Single Source Shortest Path

In this problem, you will solve the single source shortest path problem for a given directed weighted graph with no negative weight edges using Dijkstra’s Algorithm.

The input will consist of a few numbers N, Source, D, C_1 , C_2 , D_1, D_2, W_1 , W_2, W_3.

N = Number of nodes in the graph (numbered 1,2...N)

D = Max outdegree of any node

Source = The source node (between 1 and N inclusive)

C_1, C_2, D_1, D_2, W_1, W_2, W_3 are just some constants,

You have to create the graph using the pseudo code given below:

```
for i = 1 to N :  //Inclusive
 deg = (  i*C_2 + i*i*D_2  ) mod D
 for j = 1 to deg: //Inclusive
  temp_node.vertex = ( i*C_1 + j*D_1 ) mod N
  temp_node.vertex += 1
  temp_node.weight = ( i*W_1 + j*W_2 ) mod W_3  //Weight of edge ( i, temp_node.vertex)
  adj_list [ i ].enqueue( temp_node )
```

You have to print the minimum cost of traversing from the source to all the vertices [1,N]

Note-

1) Do not use Bellman Ford algorithm as it won't work here. It will give you a TLE error.

2) You are supposed to implement the priority queue yourself. You cannot use any prebuilt priority queue , (max/min) heap , map etc functions. However, you can use sort functions , queues for adjacency list etc. Please contact the SPOJ TAs if there is any doubt.

3) Make sure your program uses less than 10^6 * sizeof(long long int) memory. Creating a matrix of size N*N wont work for all the test cases.

4) There might be self loops and multiple edges between two nodes in the graph.

5) Overall time complexity should be O((E + V)lgV), otherwise you will get TLE.

Input

Input consist of 10 space separated integers in the following format.

N Source D C_1 C_2 D_1 D_2 W_1 W_2 W_3

Constraints

1 <= N <= 10^5
N*D <= 10^6
1 <= C_1, C_2, D_1, D_2 <= 10^3

1 <= Source <= N
0 <= W_1, W_2, W_3 <= 10^3

Output

Print the table of shortest path distances from the source vertex to all the vertices. This table should contain N lines. Each line should contain (space separated) vertex-id and distance, successively, for vertex-id = 1, 2, . . . , N. If you cannot reach vertex-id from source, print -1 for distance.

Example

Input:

```
8 2 5 446 192 703 336 56 75 1000
```

Output:

```
1 1103
2 0
3 262
4 187
5 711
6 636
7 561
8 486
```

Input:

```
10 1 4 315 567 647 270 15 35 1000
```

Output:

```
1 0
2 -1
3 50
4 -1
5 385
6 -1
7 200
8 350
9 -1
10 165
```