Practice #12

Graphs (Dijkstra's Algorithm)

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

Practice #12 TO-DO List

To-Do	Submission	Notes
Digraph Class	Screenshot and source code (All files including Digraph.cpp)	p.70~73, Chapter 6

- Upload your screenshot and source codes on LMS by 11pm on 5/19 (Wed).
  - All your screenshots should be merged in one pdf file, screenshot.pdf.
  - Your pdf and all source codes should be compressed into zip file.
- File name: practice12_Your Student ID_Name.zip (only zip, not pdf, docx, c, etc)
  - ex) practice12_20400022_고윤민.zip

Dijkstra's Algorithm

- Implement a Digraph class and Dijkstra's algorithm
  - Complete a Digraph.cpp (DigraphMain.cpp: no need to change)
    - Digraph.cpp defines a Digraph class and its member functions
  - Refer to p.70~73, Chapter 6
  - We use adjacency matrix for representation of directed graph.
  - Implement following member functions. You can modify the given source codes.
    - ShortestPath(int *v*): Dijkstra's Algorithm (p.73)
    - Choose(): find a vertex with minimum distance (use *int distance[]* in class)
    - PrintPath(int *src*, int *dest*): print a path from vertex *src* to vertex *dest*
      - We can implement this function in recursive method and iterative method
      - If you use iterative method, you may need Stack.cpp

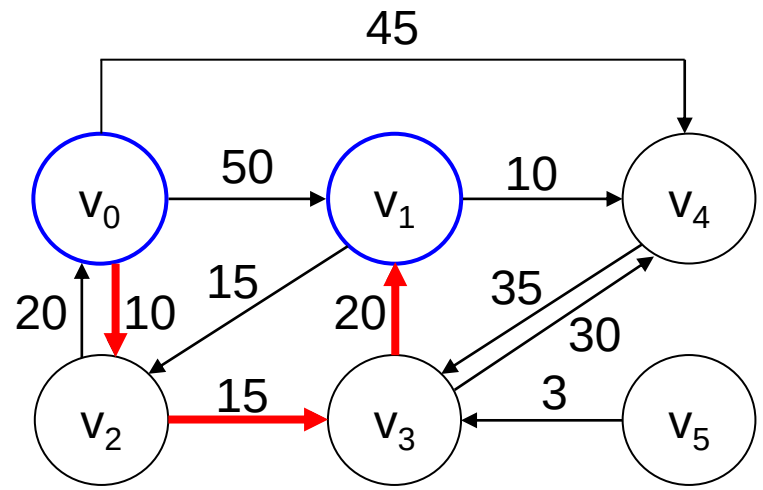
Dijkstra's Algorithm

- Expected results

```
PS C:\ds\practice12> .\DigraphMain.exe
Print Adj Matrix: 6 vertices are in use currently
     [0]  [1]  [2]  [3]  [4]  [5]
[0]   -1   50   10   -1   45   -1
[1]   -1   -1   15   -1   10   -1
[2]   20   -1   -1   15   -1   -1
[3]   -1   20   -1   -1   30   -1
[4]   -1   -1   -1   35   -1   -1
[5]   -1   -1   -1    3   -1   -1

Single Source Shortest Path from 0
Dest    Cost     Path
1       45       0 2 3 1
2       10       0 2
3       25       0 2 3
4       45       0 4
5       -1       No Path

Single Source Shortest Path from 1
Dest    Cost     Path
0       35       1 2 0
2       15       1 2
3       30       1 2 3
4       10       1 4
5       -1       No Path
```



Shortest paths starting from v_0

Destination	Path	Length
v_2	v_0v_2	10
v_3	$v_0v_2v_3$	25
v_1	$v_0v_2v_3v_1$	45
v_4	v_0v_4	45
v_5	No path	infinite

Dijkstra's Algorithm

- Dijkstra's algorithm (source vertex: *v*)

```
int i, u, w;
for (i = 0; i < n; i++) {
    found[i] = FALSE;                  // if found[i] == TRUE, i is in S
    distance[i] = cost[v][i];          // v : source vertex
}
found[v] = TRUE;                       // initially S = { v }
distance[v] = 0;
for (i = 0; i < n - 2; i++) {
    u = choose(distance, n, found); // find a vertex with minimum distance
    found[u] = TRUE;                   // add u into S
    for (w = 0; w < n; w++) {          // adjust distances to the vertices not in S
        if (!found[w] && distance[u]+cost[u][w] < distance[w])
            distance[w] = distance[u]+cost[u][w];
    }
}
```