

## Lab 13-2 (5/26) Dijkstra Algorithm

**Due : ~2022.06.07.(Tue) 23:59, Late Submission : ~2022.06.08.(Wed) 23:59**

- Implement the Dijkstra Algorithm to find the Shortest Path.
- 아래의 사진과 같이 [input].txt 파일을 입력받아 Dijkstra Algorithm을 수행하고 결과값을 [output].txt 파일에 출력하여 저장한다.
- Implement Dijkstra Algorithm that finds the shortest path by receiving the [input].txt file as input commands as below, and the output result is stored in the [output].txt file.

```

4
1-2-3 2-3-2 3-1-5
1->2 (cost : 3)
1->2->3 (cost : 5)
can not reach to node 4

```

- **<input>** :
  - 첫째줄: Graph에서 Node의 개수.
  - 둘째줄: Edge Set 정보. (공백을 사이에 두고 **SourceNode-TargetNode-Weight**의 형식, ex: “**2-3-2**” : **(Node:2)-----[weight:2]----->(Node:3)** 의 미)
  - 1st line: the number of Nodes in the graph.
  - 2nd line: information of the edge set. (**SourceNode-TargetNode-Weight** format with “one space”, ex: “**2-3-2**” : **(Node:2)-----[weight:2]----->(Node:3)**)
- **<output>** :
  - Source Node(key : 1)에서부터 나머지 모든 Node에 대한 Shortest Path 와 Distance (weight합) 출력.
  - Shortest Distance가 같은 Path가 여러개 존재하는 경우 하나의 Path만 출력.
  - All the shortest paths from the Source Node(key : 1) to other Nodes and the Distance (sum of weights) for each path should be printed.
  - If there exist more than one Shortest Paths which have the same distance, print only one path.

<Structure & Function Format>

## Structure

|                                                                                                            |                                                                                                              |
|------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <pre>typedef struct Node {<br/>    int vertex;<br/>    int dist;<br/>    int prev;<br/>}Node;</pre>        | <pre>typedef struct Graph {<br/>    int size;<br/>    int** vertices;<br/>    Node* nodes;<br/>}Graph;</pre> |
| <pre>typedef struct Heap {<br/>    int Capacity;<br/>    int Size;<br/>    Node* Element;<br/>}Heap;</pre> |                                                                                                              |

## Function

```
Graph* createGraph(int size);  
void deleteGraph(Graph* g);  
void printShortestPath(Graph* g);  
Heap* createMinHeap(int heapSize);  
void deleteMinHeap(Heap* minHeap);  
void insertToMinHeap(Heap* minHeap,  
                     int vertex, int distance);  
Node deleteMin(Heap* minHeap);
```

- 위 사진과 같은 Struct 구조체를 사용하셔야 합니다.
- 위 사진과 같은 함수들을 형식에 맞게 구현해주시면 됩니다.
- **Struct format above should be used for implementation**
- **Functions should be implemented in appropriate format as above.**

### <Execution>

- gcc 20XXXXXXXXXX.c -o 20XXXXXXXXXX
- ./20XXXXXXXXXX [input\_file\_name] [output\_file\_name]
- !!! 꼭 제공되는 testcase로 실행시켜보시기 바랍니다!!!!,
- !!! Run your solution code with the provided test case above and check whether it works properly !!!

### <Issue>

- 코드 작성시 주석을 적어주시기 바랍니다. 주석이 없는경우 Cheating으로 간주될 수 있습니다.
  - 제공된 testcase는 채점 case에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
  - 파일 입출력은 `argv[]`를 사용하여 구현해주시기 바랍니다.
  - 제출 마감 시간 이전의 가장 최신 버전의 commit을 기준으로 채점할 예정입니다.
  - 제출 파일과, 폴더 naming은 꼭 지정된 형식으로 해주세요 합니다.
  - Please write down the detailed comments when writing the code. If there is no comment, it might be considered cheating.
  - Provided test case is included in the test case for grading. Please check to see if it makes a proper result.
  - Do not use a fixed file name when inputting and outputting files, but implement it using `argv[]` as in skeleton code.
  - Scoring will be based on the latest version of commit before the deadline.
  - The names of the .c file and directory should be named in proper format.
- 
- 출력시 꼭 정해진 형식에 맞게 출력해주셔야 합니다. 모든 공백은 띄어쓰기 한칸입니다.

- Graph에서 Source Node는 Key 값이 1인 Node입니다. Node의 개수가 5개라면, 각 Node의 Key 값은 1, 2, 3, 4, 5입니다.
  - 결과 출력은 Source Node(key=1)를 제외한 모든 node에 대해서 Path와 Distance를 모두 출력하시면 됩니다. ex) “source->...PATH...->target (cost : cost값)”
  - 만약 Source Node에서 갈 수 있는 Path가 없을 경우 위 사진처럼 “can not reach to node keyValue”를 출력하시면 됩니다.
  - Distance가 같은 Shortest Path가 여러개 존재하는 경우, 아무 경로나 하나만 출력하시면 됩니다.
  - All the messages must be printed out according to the appropriate format as shown in the example above. All the spaces are “one space”.
  - The Source Node of the graph is the node which has Key value “1”. If the number of nodes in the graph is 5, each Node's Key value is 1, 2, 3, 4, 5.
  - For printing the Path result, paths and its distance to all nodes except the Source Node(key=1) should be printed. ex) “source->...PATH...->target (cost : costValue)”
  - If the path does not exist, “can not reach to node keyValue” should be printed.
  - If there are multiple Shortest Paths, any of them can be printed.

## <Directory Format>

- 아래와 같이 git 프로젝트 폴더에 "lab13-2" 폴더 생성후, "lab13-2" 폴더 안에 "20XXXXXXXXXX.c" 파일을 위치시키시면 됩니다.
  - After creating the "lab13-2" directory in the git-project-directory as below, place the "20XXXXXXXXXX.c" file in the "lab13-2" directory.

---...  
--lab11/  
-----2020XXXXXX.c  
---...  
--lab12/  
-----2020XXXXXX.c  
---...  
--lab13-1/  
-----2020XXXXXX.c  
---...  
--lab13-2/  
-----2020XXXXXX.c