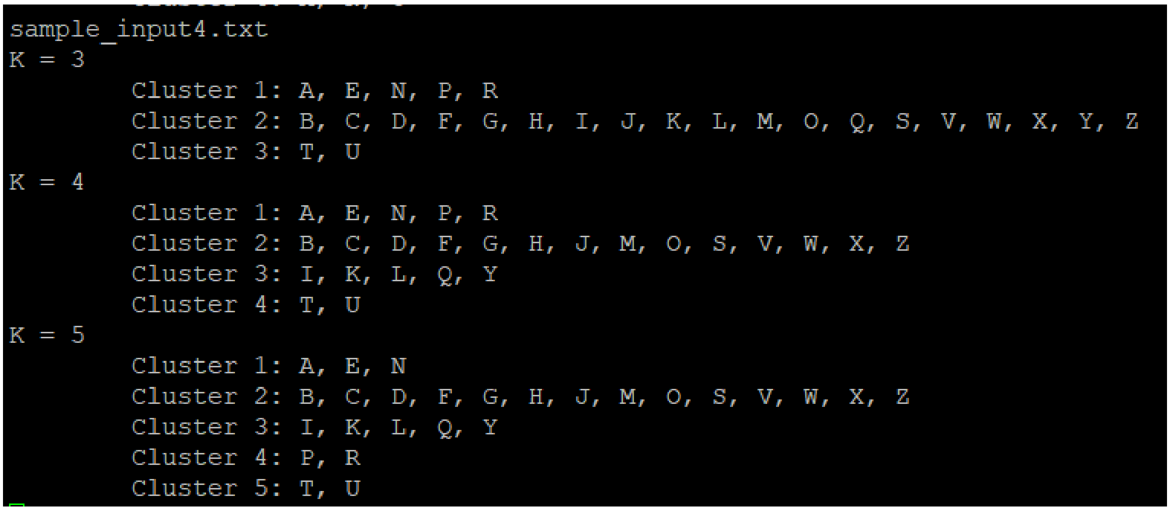
```
26

 A  27   9
 B   8   6
 C   4  23
 D   8  19
 E  27  11
 F   1  27
 G   6   8
 H   9  17
 I  17   9
 J   8  20
 K  20   0
 L  20   5
 M   8  29
 N  25  15
 O   5  25
 P  19  18
 Q  21   3
 R  23  20
 S   2  10
 T  22  29
 U  24  30
 V   6  25
 W   9  13
 X  11  12
 Y  19   5
 Z   7   7

3
4
5
```

output:

```
K=2
    Cluster 1: A, B, C, D, E, F, G, H, I, J, K, L
    Cluster 2: M, N, O
K=3
    Cluster 1: A, B, C, D
    Cluster 2: E, F, G, H, I, J, K, L
    Cluster 3: M, N, O
K=4
    Cluster 1: A, B, C, D
    Cluster 2: E, F, G, H
    Cluster 3: I, J, K, L
    Cluster 4: M, N, O
```

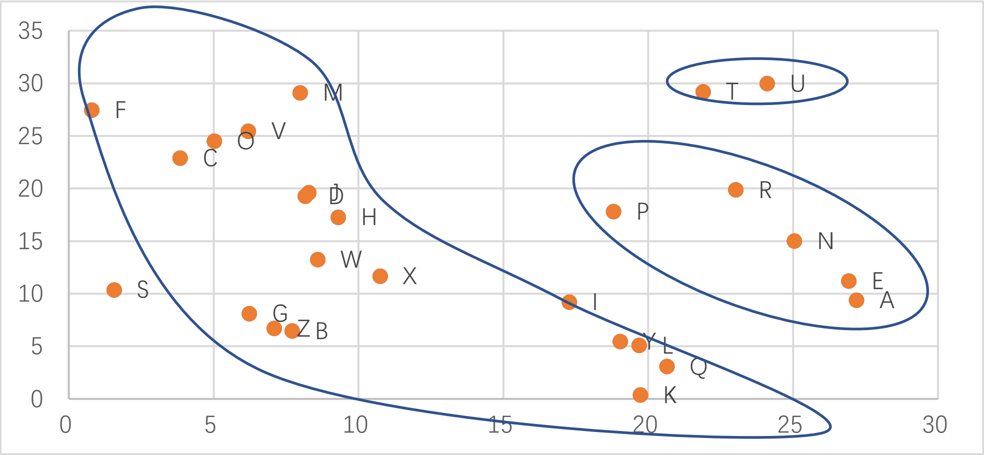
Input:



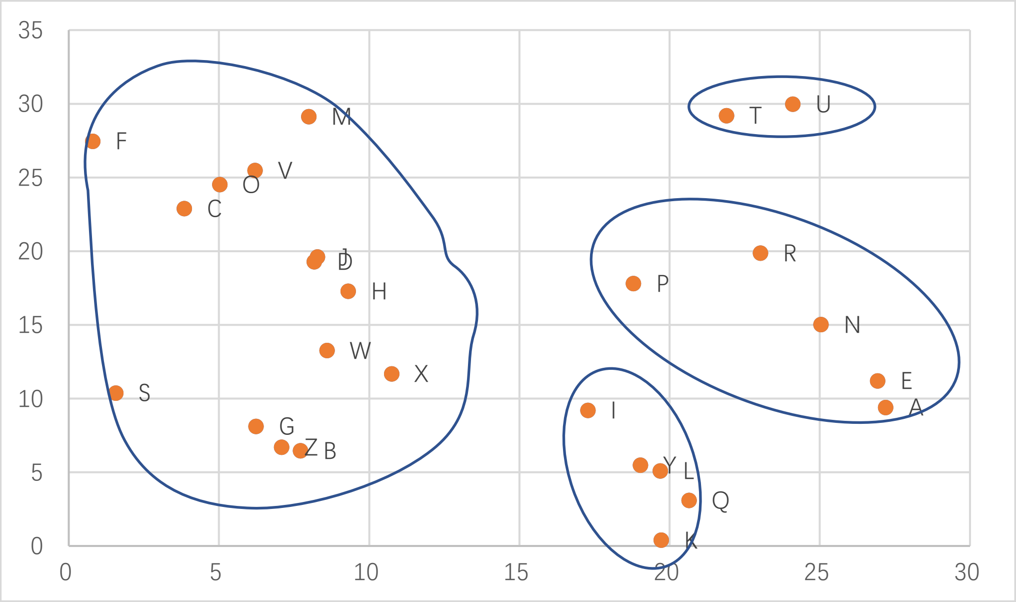


```
sample_input4.txt
K = 3
          Cluster 1: A, E, N, P, R
          Cluster 2: B, C, D, F, G, H, I, J, K, L, M, O, Q, S, V, W, X, Y, Z
          Cluster 3: T, U
K = 4
          Cluster 1: A, E, N, P, R
          Cluster 2: B, C, D, F, G, H, J, M, O, S, V, W, X, Z
          Cluster 3: I, K, L, Q, Y
          Cluster 4: T, U
K = 5
          Cluster 1: A, E, N
          Cluster 2: B, C, D, F, G, H, J, M, O, S, V, W, X, Z
          Cluster 3: I, K, L, Q, Y
          Cluster 4: P, R
          Cluster 5: T, U
```

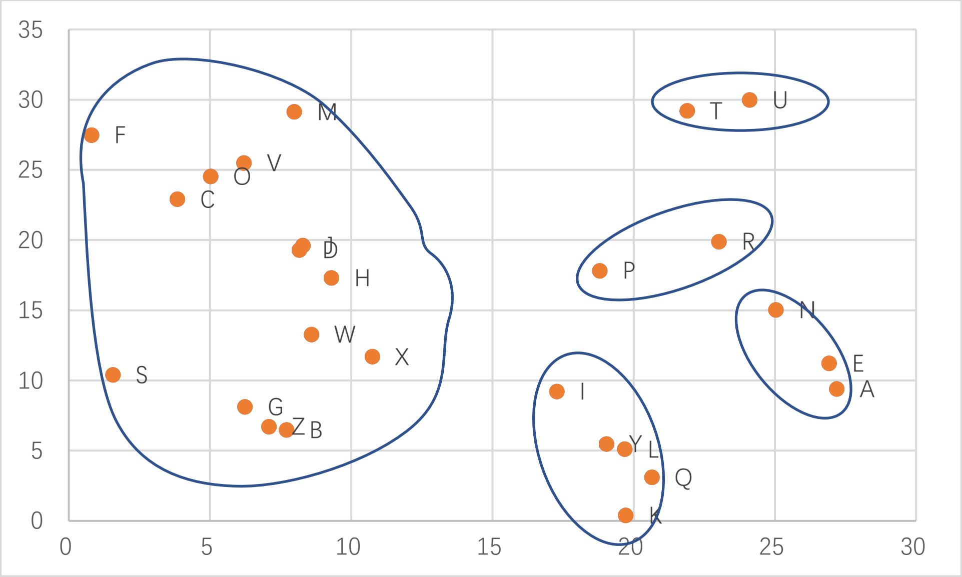
// k = 3



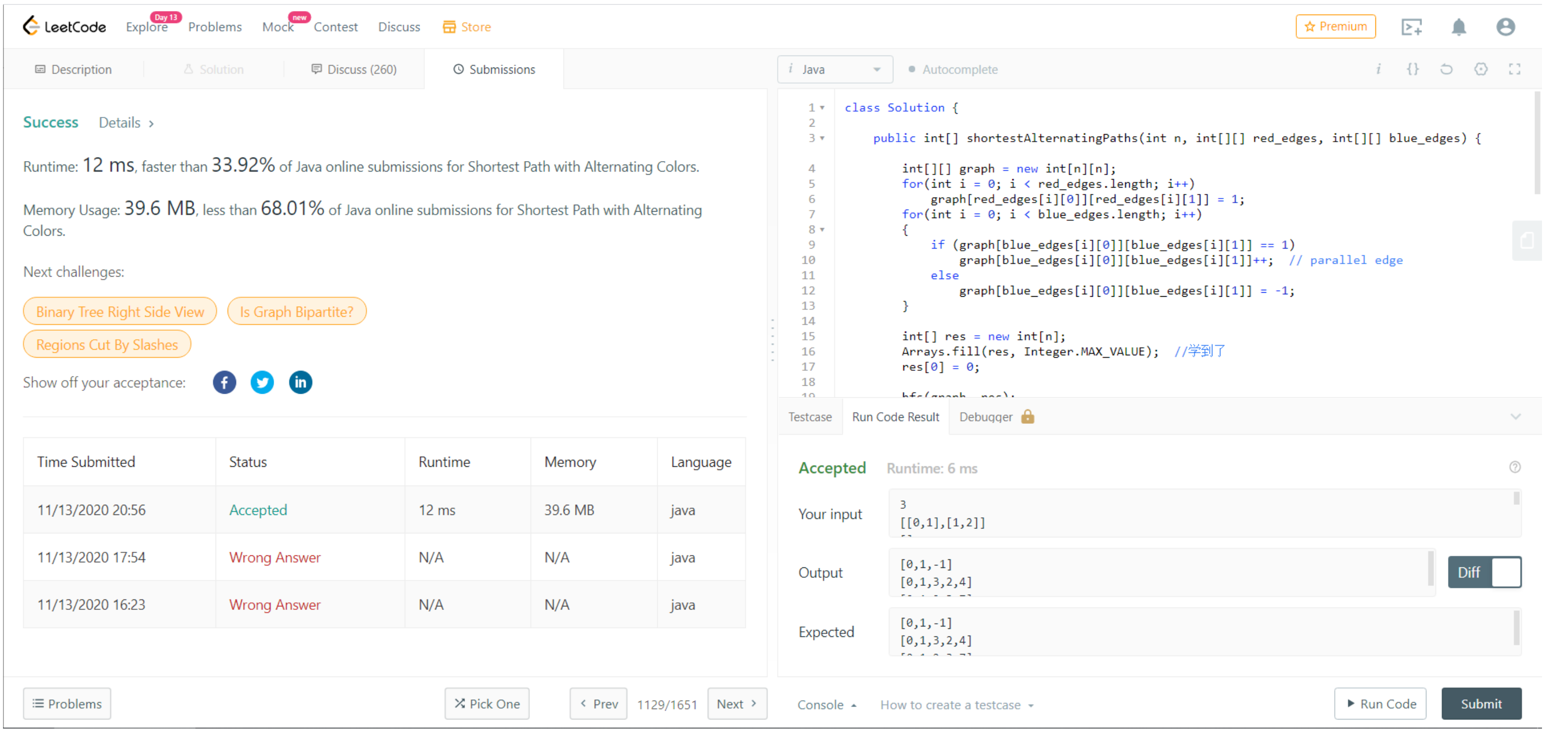
// k = 4



// k = 5


35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
M
F
V
O
C
J
D
H
W
X
S
G
Z
B
T
U
R
P
N
E
A
I
Y
L
Q
K

Leetcode 1129



Success Details ›

Runtime: 12 ms, faster than 33.92% of Java online submissions for Shortest Path with Alternating Colors.

Memory Usage: 39.6 MB, less than 68.01% of Java online submissions for Shortest Path with Alternating Colors.

Next challenges:

Binary Tree Right Side View · Is Graph Bipartite? · Regions Cut By Slashes

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
11/13/2020 20:56	Accepted	12 ms	39.6 MB	java
11/13/2020 17:54	Wrong Answer	N/A	N/A	java
11/13/2020 16:23	Wrong Answer	N/A	N/A	java

```java
class Solution {

    public int[] shortestAlternatingPaths(int n, int[][] red_edges, int[][] blue_edges) {

        int[][] graph = new int[n][n];
        for(int i = 0; i < red_edges.length; i++)
            graph[red_edges[i][0]][red_edges[i][1]] = 1;
        for(int i = 0; i < blue_edges.length; i++)
        {
            if (graph[blue_edges[i][0]][blue_edges[i][1]] == 1)
                graph[blue_edges[i][0]][blue_edges[i][1]]++;  // parallel edge
            else
                graph[blue_edges[i][0]][blue_edges[i][1]] = -1;
        }

        int[] res = new int[n];
        Arrays.fill(res, Integer.MAX_VALUE);  //学到了
        res[0] = 0;

        bfs(graph, res);
```

Accepted Runtime: 6 ms

Your input

```
3
[[0,1],[1,2]]
```

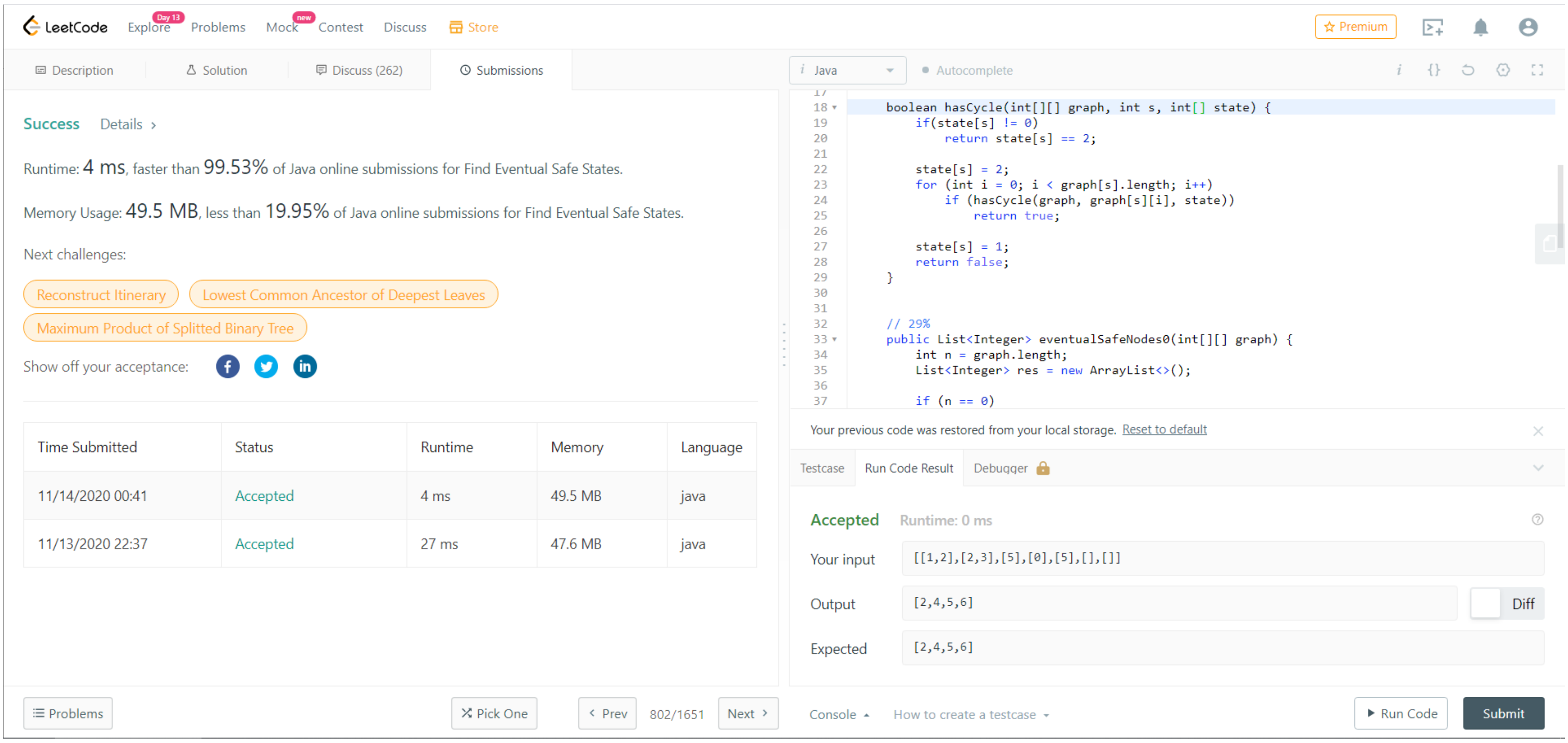
Output

```
[0,1,-1]
[0,1,3,2,4]
```

Expected

```
[0,1,-1]
[0,1,3,2,4]
```

LeetCode 802



Success Details ›

Runtime: 4 ms, faster than 99.53% of Java online submissions for Find Eventual Safe States.

Memory Usage: 49.5 MB, less than 19.95% of Java online submissions for Find Eventual Safe States.

Next challenges:

Reconstruct Itinerary | Lowest Common Ancestor of Deepest Leaves | Maximum Product of Splitted Binary Tree

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
11/14/2020 00:41	Accepted	4 ms	49.5 MB	java
11/13/2020 22:37	Accepted	27 ms	47.6 MB	java

```java
    boolean hasCycle(int[][] graph, int s, int[] state) {
        if(state[s] != 0)
            return state[s] == 2;

        state[s] = 2;
        for (int i = 0; i < graph[s].length; i++)
            if (hasCycle(graph, graph[s][i], state))
                return true;

        state[s] = 1;
        return false;
    }


    // 29%
    public List<Integer> eventualSafeNodes0(int[][] graph) {
        int n = graph.length;
        List<Integer> res = new ArrayList<>();

        if (n == 0)
```

Your previous code was restored from your local storage. Reset to default

Accepted Runtime: 0 ms

Your input `[[1,2],[2,3],[5],[0],[5],[],[]]`

Output `[2,4,5,6]`

Expected `[2,4,5,6]`