Assignment 2 Dijkstra's Algorithm

Implement Dijkstra’s algorithm to solve Single Destination Shortest Path problem. Single destination shortest path is finding shortest path from all the vertices to the given vertex.

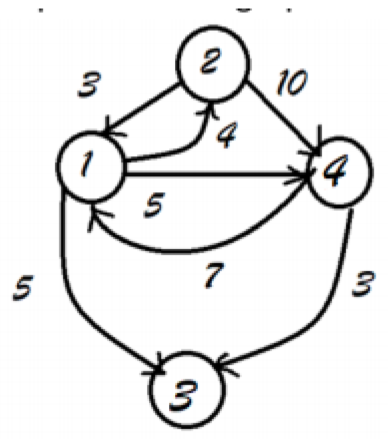
Input file

- Vertices are numbered 1 to n
- First line represents number of vertices
- This is followed by a set of lines
- Each line starts with a integer (any integer from 1 to n in any order) which represents vertex number (v_id) followed by space followed by a set of d pairs where d represents outdegree of the vertex v_id .First number in the pair represents neighbour vertex id and second number weight of the edge which connects vertex to the neighbour. Weight is always a non-negative integer

Sample Input file

```
4
1 2 4 3 5 4 5
4 1 7 3 3
2 1 3 4 10
```

Representation



Code Documentation

Graph

```
typedef struct vertex{
    int id;
    int weight;
    struct vertex *next;
}Vertex;

typedef struct adjacencyList{
    Vertex *head;
}AdjacencyList;

/*
 * Initialises adjacency list with n nodes
 * */
Graph_t* Graph_Init(int n);

/*
 * Appends a directional connection with a particular
 * DOES NOT HEAPIFY
 * */
void Graph_Append(Graph_t* g, int source, int destination, int weight);

/*
 * Frees all memory associated with the graph
 * Including the adj list
 * */
void Graph_Destroy(Graph_t* g);
```

MinHeap

```
typedef struct heapnode {
    int id;
    int distance;
    int predecessor; // Holds the id of the predecessor towards the
destination
    // Used to print out the entire path from the node to the destination
} Heap_Node;

typedef struct minheap{
    int currentSize;
    int capacity;
    int *positionArray; // This array holds the position of the various
vertices in the heap
    Heap_Node *heap;
} MinHeap_t;

/*
 * Initialises heap with n slots
 * Including the position array
 * */
MinHeap_t* Heap_Init(int n);

/*
 * Frees heap
 * Including the position array
 * */
```

```
void Heap_Destroy(MinHeap_t* h);

/*
 * Frees heap
 * Including the position array
 * */
void Heap_Insert(MinHeap_t* h, int id, int distance, int predecessor);

/*
 * Deletes 0th (smallest) element fromt heap
 * HEAPIFIES TOP DOWN
 * */
int Heap_Delete(MinHeap_t* h);

/*
 * Deletes 0th (smallest) element fromt heap
 * HEAPIFIES BOTTOM UP
 * */
void Heap_Update(MinHeap_t* h, int index, int newDistance, int
newPredecessor);

/*
 * Returns the position of a particular id
 * */
int Heap_Search(MinHeap_t* h, int id);

/*
 * Returns the current size of the heap after deletions
 * */
int Heap_Size(MinHeap_t* h);

/*
 * Returns the distance of a particular id from the destination
 * */
int Heap_GetDistance(MinHeap_t* h, int index);
```

Dijkstra

```
/*
 * Prints all the shortest paths from all nodes to the destination
 * Each node has a 'predecessor' which points in the shortest direction to
the destination
 *
 * If we start from a node and look at its predecessor and then look at its
predecessor,
 * eventually we reach the destination.
 *
 * We print out this path for all nodes (except the destination itself)
 * */
void printPath(MinHeap_t* h, int destination);
```

```
/*
 * Implementation of Dijkstra's Algorithm
 * */
void Dijkstra(Graph_t g, int vertex);
```

Testing

Implemented a testing script for the program that looks at all txt files within the specified `--input` folder and pair it with the file with the file with the same name in the `--output` folder and verify that the program is running appropriately

```
$ ./compile.sh  # gcc *.c
$ ./test.sh     # python3 tests/t1.py --input inputs/ --output outputs/
Testing  inputs/1.txt outputs/1.txt
Success
Testing  inputs/2.txt outputs/2.txt
Success
Testing  inputs/3.txt outputs/3.txt
Success
Testing  inputs/4.txt outputs/4.txt
Success
Testing  inputs/5.txt outputs/5.txt
Success
Testing  inputs/6.txt outputs/6.txt
Success
Testing  inputs/7.txt outputs/7.txt
Success
```

Test 7

If there exist more than one path which all are minimum distance, one of them will be printed out. Which one is printed out depends on the input.

Source

Project code available at : https://github.com/AdityaNG/DAA_A2