광운대학교

데이터구조설계

Project 3

제출일자: 2017년 12월 8일 (금)

학　　과: 컴퓨터공학과

담당교수: 이기훈 교수님

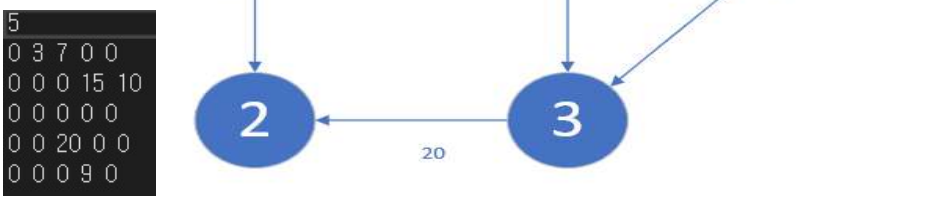
학　　번: 2014722075

성　　명: 이동준

O Introduction

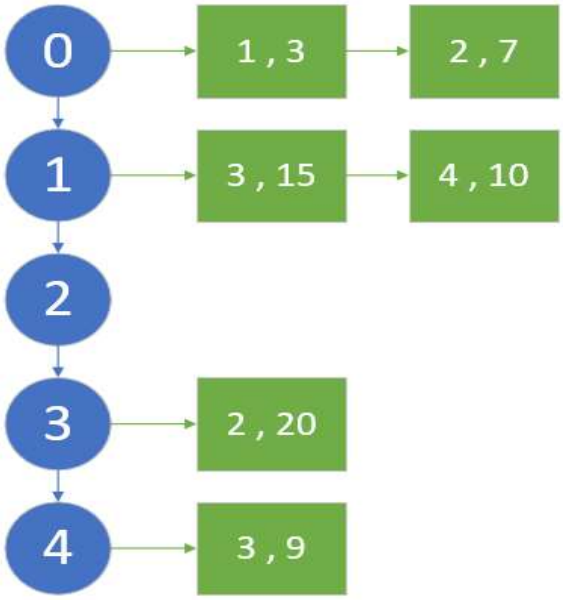
위 프로젝트는 그래프를 이용하여 미로 찾기 프로그램을 구현한 것이다. 이 프로그램은 미로에 대한 Data 가 저장된 mapdata 텍스트 파일을 통해서 그래프를 구성하며, Vertex 의 Key 에 대한 오름차순으로 구현되어 있으며 각 Vertex 와 연결된 Edge 또한 Key 에 대한 오름차순으로 구현되어 있다. 이 그래프를 기반으로 하여 DFS, Dijkstra, Bellman-Ford 의 알고리즘을 이용하여 경로와 거리를 구하는 프로그램이다. command 에 따라 동작을 실행하며, 실행 결과는 result text(result.log)파일에 저장하고, 그에 대한 error 는 error text(error.log)파일에 나누어 저장한다.

그래프는 그래프의 크기와 각 Vertex 간의 방향성과 Weight 를 나타낸다.



-> 맵 데이터, 그래프의 예시 (그래프 크기 : 5)

위 그래프를 기반으로 하여 2D linked list 를 이용한 그래프를 구현한다.



* command

- LOAD

mapdata 의 정보를 기반으로 하여 그래프를 형성한다. 위의 그림처럼 2D linked list 의 형태로 그래프의 정보를 관리한다. 만약 LOAD 할 텍스트 파일이 존재하지 않을 경우, 에러(LoadFileNotExist)를 출력하고, 오류코드(101)을 출력한다.

- PRINT

mapdata 를 읽어 미로 맵을 출력하는 명령어. Matrix 형태로 출력하며, LOAD 된 데이터가 존재하지 않을 경우, 에러(GraphNotExist)를 출력하고, 오류코드(202)를 출력한다.

- DFS

mapdata 를 기반으로 하여 시작점과 끝점의 경로를 찾는 알고리즘. Depth First Search,깊이 우선 탐색이라 하며, 스택을 이용하여 구현하였다. 순환하지 않는 경로를 찾아 최종 경로와 시작점과 끝점의 Distance 를 반환하여 출력한다. 마찬가지로 LOAD 가 된 데이터가 없을 경우 에러를 출력하며, 명령어 인자가 부족한 경우에는 VertexKeyNotExist, 인자로 입력한 Vertex 가 존재하지 않는 경우에는 InvalidVertexKey, 맵 데이터 중 음수의 Weight 가 존재하는 경우에는 InvalidAlgorithm 을 출력하며, 에러없이 정상적인 구동을 완료하면 에러코드는 0 이다.

- DIJKSTRA

mapdata 를 기반으로 하여 시작점과 끝점의 최단경로와 거리를 찾는 알고리즘. STL set 을 이용하여 구현하였다. set 을 이용하여 최단 Distance 를 구분하여 최단거리와 경로를 찾아내고, 시작점과 끝점의 Distance 를 반환하여 출력한다. 마찬가지로 LOAD 가 된 데이터가 없을 경우 에러를 출력하며, 명령어 인자가 부족한 경우에는 VertexKeyNotExist, 인자로 입력한 Vertex 가 존재하지 않는 경우에는 InvalidVertexKey, 맵 데이터 중 음수의 Weight 가 존재하는 경우에는 InvalidAlgorithm 을 출력하며, 에러없이 정상적인 구동을 완료하면 에러코드는 0 이다.

- DIJKSTRAM – MinHeap 을 이용하였다. 나머지는 위의 알고리즘과 동일

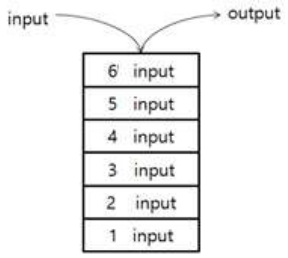
- BELLMANFORD

mapdata 를 기반으로 하여 시작점과 끝점의 최단경로와 거리를 찾는 알고리즘. 각 Vertex 를 방문하여 Distance 를 최신화하고 최종적으로 최단경로와 거리를 찾는다. 음수 Weight 가 있어도 동작이 가능하지만 음수 사이클(Negative Cycle)이 발생하는 경우에는 확인하여 에러를 출력한다. 마찬가지로 LOAD 가 된 데이터가 없을 경우 에러를 출력하며, 명령어 인자가 부족한 경우에는 VertexKeyNotExist, 인자로 입력한 Vertex 가 존재하지 않는 경우에는 InvalidVertexKey 를 출력하며 음수 사이클이 존재할 경우에는 NegativeCycleDetected 을 출력한다. 에러없이 정상적인 구동을 완료하면 에러코드는 0 이다.

- else

만약 command text 파일이 존재하지 않을 경우에는 CommandFileNotExist 을 출력하고, 존재하지 않는 명령어가 입력된 경우에는 NonDefinedCommand 을 출력한다.

* Stack



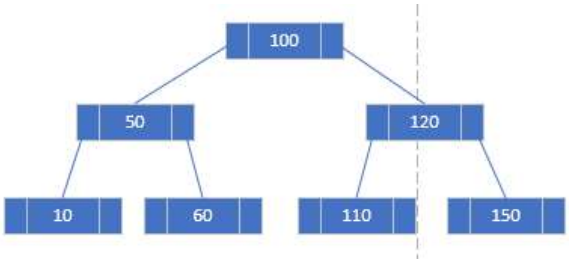
First in Last out – 먼저 입력된 데이터가 가장 밑으로 저장되고 가장 마지막에 입력된 데이터부터 출력을 한다. DFC 에서는 하나의 Vertex 의 Edge 를 오름차순으로 Stack 에 insert 하고 가장 마지막에 insert 된 즉, Vertex 의 Key 가 가장 큰 Vertex 를 Visit 한다.

* Set

단순 컨테이너로 Key 라 불리는 원소의 집합으로 이루어진 컨테이너

노드 기반 컨테이너이며 균형 이진 트리로 구현되었다.

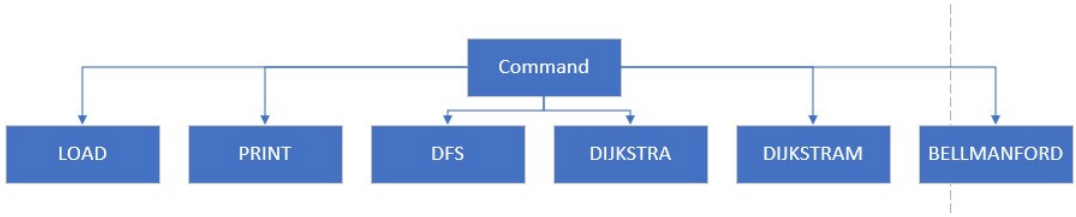
중복된 값을 가지지 못한다.

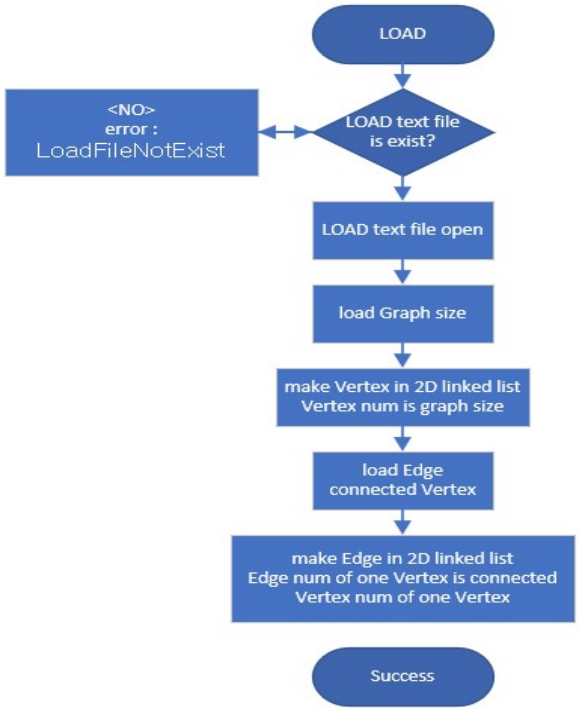


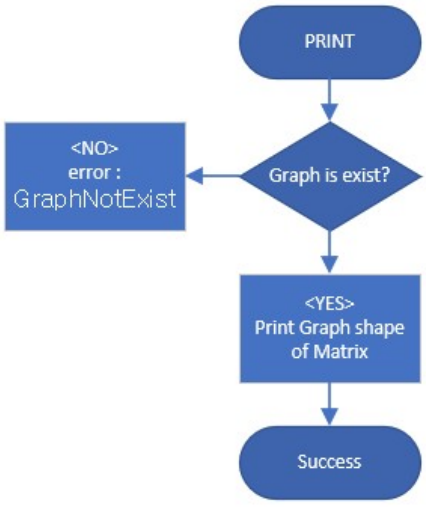
* MinHeap

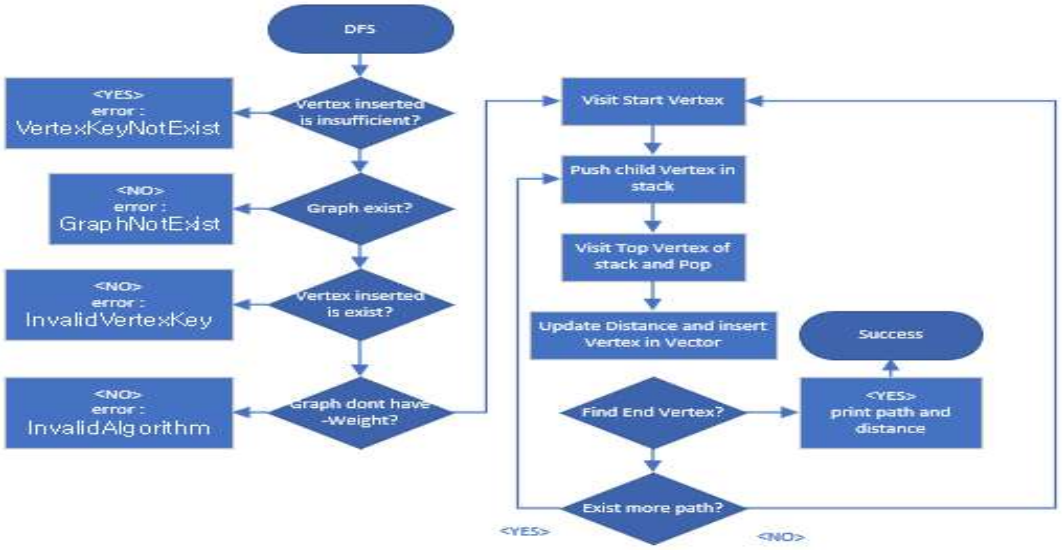
가장 작은 Key 값이 root 노드가 되는 트리형태의 배열이다.

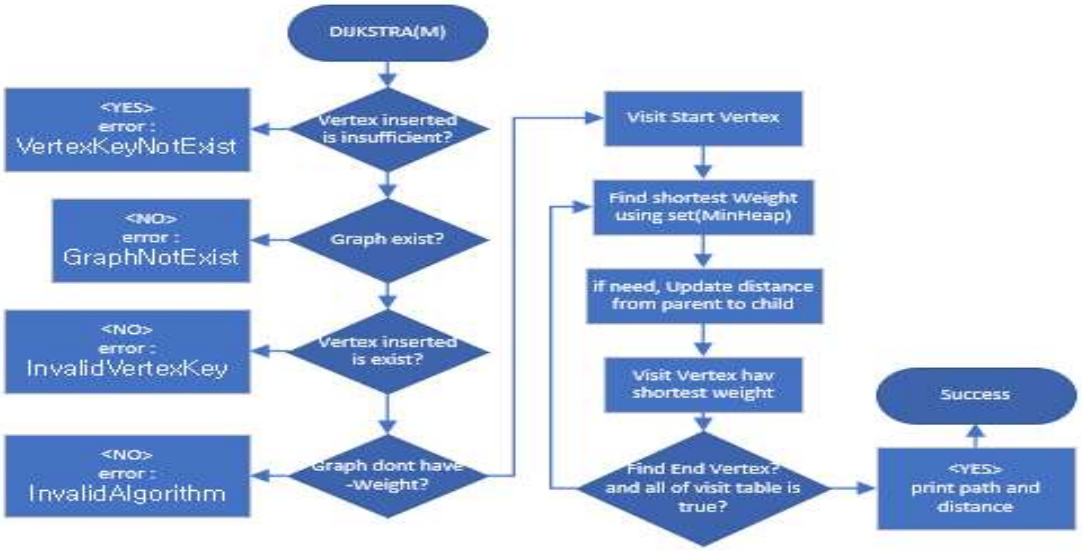
O Flowchart

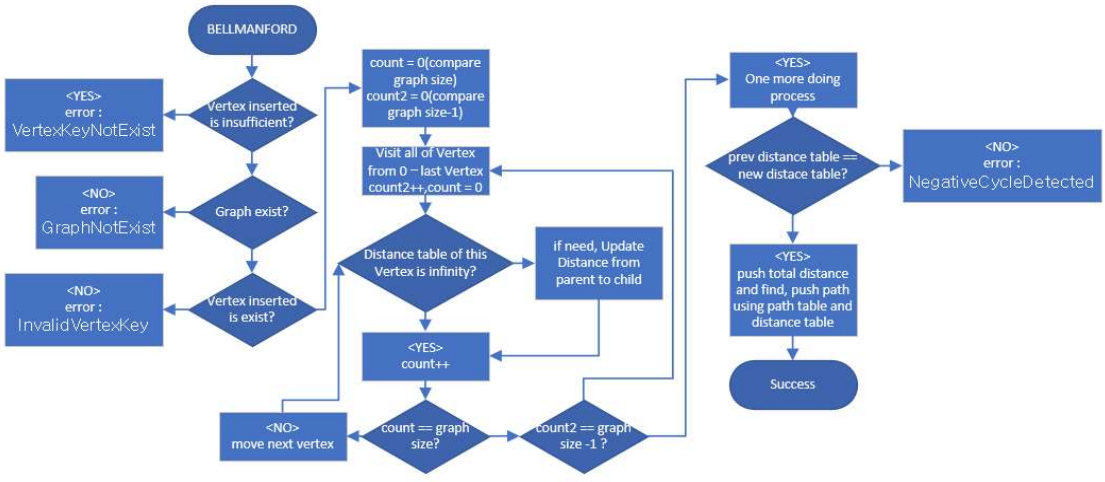







DFS
Vertex inserted is insufficient?
<YES> error : VertexKeyNotExist
Graph exist?
<NO> error : GraphNotExist
Vertex inserted is exist?
<NO> error : InvalidVertexKey
Graph dont have -Weight?
<NO> error : InvalidAlgorithm
Visit Start Vertex
Push child Vertex in stack
Visit Top Vertex of stack and Pop
Update Distance and insert Vertex in Vector
Find End Vertex?
<YES> print path and distance
Success
Exist more path?
<YES>
<NO>


DIJKSTRA(M)
Vertex inserted is insufficient?
<YES> error : VertexKeyNotExist
Graph exist?
<NO> error : GraphNotExist
Vertex inserted is exist?
<NO> error : InvalidVertexKey
Graph dont have -Weight?
<NO> error : InvalidAlgorithm
Visit Start Vertex
Find shortest Weight using set(MinHeap)
if need, Update distance from parent to child
Visit Vertex hav shortest weight
Find End Vertex? and all of visit table is true?
<YES> print path and distance
Success


BELLMANFORD
Vertex inserted is insufficient?
<YES> error : VertexKeyNotExist
Graph exist?
<NO> error : GraphNotExist
Vertex inserted is exist?
<NO> error : InvalidVertexKey
count = 0(compare graph size) count2 = 0(compare graph size-1)
Visit all of Vertex from 0 ~ last Vertex count2++,count = 0
Distance table of this Vertex is infinity?
if need, Update Distance from parent to child
<YES> count++
count == graph size?
<NO> move next vertex
count2 == graph size -1 ?
<YES> One more doing process
prev distance table == new distace table?
<NO> error : NegativeCycleDetected
<YES> push total distance and find, push path using path table and distance table
Success

O Algorithm

* Run

fileopen(command file)

if(not exist file)

	error : CommandFileNotExist

while(get one line from file)

	find command

	if(LOAD)

		if(load text file is not exist)

			error : LoadFileNotExist

		LOAD(textfile) //make 2D linked list

		error : Success

	else if(PRINT)

		if(graph is not exist)

			error : GraphNotExist

		Print()

		error : Success

	else if(DFS)

		find start key, end key

		if(inserted vertex is insufficient)

			error : VertexKeyNotExist

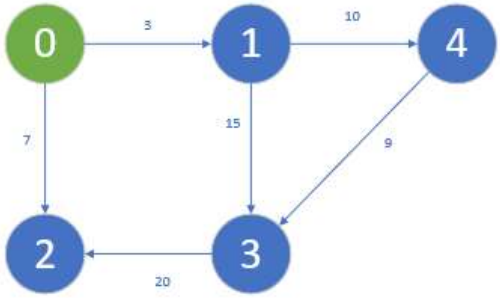
		FindPathDfs(start,end)

		print error

```
else if(DIJKSTRA)
        find start key, end key
        if(inserted vertex is insufficient)
                error : VertexKeyNotExist
        FindShortestPathDijkstraUsingSet(start,end)
        print error
else if(DIJKSTRAM)
        find start key, end key
        if(inserted vertex is insufficient)
                error : VertexKeyNotExist
        FindShortestPathDijkstraUsingMinHeap(start,end)
        print error
else if (BELLMANFORD)
        find start key, end key
        if(inserted vertex is insufficient)
                error : VertexKeyNotExist
        FindShortesPathBellmanFord(start,end)
        print error
else
        error : NonDefinedCommand
```

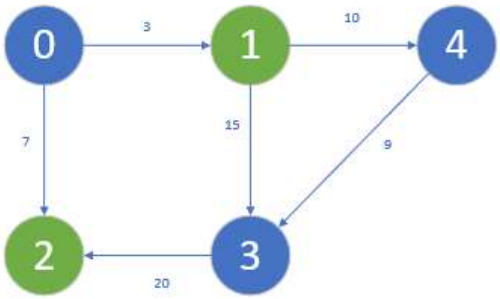
* FindPathDfs(startVertexKey,endVertexKey)

DFS 동작 예시) DFS 0 3



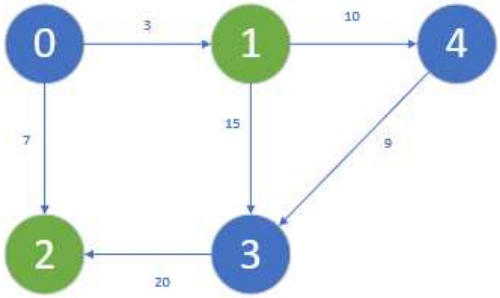
Path :

0	1	2	3	4
F	F	F	F	F



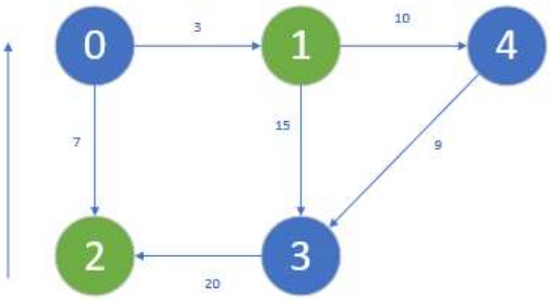
Path : 0

0	1	2	3	4
T	F	F	F	F



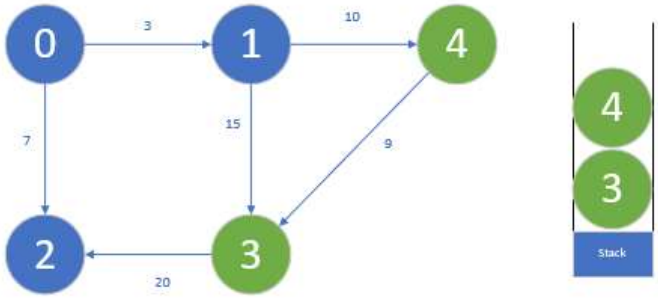
Path : 0->2(fail)

0	1	2	3	4
T	F	T	F	F



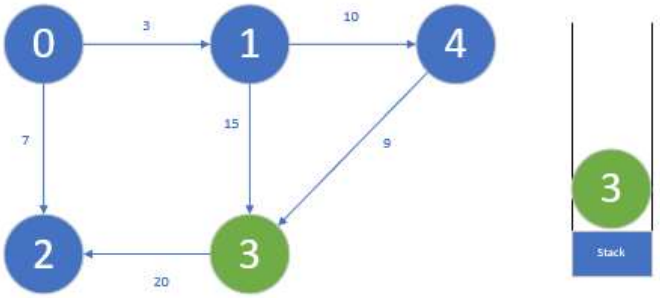
Path : 0

0	1	2	3	4
T	T	T	F	F



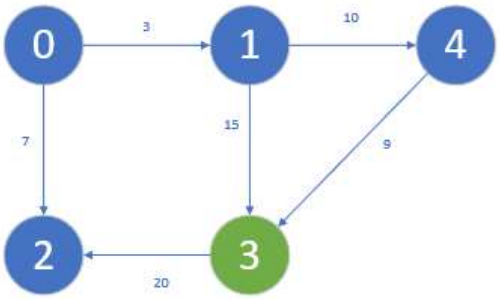
Path : 0->1

0	1	2	3	4
T	T	T	F	F



Path : 0->1->4

0	1	2	3	4
T	T	T	F	T



Path : 0->1->4->3, Cost : 22

0	1	2	3	4
T	T	T	T	T

if(Graph is not exist)

error : GraphNotExist

if(Vertex is not exist(start or end)

error : InvalidVertexKey

if(Graph have – Weight)

error : InvalidAlgorithm

p = findvertex(startVertexKey)

insert p in vector

while(v.back() != endVertexKey)

while(visit all of edge of p)

if(visit table of edge is false)

push in stack

else

if(edge have more path)

push in stack

p = Vertex of Top Key in stack

insert p in vector

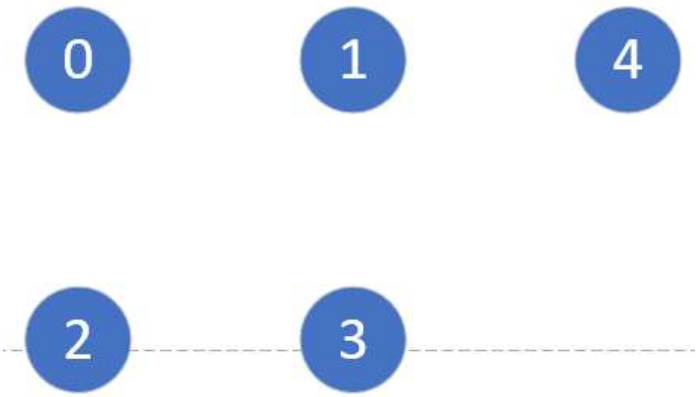
if(find endVertex)

return vector(include path and total distance)

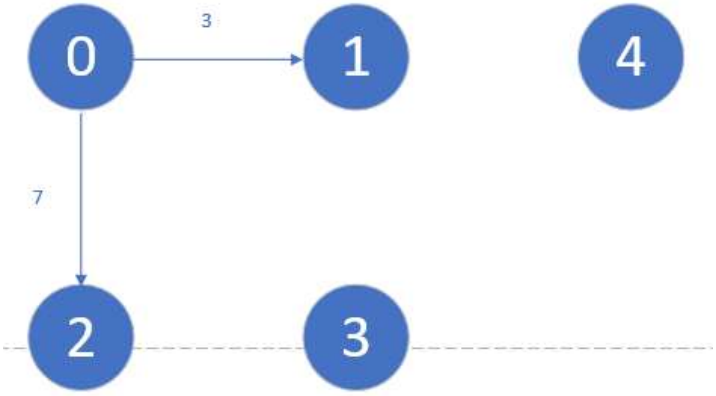
* FindShortestPathDijkstraUsingSet(startVertexKey,endVertexKey)

- using minheap is same process

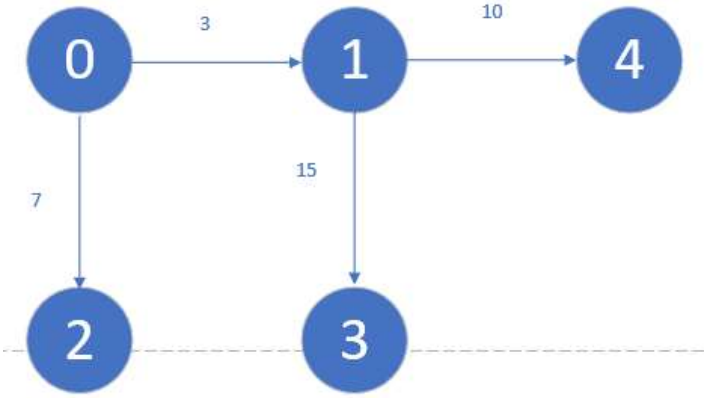
DIJKSTRA 동작 예시) DIJKSTRA 0 3



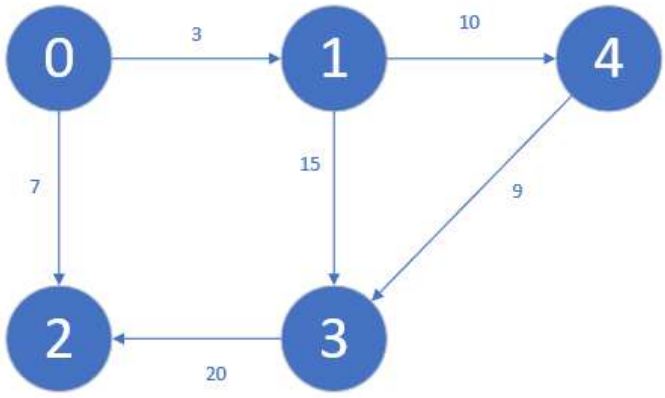
Vertex	0	1	2	3	4
Distance	0	∞	∞	∞	∞
Visit	0	0	0	0	0
path	-1	-1	-1	-1	-1



Vertex	0	1	2	3	4
Distance	0	3	7	∞	∞
Visit	1	0	0	0	0
path	-1	0	0	-1	-1



Vertex	0	1	2	3	4
Distance	0	3	7	(3+15)	(3+10)
Visit	1	1	0	0	0
path	-1	0	0	1	1



Vertex	0	1	2	3	4
Distance	0	3	7	(3+15)	(3+10)
Visit	1	1	1	0	0
path	-1	0	0	1	1

모든 Vertex 방문 후 0-3 최단거리,경로 찾기

if(Graph is not exist)

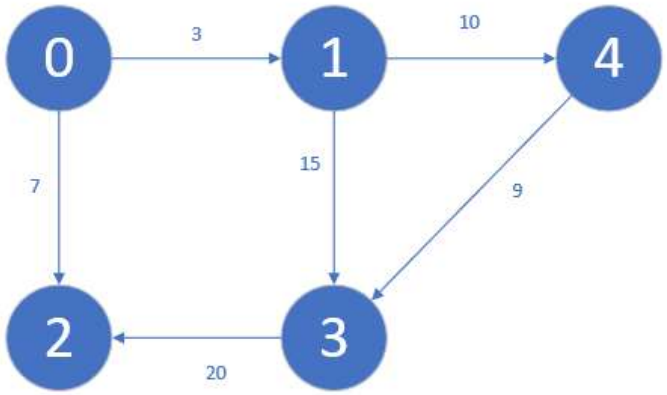
error : GraphNotExist

if(Vertex is not exist(start or end)

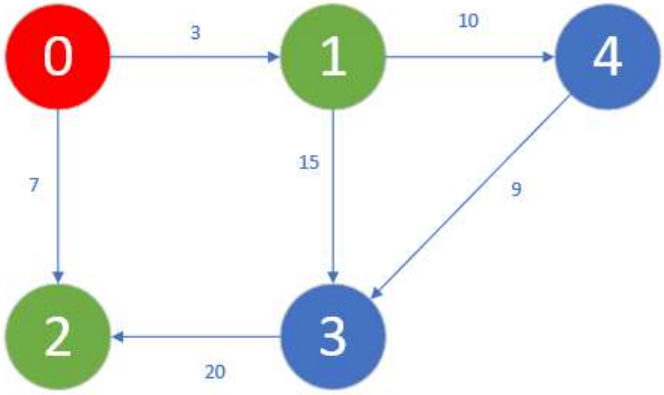
```
        error : InvalidVertexKey
if(Graph have – Weight)
        error : InvalidAlgorithm
p = findvertex(startVertexKey)
while(1)
        while(visit all of edge of p)
                if( need update distance of edge)
                        update distance table
                                if(already exist distance in set)
                                        erase distance of updated distance
                                        push new distance
        find shortest path using set
        p = Vertex found shortest path
        if(all of visit table is true or set size ==0 and visited endvertex
make vector
push total distance in vector
push path using path table
return vector(include path and total distance)
```

* FindShortesPathBellmanFord(startVertexKey,endVertexKey)

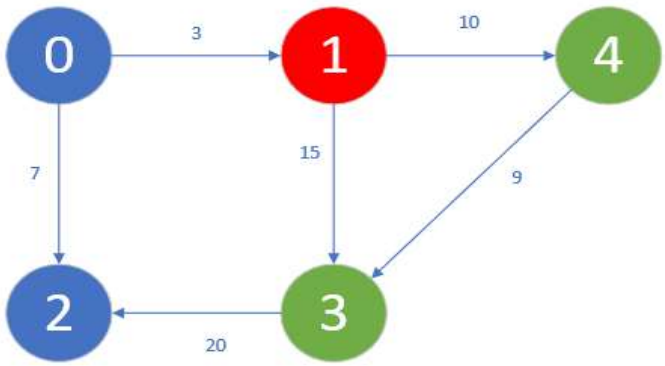
BELLMANFORD 동작 예시) BELLMANFORD 0 3



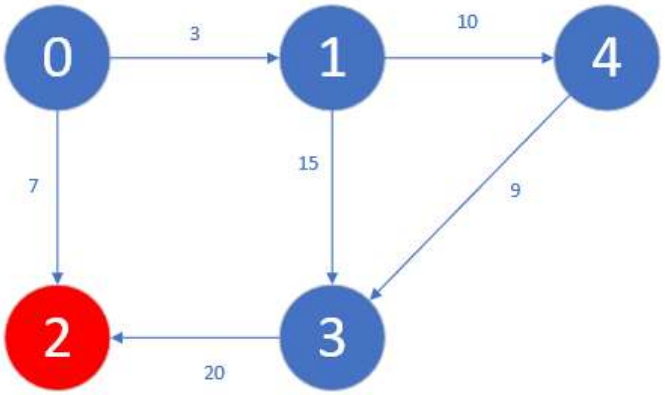
Vertex	0	1	2	3	4
Distance	0	∞	∞	∞	∞
path	-1	-1	-1	-1	-1



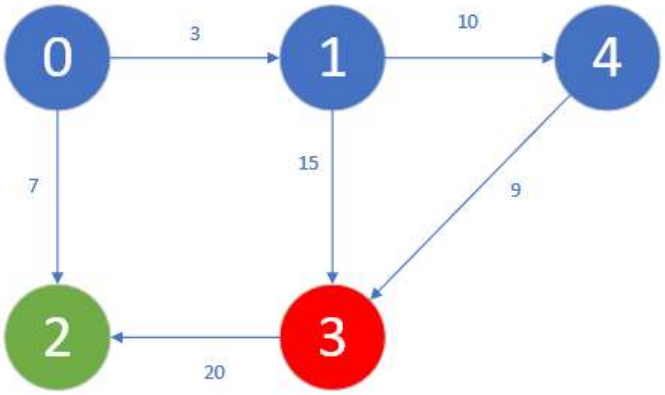
Vertex	0	1	2	3	4
Distance	0	3	7	∞	∞
path	-1	0	0	-1	-1



Vertex	0	1	2	3	4
Distance	0	3	7	18	13
path	-1	0	0	1	1

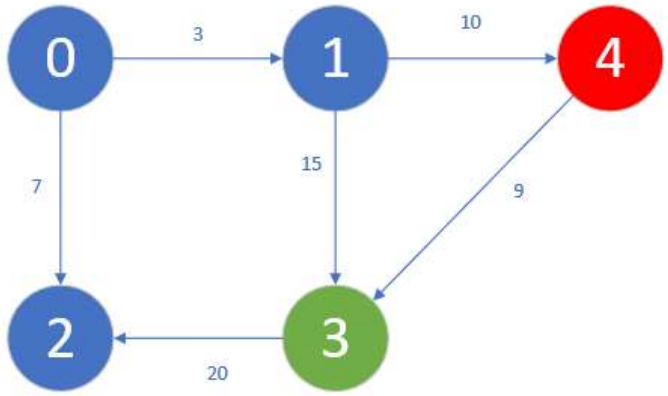


Vertex	0	1	2	3	4
Distance	0	3	7	18	13
path	-1	0	0	1	1



Vertex	0	1	2	3	4
Distance	0	3	7	18	13

path	-1	0	0	1	1



Vertex	0	1	2	3	4
Distance	0	3	7	18	13
path	-1	0	0	1	1

if(Graph is not exist)

error : GraphNotExist

if(Vertex is not exist(start or end)

error : InvalidVertexKey

if(Graph have – Weight)

error : InvalidAlgorithm

p = Head vertex

count = 0, count2 = 0, dis(arr), dis2(arr)

while(count != graph size-1)

while(count2 != graph size)

if(dis of p != infinity)

while(visit all of edge of p)

if(need update distance of edge)

update dis,dis2

count2++

p = next vertex

count2 = 0

count++

p = Head vertex

doing again only use dis

if(dis != dis2)

return empty vector //error

else

push total distance

push path

return vector(include path and total distance)

O Result Screen

* LOAD

```
====== LOAD ======
LoadFileNotExist
==================
====== LOAD ======
Success             Error code: 101
==================  Error code: 0
```

load text file 이 존재하지 않는 경우 에러를 출력하며 LOAD 가 정상적으로 이루어 지면 Success 를 출력한다.

* PRINT

```
====== PRINT ======
0 70 90 0 10 0 0
0 0 10 0 0 0 8
0 0 0 0 7 0 0
0 0 0 0 16 3 0
0 0 0 0 0 0 5
0 12 0 1 0 0 9
0 13 0 0 0 7 0
==================== Error code: 0
```

PRINT 명령어가 실행되면 LOAD 가 완료된 맵 데이터를 출력한다.

* DFS

```
====== DFS ======
0 4 6 5 3 23
==================
====== DFS ======
VertexKeyNotExist
==================
====== DFS ======
0 4 6 5 22
==================  Error code: 0
====== DFS ======   Error code: 200
InvalidVertexKey    Error code: 0
==================  Error code: 201
```

DFS 를 통해 시작점과 끝점의 경로를 찾아내고 올바르지 않은 명령어 실행의 경우 에러출력.

* DIJKSTRA

```
====== DIJKSTRA ======
0 4 6 5 3 23
==================
====== DIJKSTRA ======
VertexKeyNotExist
==================
====== DIJKSTRAM ======
0 4 6 5 22
==================          Error code: 0
====== DIJKSTRAM ======     Error code: 200
InvalidVertexKey            Error code: 0
==================          Error code: 201
```

DIJKSTRA 를 통해 시작점과 끝점의 최단경로와 최단거리를 찾아내고 올바르지 않은 명령어 실행의 경우 에러출력.

* BELLMANFORD

```
====== BELLMANFORD ======
0 4 6 5 3 23
==================
====== BELLMANFORD ======
VertexKeyNotExist
==================
====== BELLMANFORD ======
0 4 6 5 22
==================          Error code: 0
====== BELLMANFORD ======   Error code: 200
InvalidVertexKey            Error code: 0
==================          Error code: 201
```

BELLMANFORD 를 통해 시작점과 끝점의 최단경로와 최단거리를 찾아내고 올바르지 않은 명령어 실행의 경우 에러출력

```
====== BELLMANFORD ======
NegativeCycleDetected
==================          Error code: 204
```

더불어 음수의 Weight 가 존재하여도 실행 가능하지만 음수의 사이클이 존재하는 경우에는 위와 같은 에러를 출력한다.

* else

```
====== ASTAR ======
NonDefinedCommand
==================== Error code: 300
```

잘못된 명령어 입력 시 다음과 같은 에러를 출력한다.

O Consideration

이번 프로젝트를 진행하면서 2 차 프로젝트보다 어려움을 겪었다. 그래프에 대한 개념이 부족한 상태였기 때문에 기초부터 다시 잡아가야만 했다. 더불어 타전공과목의 프로젝트 설계도 있었기 때문에 시간활용이 쉽지 않았다. 하지만 빠르게 개념을 이해하고 코드로 완성을 하였고, 실질적으로 오랜 기간이 걸리지 않았다. 이번 프로젝트에서 3 개의 알고리즘을 사용하여 구현을 하였다. 그 과정에서 예외처리의 부분에서 어려움을 겪었는데, 다양한 케이스를 다루기 때문에 많은 케이스를 모두 구동 가능하도록 하고 싶었다. 다양한 케이스를 실험해보고 특수 케이스까지 실험을 해보았고, 그 결과 프로젝트를 정상적으로 구현하였다. 특히 DFS 에서는 타 알고리즘보다 예외처리를 하는 과정에서 어려움이 컸고, Minheap 을 구현하는데 있어서 지난 Maxheap 을 구현한 이력이 있었기에 그것을 바탕으로 구현하였다. DIJKSTRA 를 구현하며서 STL set 에 대하여 알게 되었고 생각보다 유용한 STL 이였다. B++ Tree 와 BST 를 구현했기 때문에 활용 가치도 쉽게 알 수 있었다. 학기를 마무리하면서 마지막 3 차 프로젝트도 성공적으로 완성할 수 있었는데, 결과도 긍정적으로 마무리 되었으면 좋겠다.