CS2233: Data Structures
Assignment 6
30th September, 2018

Problem Statement

- Input: A directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{N}$.
- Goal: Serve the following requests:
  - Given $u, v \in V$, what is the weight of edge (u, v)?
  - Given $v \in V$, run Dijkstra's algorithm on G from vertex v.
  - Given $u, v \in V$, output shortest path from a vertex u to v.

Input Format

General format:
A graph $G = (V, E)$ is defined by first providing the number of vertices $n = |V|$.
Henceforth you shall assume that the vertex set is $V = \{1, 2, \ldots, n\}$.
The edge set E and the weight function w are provided together.
Each neighbour v of a vertex u will be given in the adjacency list of u along with $w(u, v)$.
Requests arrive only after the definition of G.
Format in detail:
Each line of the input looks like one of the following:

- 'N' followed by number of vertices $n \in \mathbb{N}$.
- 'E' followed by a vertex and vertex-weight pairs that look like:
  $u,\ v_1, w(u, v_1),\ v_2, w(u, v_2)\ \cdots,\ v_k, w(u, v_k)$
  This list gives the adjacency list of vertex u along with the respective edge weights. The list is given as a space-separated list.
- '?' followed by $u, v \in [n]$ with a space separating them.
- 'D' followed by a $u \in [n]$.
- 'P' followed by $u, v \in [n]$ with a space separating them.

Each of the lines above ends with a '\n' character. All lists are given as elements separated by a space. No commas are used anywhere. All numbers used will fit inside an `int`. End of input is indicated by `EOF`.

Output Format

- If input line started with 'N' or 'E', then no corresponding output.
- If input line was "? u v": Output $w(u, v)$ if $(u, v) \in E$, -1 otherwise.
- If input line was "D u":
  Let $v_1, v_2, \ldots, v_n$ be the order of vertices visited on a run of of Dijkstra's algorithm from u (note that $u = v_1$). Let $\delta(u, v)$ denote the shortest path from u to v. Output a list of pairs: $(v_1, \delta(u, v_1)), \ldots, (v_n, \delta(u, v_n))$. If $\delta(u, v) = \infty$ then output -1 in its place. Use a space to separate v and $\delta(u, v)$ within a pair. Use a '\n' between pairs.
- If input line was "P u v":
  If v is not reachable from u, output -1. Else output the shortest distance from u to v followed by a shortest path from u to v as a space-separated list of vertices starting with u.

All output lines have to end with a '\n' character.

Implementation rules

- The input graph $G = (V, E)$ is stored similar to Assignment 4. The edge weights are stored in an additional variable inside the nodes in the adjacency list (see CLRS).
- Store each vertex's adjacency list in the same order as provided in the input.
- Dijkstra's algorithm requires a min-priority queue. Implement a min-priority queue by using a min-heap.

Other Remarks

- A good starting point is to program a min-priority queue. Test the queue thoroughly to make sure it does not become the source of bugs later on.
- Study the Dijkstra's algorithm to find out where would be the correct place to output a vertex-distance pair. The output to the request "D u" is straightforward after this observation.
- **Deadline:** 20th October, 2018.

Example input

Input:

```
--------------------------
N 5
E 1 2 10 3 5
E 2 3 2 5 1
E 3 2 3 5 9 4 2
E 4 5 6 1 7
E 5 4 4
? 2 1
? 2 3
? 5 4
? 3 5
P 1 5
D 1
P 2 4
P 2 1
? 1 4

--------------------------
```

Output:

```
--------------------------
-1
2
4
9
9 1 3 2 5
1 0
3 5
4 7
2 8
5 9
4 2 3 4
11 2 3 4 1
-1

--------------------------
```

Differently colored numbers indicate edge weights. See next page for the graph given in the above input.

Example graph