알고리즘 과제

Practice.08

학번 : 201402432

이름 : 조디모데

8-1. Bellman-Ford Algorithm

- 알고리즘 설명

  Bellman-Ford algorithm 은 가중 유향 그래프에서 최단 경로 문제를 푸는 알고리즘이다. 이때 간선의 가중치는 음수일 수도 있다. Dijkstra's algorithm 은 벨만-포드 알고리즘과 동일한 작업을 수행하고 실행속도도 더 빠르다. 하지만 Dijkstra's algorithm 은 가중치가 음수인 경우는 처리할 수 없으므로, 이런 경우에는 벨만-포드 알고리즘을 사용한다.

  V 와 E 가 각각 그래프에서 꼭지점과 모서리의 개수라고 한다면, 벨만-포드 알고리즘의 실행시간은 O(VE)이다.

- 컴파일 방법

  graph_sample_bellman.txt 파일을 바탕화면에 넣는다

  C:\\Users\\Administrator\\Desktop\\graph_sample_bellman.txt

● Code (.C)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <memory.h>
#include <math.h>

/* bellman-ford.c57. C57 code to run the Bellman-Ford
algorithm on a
   small directed graph. */

#include <stdio.h>

/* Relax edge (u,v) with weight w. */
void relax(int u, int v, double w, double d[], int pi[]) {
  if (d[v] > d[u] + w) {
    d[v] = d[u] + w;
    pi[v] = u;
  }
}

/* Initialize a single-source shortest-paths computation. */
void initialize_single_source(double d[], int pi[], int s, int n)
{
  int i;
  for (i = 1; i <= n; ++i) {
    d[i] = 1000000000.0;
    pi[i] = 0;
  }

  d[s] = 0.0;
}

/* Run the Bellman-Ford algorithm from vertex s. Fills in
arrays d
   and pi. */
int bellman_ford(int first[], int node[], int next[], double w[],
double d[],
             int pi[], int s, int n) {
  int u, v, i, j;

  initialize_single_source(d, pi, s, n);

  for (i = 1; i <= n-1; ++i) {
    for (u = 1; u <= n; ++u) {
      j = first[u];

      while (j > 0) {
       v = node[j];
```

```
      relax(u, v, w[j], d, pi);
      j = next[j];
     }
    }
  }

  for (u = 1; u <= n; ++u) {
    j = first[u];

    while (j > 0) {
      v = node[j];
      if (d[v] > d[u] + w[j])
            return 0;
      j = next[j];
    }
  }

  return 1;
}

void printArray(int* arr)
{
      int i ;
    for(i=0 ; i<31 ; i++){
            printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;
      }

}

void getArr(int*num){
      FILE *fps ;
      int temp = 0 , i ;
      int n1, n2, n3 ;

      // input의C 숫ù y자U들ì e을¡í 배ö e열¯ ¡© num에¯ ¢ ® 저u장a하I는¥ A
부¬I분¬¨¢
      fps =
fopen("C:\\Users\\Administrator\\Desktop\\알ú E고Æi8주O\\grap
h_sample_bellman.txt","rt");
      fscanf(fps,"%d",&temp) ;
      num[0] = temp ;
      for(i=1; i < 31 ; i=i+3){
            fscanf(fps,"%d %d %d",&n1, &n2, &n3) ;
            num[i] = n1 ;
            num[i+1] = n2 ;
            num[i+2] = n3 ;
      }

      fclose(fps) ;

}

int main(void)
{
```

```
	int first[6], node[11], next[11], pi[6], num[31];
	double w[11], d[6];
	int s;
	int i, j;
	int ok;
	clock_t start,end ;

	getArr(num) ; // Get Input Data

	// clock start
	start = clock() ;

	// search
	first[1] = 1 ;
	first[2] = 3 ;
	first[3] = 6 ;
	first[4] = 8 ;
	first[5] = 9 ;

	j = 0 ;
	for(i=1 ; i<31 ; i=i+3){
		node[j] = num[i] ;
		next[j] = num[i+1] ;
		w[j] = num[i+2] ;
		j++ ;
	}

	for(i=1 ; i<6 ; i++){
		ok = bellman_ford(first, node, next, w, d, pi, i, 5);

		printf("bellman_ford returns ");
		printf("%d\n\n", ok);

		for (i = 1; i <= 5; ++i) {
			printf("%d: %f  %d\n", i, d[i], pi[i]);
		}
	}

	// clock stop
	end = clock() ;

	// print using time
	printf("실ö C행a시ö A간Æ¡Ì  : %lf초E\n",(end-
start)/(double)1000) ;

	// print result
	//printf("-------- 정¢ ´렬¤A 결Æa과Æu -------- \n") ;
	//printArray(num) ;

	system("pause") ;

   return 0;
}
```

8-3. Dijkstra's algorithm

- 알고리즘 설명

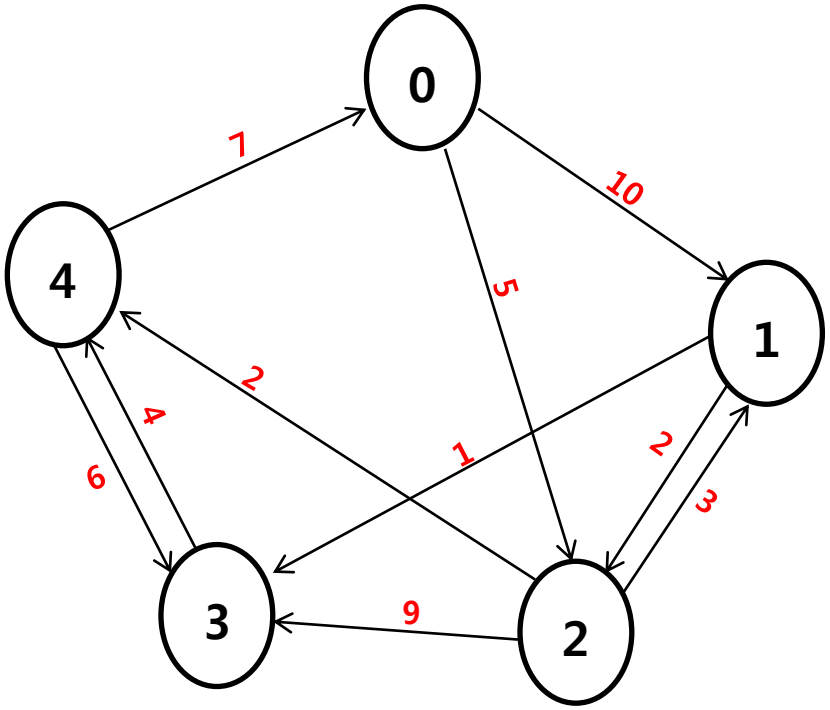
  방향성 있는 그래프에서 임의의 두 노드 간의 최단 거리(간선의 가중치 합)이 가장 적은 경로를 찾는 알고리즘이다.

- 컴파일 방법

  graph_sample_dijkstra.txt 파일을 바탕화면에 넣는다.

  C:\\Users\\Administrator\\Desktop\\graph_sample_dijkstra.txt

- **Shortest Path**



<직접 찾은 값>

0 to 1 : 0 – 2 – 1 : 8 (5+3)

0 to 2 : 0 – 2 : 5

0 to 3 : 0 – 1 – 3 : 11 (10+1)

0 to 4 : 0 – 2 – 4 : 7 (5+2)

1 to 0 : 1 – 2 – 4 – 0 : 11 (2+2+7)

1 to 2 : 1 – 2 : 2

1 to 3 : 1 – 3 : 1

1 to 4 : 1 – 2 – 4 : 4 (2+2)

2 to 0 : 2 – 4 – 0 : 9 (2+7)

2 to 1 : 2 – 1 : 3

2 to 3 : 2 – 4 - 3 : 8 (2+6)

2 to 4 : 2 – 4 : 2

3 to 0 : 3 – 4 - 0 : 11 (4+7)

3 to 2 : 3 – 4 – 0 – 2 : 16 (4+7+5)

3 to 4 : 3 – 4 : 4

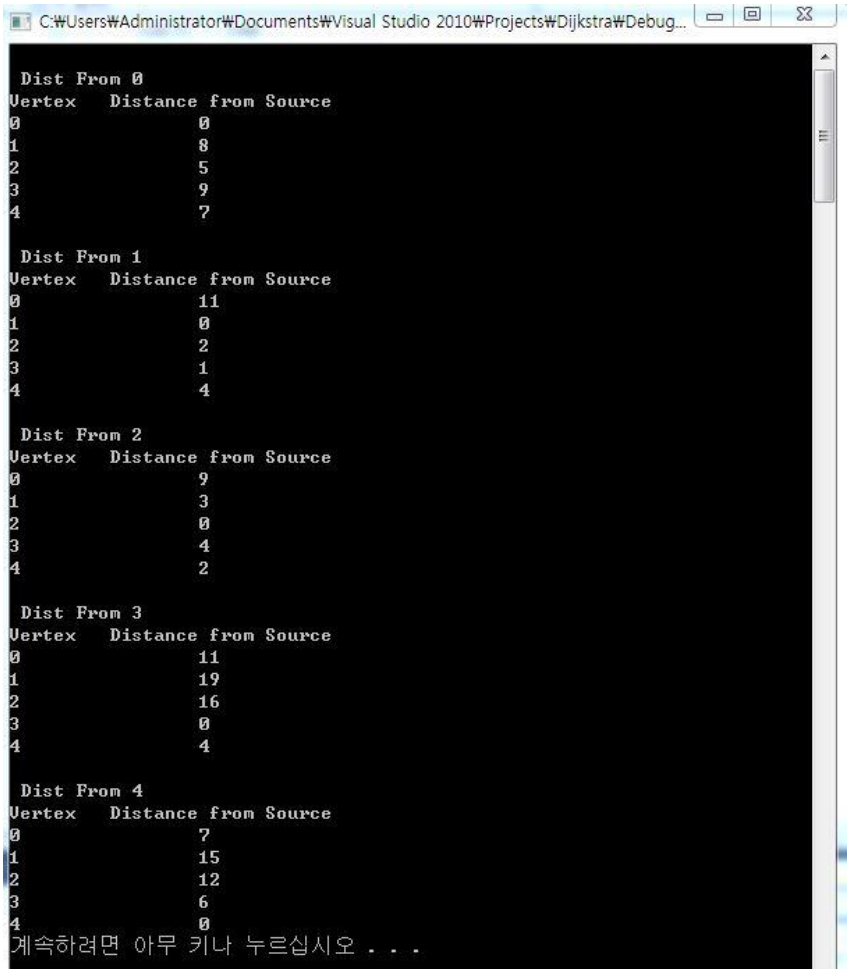
4 to 0 : 4 – 0 : 7

4 to 1 : 4 – 0 – 2 – 1 : 15 (7+5+3)

4 to 2 : 4 – 0 – 2 : 12 (7+5)

4 to 3 : 4 – 3 : 6

<프로그램 결과값>



● Code (.Cpp)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <iostream>

// Number of vertices in the graph
#define V 5

int minDistance(int dist[], bool sptSet[]){
  // Initialize
  int min = INT_MAX, min_index;

  for (int v = 0; v < V; v++)
    if (sptSet[v] == false && dist[v] <= min)
      min = dist[v], min_index = v;

  return min_index;
}

void printSolution(int dist[], int n)
{
```

```
  printf("Vertex   Distance from Source\n");
  for (int i = 0; i < V; i++)
    printf("%d \t\t %d\n", i, dist[i]);

}

void dijkstra(int graph[V][V], int src){
    int dist[V];

    bool sptSet[V];

    for (int i = 0; i < V; i++)
      dist[i] = INT_MAX, sptSet[i] = false;

    dist[src] = 0;

    for (int count = 0; count < V-1; count++){
      int u = minDistance(dist, sptSet);

      sptSet[u] = true;

      for (int v = 0; v < V; v++)
        if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX &&
dist[u]+graph[u][v] < dist[v])
          dist[v] = dist[u] + graph[u][v];
    }
    printSolution(dist, V);
}

void getArr(int*num){
      FILE *fps ;
      int temp = 0 , i ;
      int n1, n2, n3 ;

      // input의C 숫ù y자U들ì e을¡í 배ö e열¯ ¡© num에¯ ¢ ® 저u장a하I는¥ A
부¬I분¬¨¢
      fps =
fopen("C:\\Users\\Administrator\\Desktop\\알ú E고Æi8주O\\grap
h_sample_dijkstra.txt","rt");
      fscanf(fps,"%d",&temp) ;
      num[0] = temp ;
      for(i=1; i < 31 ; i=i+3){
            fscanf(fps,"%d %d %d",&n1, &n2, &n3) ;
            num[i] = n1 ;
            num[i+1] = n2 ;
            num[i+2] = n3 ;
      }

      fclose(fps) ;

}

int main(){
      int num[31], i, j ;
```

```
    int graph[V][V] ;

    getArr(num) ;
    for(i=0 ; i<5 ; i++){
          for(j=0 ; j<5 ; j++){
                graph[i][j] = 0 ;
          }
    }

    for(i=1 ; i<30 ; i=i+3){
          graph[num[i]][num[i+1]] = num[i+2] ;
    }

    for(i=0 ; i<5 ; i++){
          printf("\n Dist From %d\n", i) ;
          dijkstra(graph, i);
    }

    system("pause");

  return 0;

}
```