

자료구조 #4.

32181854 박준영

1. 실행결과

```
Microsoft Visual Studio 디버깅 콘솔
5번째 간선과 가중치 입력> 1 1 0
6번째 간선과 가중치 입력> 1 2 10
7번째 간선과 가중치 입력> 1 3 15
8번째 간선과 가중치 입력> 2 2 0
9번째 간선과 가중치 입력> 2 4 30
10번째 간선과 가중치 입력> 3 0 20
11번째 간선과 가중치 입력> 3 3 0
12번째 간선과 가중치 입력> 3 4 15
13번째 간선과 가중치 입력> 4 1 20
14번째 간선과 가중치 입력> 4 2 35
15번째 간선과 가중치 입력> 4 4 0
16번째 간선과 가중치 입력> 5 4 3
17번째 간선과 가중치 입력> 5 5 0
시작 정점 입력> 0
<dist>
0 45 45 10 25 50000
<shortest path>
0 -> 0: 0
0 -> 1: 0 3 4 1
0 -> 2: 0 2
0 -> 3: 0 3
0 -> 4: 0 3 4
0 -> 5: 0 5
C:\Users\user\Documents\GitHub\Graph\weight_graph\Debug\weight_graph.exe(프로세스 19188개)이(가) 종료되었습니다(코드: 0
개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

```
Microsoft Visual Studio 디버깅 콘솔
정점 수 와 간선 수 입력> 5 8
1번째 간선과 가중치 입력> 0 2 6
2번째 간선과 가중치 입력> 0 3 3
3번째 간선과 가중치 입력> 1 0 3
4번째 간선과 가중치 입력> 2 3 2
5번째 간선과 가중치 입력> 3 2 1
6번째 간선과 가중치 입력> 3 1 1
7번째 간선과 가중치 입력> 4 1 4
8번째 간선과 가중치 입력> 4 3 2
시작 정점 입력> 4
<dist>
6 3 3 2 50000
<shortest path>
4 -> 0: 4 3 1 0
4 -> 1: 4 3 1
4 -> 2: 4 3 2
4 -> 3: 4 3
4 -> 4: 4
C:\Users\user\Documents\GitHub\Graph\weight_graph\Debug\weight_graph.exe(프로세스 37284개)이(가) 종료되었습니다(코드: 0
개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

```
Microsoft Visual Studio 디버깅 콘솔
정점 수 와 간선 수 입력> 6 9
1번째 간선과 가중치 입력> 3 0 10
2번째 간선과 가중치 입력> 3 1 30
3번째 간선과 가중치 입력> 3 2 15
4번째 간선과 가중치 입력> 0 4 20
5번째 간선과 가중치 입력> 1 5 5
6번째 간선과 가중치 입력> 2 1 5
7번째 간선과 가중치 입력> 2 5 20
8번째 간선과 가중치 입력> 5 2 20
9번째 간선과 가중치 입력> 4 5 20
시작 정점 입력> 3
<dist>
10 20 15 50000 30 25
<shortest path>
3 -> 0: 3 0
3 -> 1: 3 2 1
3 -> 2: 3 2
3 -> 3: 3
3 -> 4: 3 0 4
3 -> 5: 3 2 1 5
C:\Users\user\Documents\GitHub\Graph\weight_graph\Debug\weight_graph.exe(프로세스 27552개)이(가) 종료되었습니다(코드: 0
개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

2. 코드

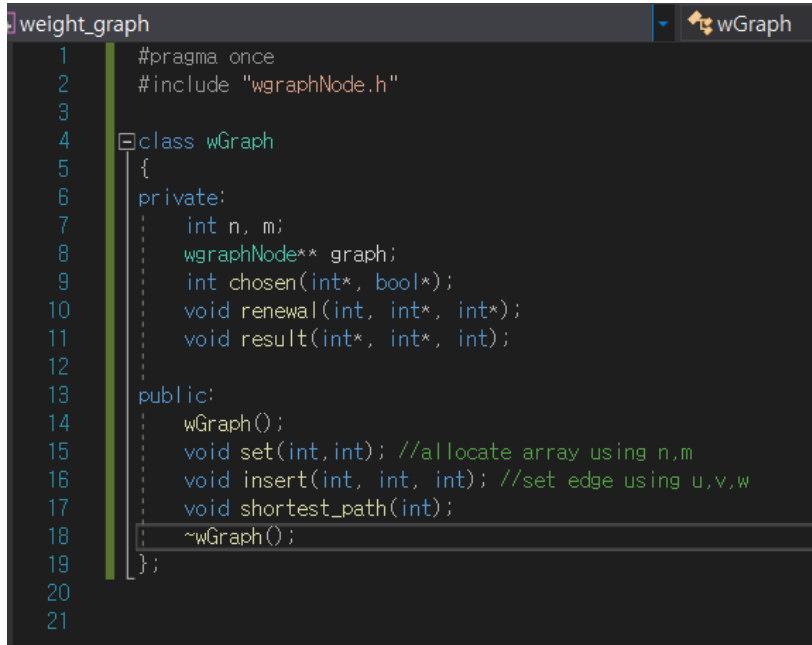
〈wGraphNode〉

-wGraphNode.h

```
wGraph.cpp  wGraphNode.h  main.cpp
weight_graph
1  #pragma once
2  class wGraph;
3  class wGraphNode
4  {
5  private:
6      int weight = 0;
7      friend wGraph;
8  public:
9      wGraphNode() { }
10 };
11
12
```

<wGraph>

-wGraph.h



```
1 #pragma once
2 #include "wGraphNode.h"
3
4 class wGraph
5 {
6 private:
7     int n, m;
8     wGraphNode** graph;
9     int chosen(int*, bool*);
10    void renewal(int, int*, int*);
11    void result(int*, int*, int);
12
13 public:
14    wGraph();
15    void set(int, int); //allocate array using n,m
16    void insert(int, int, int); //set edge using u,v,w
17    void shortest_path(int);
18    ~wGraph();
19 };
20
21
```

- n: 정점의 수
- m: 간선의 수
- set(): 정점과 간선 개수를 매개변수로 받아 2차원배열을 할당하고 멤버변수 n,m을 셋팅
- insert(): edge들의 시작정점과 끝정점, 가중치를 매개변수로 받아 간선을 삽입
- shortest_path(): 시작정점을 매개변수로 받아 최단거리를 구함
- chosen(): 이미 경로에 포함되지 않은 정점들 중 가장 가중치가 가장 작은 정점을 return
- renewal(): dist를 갱신
- result(): 결과를 출력
- chosen, renewal, result는 shortest_path 내부에서 사용됨

-wGraph.cpp

```
wGraph.cpp  x wGraphNode.h  main.cpp  wGraph.h
weight_graph  wGraph

1  #include "wGraph.h"
2  #include <stack>
3  #include <iostream>
4  #define _MAX_INT 50000
5  using namespace std;
6
7  wGraph::wGraph() { n = 0; m = 0; graph = 0; }
8  void wGraph::set(int n, int m) {
9      int i, j;
10     this->n = n;
11     this->m = m;
12     graph = new wGraphNode*[n];
13     for (i = 0; i < n; i++)
14         for (j = 0; j < n; j++)
15             graph[i][j] = new wGraphNode[n];
16
17     for (i = 0; i < n; i++) //initailization
18         for (j = 0; j < n; j++)
19             graph[i][j].weight = _MAX_INT;
20 }
21 void wGraph::insert(int u, int v, int w) {
22     graph[u][v].weight = w;
23 }
24 void wGraph::shortest_path(int s) {
25     //variable definition
26     int* path = new int[n];
27     for (int i = 0; i < n; i++)
28         path[i] = s;
29     bool* check = new bool[n] {false}; //checking already chosen vertex
30     int* dist = new int[n];
31     for (int i = 0; i < n; i++)
32         dist[i] = graph[s][i].weight;
33     check[s] = true;
34
35     //finding shortest path
36     //int u = 0; //s, p = 0;
37     for (int i = 0; i < n - 2; i++) {
38         //p = u;
39         int u = chosen(dist, check);
40         check[u] = true;
41         renewal(u, dist, path);
42     }
43
44     //print result;
45     result(dist, path, s);
46
47     delete[] path;
48     delete[] dist;
49     delete[] check;
50 }
51 int wGraph::chosen(int* a, bool* ch) {
52     int min = _MAX_INT;
53     int j = 0;
54     for (int i = 0; i < n; i++) {
55         if (min > a[i] && !ch[i]) {
56             min = a[i];
```

```

57         j = i;
58     }
59 }
60 return j;
61 }
62 void wGraph::renewal(int u,int* dist,int* path) {
63     wgraphNode** length = graph;
64     for (int i = 0; i < n; i++) {
65         if (dist[i] > dist[u] + length[u][i].weight) {
66             dist[i] = dist[u] + length[u][i].weight;
67             path[i] = u;
68         }
69     }
70 }
71 }
72 void wGraph::result(int* dist, int* path, int start) {
73     stack<int> stack;
74
75     cout << "<dist>" << endl;
76     for (int i = 0; i < n; i++) {
77         cout << dist[i] << ' ';
78     }
79
80     cout << endl;
81
82     cout << "<shortest path>" << endl;
83     for (int i = 0; i < n; i++) {

```

```

84         int j = i; //j is destination
85         while(j != start /*&& (j != _MAX_INT)*/){
86             stack.push(j);
87             //stack.push(path[j]);
88             j = path[j];
89         }
90         stack.push(start);
91         cout << start << " -> " << j << ": ";
92         while (!stack.empty()) {
93             cout << stack.top() << ' ';
94             stack.pop();
95         }
96         cout << endl;
97     }
98 }
99 }
100 wGraph::~wGraph() { delete[] graph; }
101

```

<main>

```
wGraph.cpp  wGraphNode.h  main.cpp  wGraph.h
weight_graph  (전역 범위)
1  #include <iostream>
2  #include "wGraph.h"
3  using namespace std;
4
5  int main() {
6      int n, m; //the number of vertices and edges
7      int u, v, weight;
8      int start;
9      wGraph g;
10
11      //input
12      cout << "정점 수 와 간선 수 입력> ";
13      cin >> n >> m;
14      g.set(n, m);
15
16      for (int i = 0; i < m; i++) {
17          cout << i + 1 << "번째 간선과 가중치 입력> ";
18          cin >> u >> v >> weight;
19
20          if (weight < 0) {
21              cout << "warning: weight must be positive integer" << endl;
22              i--; //re-input weight
23          }
24          else g.insert(u, v, weight);
25      }
26
27      cout << "시작 정점 입력> ";
28      cin >> start;
29
30
31      //output result
32      g.shortest_path(start);
33
34  }
```