*11주차 : Bellman-Ford Algorithm*

# 알 고 리 즘

2015. 11. 19.

충남대학교 컴퓨터공학과 임베디드 시스템 연구실
TA 권진세

# Overview

## ▶ 이번 주 실습 / 과제

1) Dijkstra's Algorithm(shortest path) 구현하기

=> Bellman-Ford Algorithm 구현하기

# BELLMAN-FORD Algorithm

## ▶ BELLMAN-FORD ALGORITHM

1) 벨만포드 알고리즘

- 간선의 가중치가 음수인 경우
- 단일 출발지 최단 경로 문제 해결
- 음의 가중치를 갖는 순환이 있는 논리값 리턴한다.
    - 순환이 존재 하면 False 리턴
    - 순환이 존재 하지 않으면 최단 경로 계산

# Bellman-Ford Algorithm

```
BELLMAN-FORD ( G, w, s )
    INITIALIZE-SINGLE-SOURCE (G, s)
    for  i = 1 to |G.V| - 1
        for  각 간선 ( u, v ) ∈ G.E
            RELAX (u, v, w)
    for  각 간선 (u, v) ∈ G.E
        if v.d > u.d + w (u, v)
            return False
    return True
```

V = 정점
E = 간선
G(그래프) = (V, E)

s = 출발점
u = 중간 정점
v = 정점

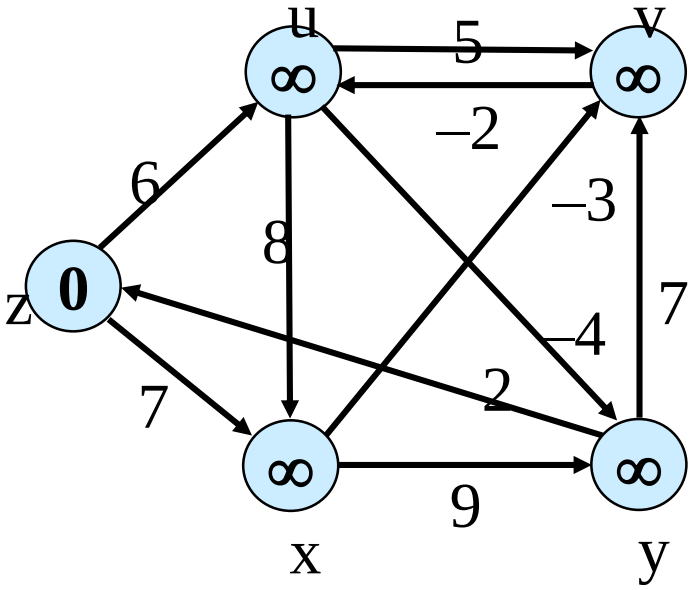
.d = distance(거리)
w = 경로가중치

# BELLMAN-FORD Algorithm

## ▶ BELLMAN-FORD ALGORITHM

2) Relax (완화)

- 각 정점 사이의 d를 최대값으로 초기화 한다.
- 각 간선의 가중치를 계산하여 완화한다.

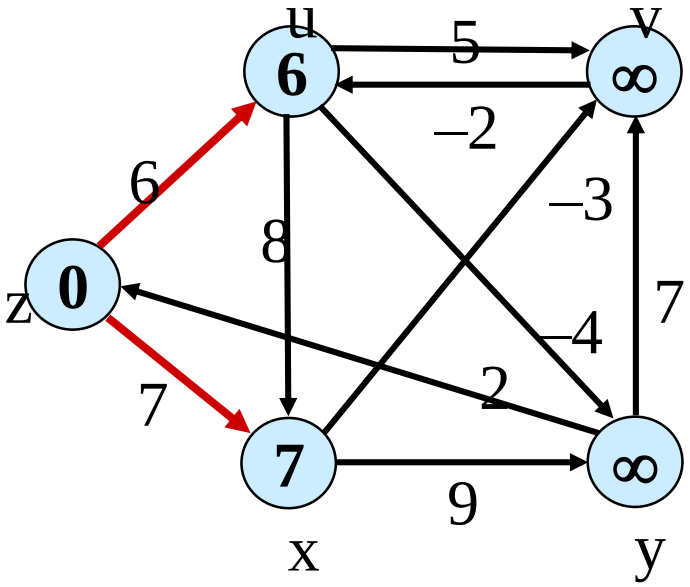


# BELLMAN-FORD Algorithm

## ▶ BELLMAN-FORD ALGORITHM

2) Relax (완화)

- 각 정점 사이의 d를 최대값으로 초기화 한다.
- 각 간선의 가중치를 계산하여 완화한다.



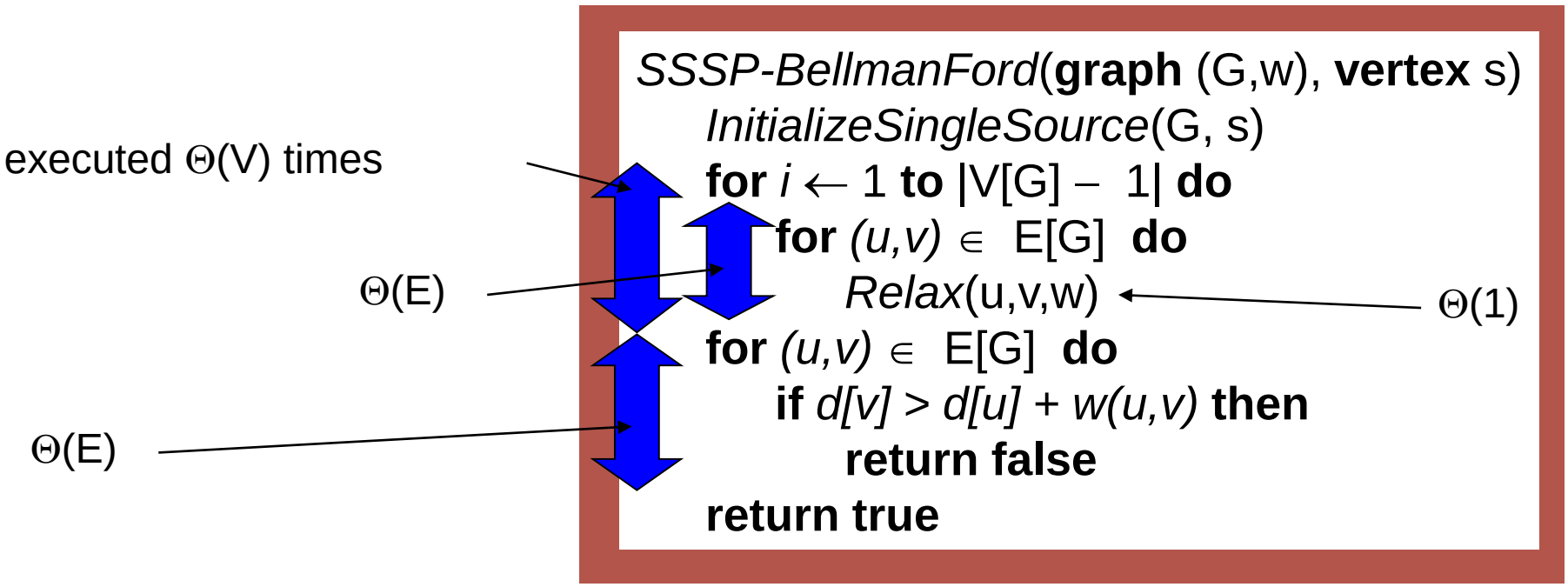
# BELLMAN-FORD Algorithm

## ▶ BELLMAN-FORD ALGORITHM

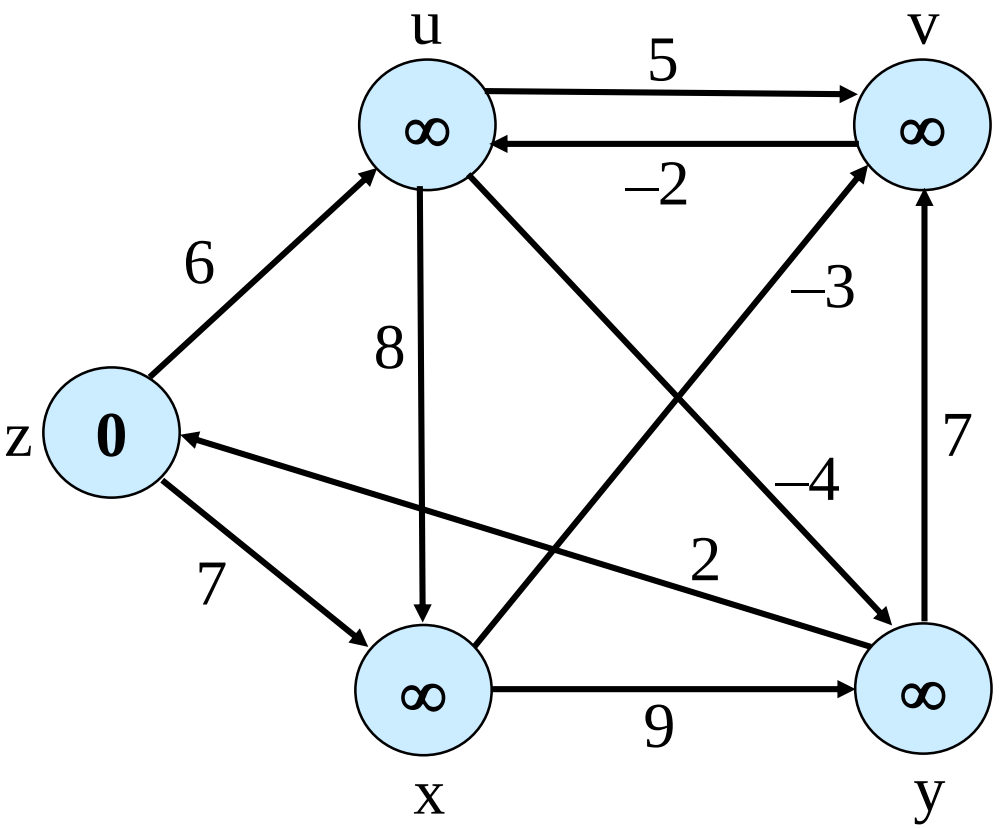
2) Relax (완화)

- 각 정점 사이의 d를 최대값으로 초기화 한다.
- 각 간선의 가중치를 계산하여 완화한다.
- 간선은 각 정점 사이에 있으므로
    - 정점 – 1 번 반복한다.

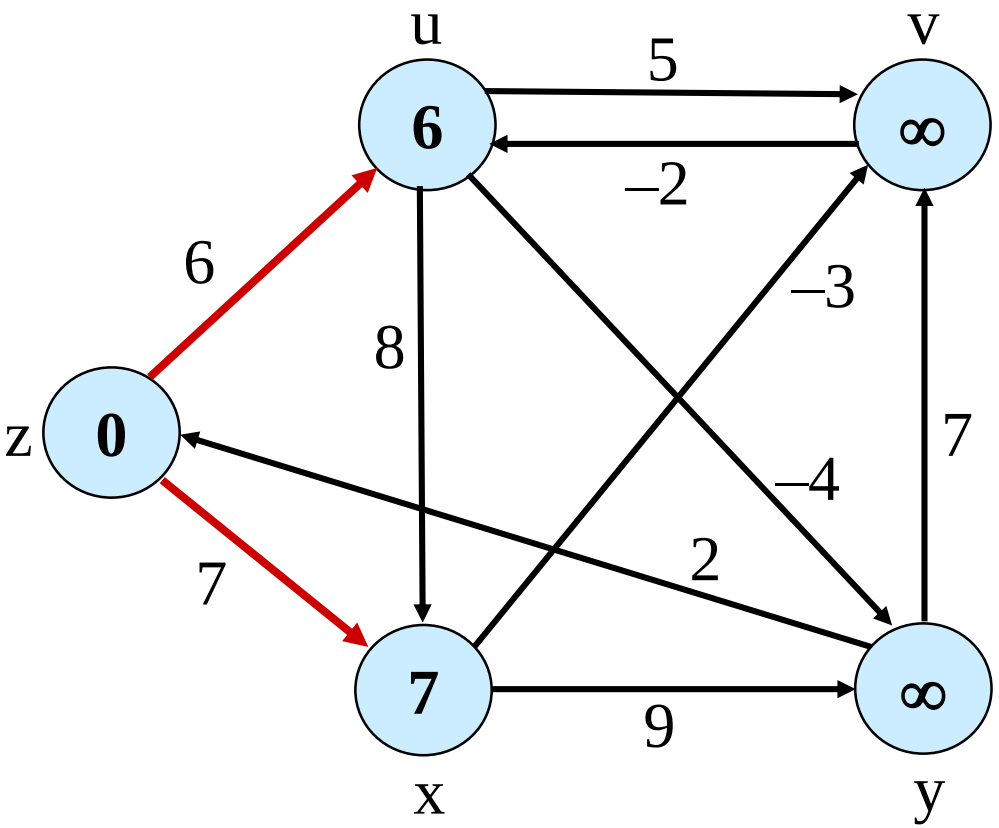
# Bellman-Ford Algorithm - Complexity



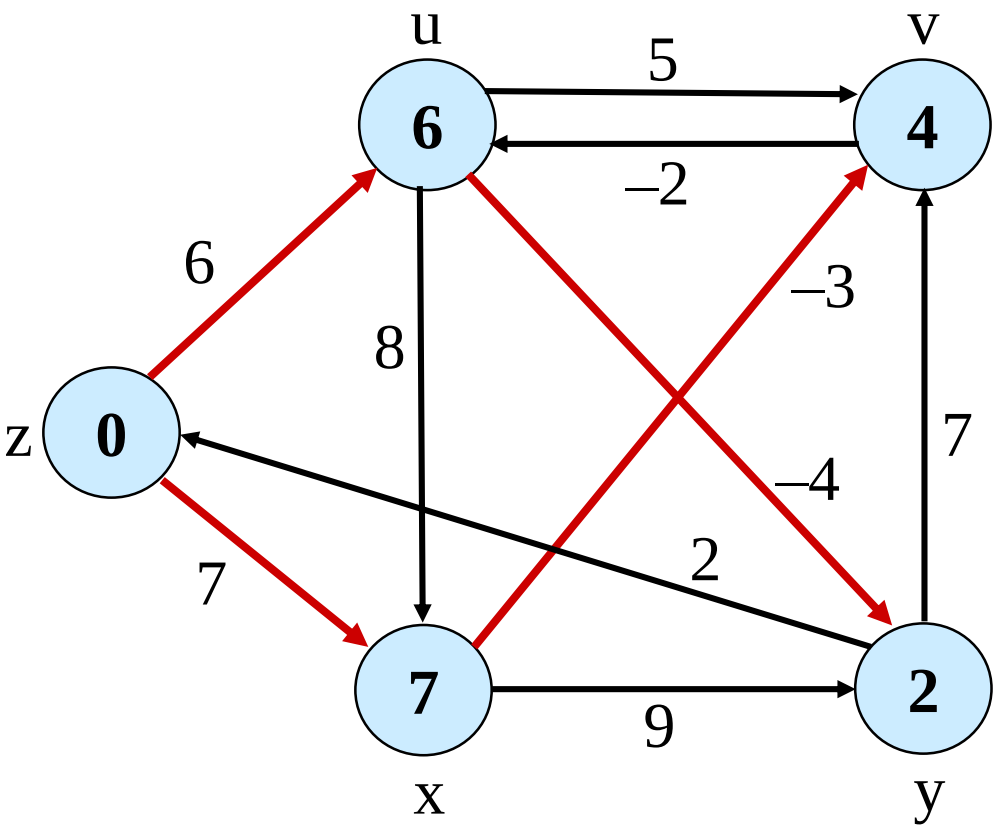
# Example



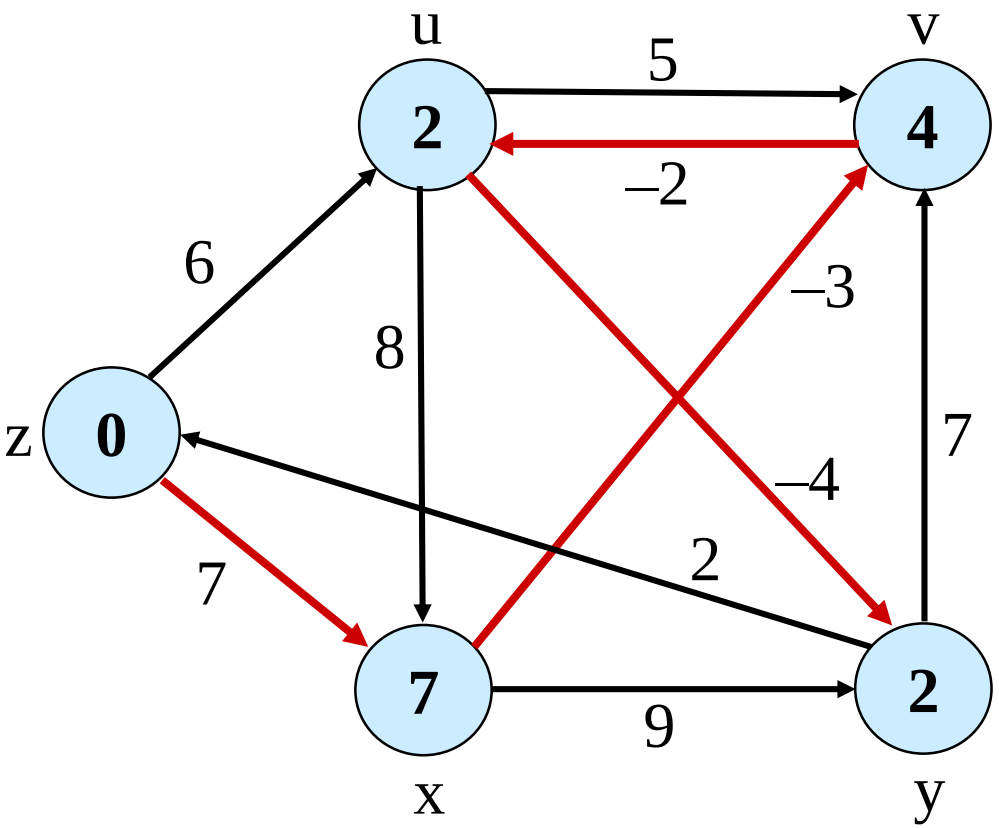
# Example



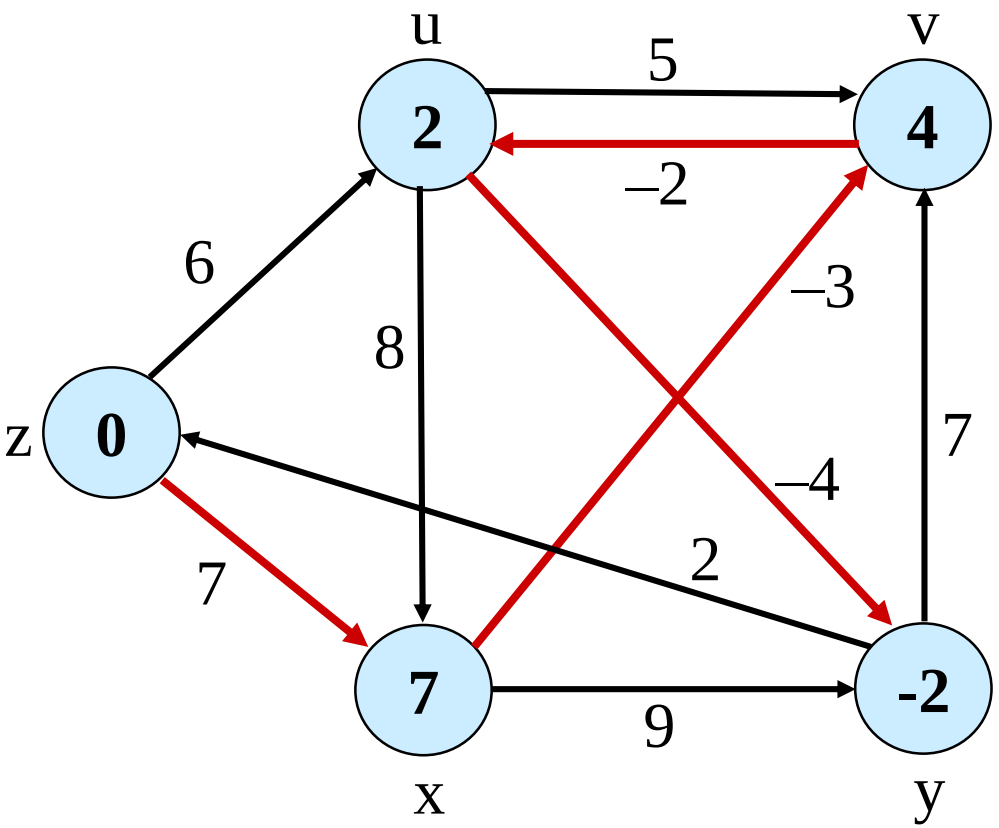
# Example



# Example



# Example

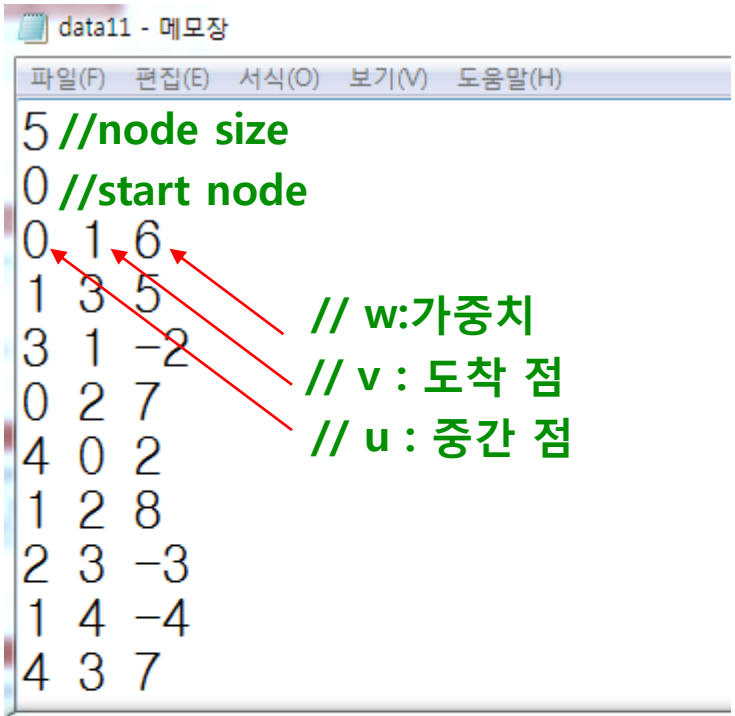


# Homework

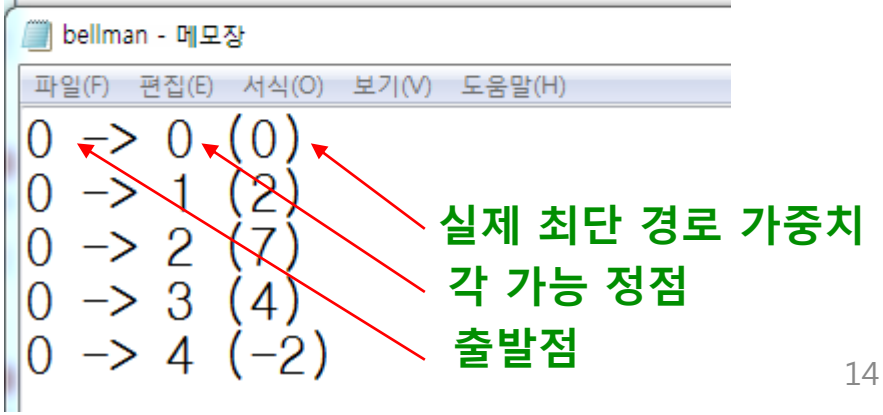
1. Bellman-Ford Algorithm

과제 예시)

Input Data :




Output Data :



# Practice / Homework

※ 그 외 실습 과제 수행 중 유의 사항

- 포함내용 : 코드만 제출
  ※ 201500000_11_bellmanford.c

- 제출이름 :
  메일 : [알고리즘00반]_201500000_홍길동_11주차
  ( 파일 : 201500000_11.zip )

- 제출기한 : 2015-11-26 18:00까지

- 메일주소 : kwonse@cnu.ac.kr

# APPENDIX 1. File I/O

1. 파일 입출력 방법

```
FILE* fp;          //fp : input file pointer
FILE* fop;         //fop : input file pointer

//파일 이름은 "00_201500000_insertion.txt"
//입출력 파일은 *.c 소스파일과 같은 폴더에 있어야 한다.
fp = fopen(FILENAME,"rt");                  //입력 파일 열기
fop = fopen(FILENAME2,"wt");                //출력 파일 열기

if (fp == NULL) {
        printf("**** Input File open error ****₩n");
        exit(1);
} //파일 없을 경우 예외처리로 프로그램 종료
```

# APPENDIX 1. File I/O (계속)

```
while(!feof(fp)){
} //파일의 끝날때 까지 반복

fscanf(fp,"%d, ", &변수);        // ex) 123, 456,
                                 // int 값만 추출함 ', '는 제외됨

fprintf(fop, "%d", 출력할 값);  // 파일 출력 시 사용

fclose(fp);
```

# APPENDIX 2. 배열 넘기기

1. Main 함수의 배열을 주소로 넘겨서 다룰 때 !! 이중포인터 사용
   - 장점 : 메모리 절약, 리턴 불필요.

```
#include <stdio.h>
#include <stdlib.h>

void user_malloc(int** num);

void main(void){
    int *ptr;
    user_malloc(&ptr);      //포인터 변수 ptr의 주소를 인자로 보냄.
    printf("%d₩n", *ptr);  //출력 값은 10 이다.
    return 0;
}

void user_malloc(int** num){
    *num = (int*)malloc(sizeof(int));
    (*num)[0] = 10;
}
```

# APPENDIX 3. 동적 할당 메모리 크기

## Q & A.

**포인터로 받은 배열의 크기를 구하는 방법? (있다)**

- Malloc 함수의 선언을 보면 void* malloc(size_t size)
- Size_t는 많은경우 unsinged long int로 되어있으므로
- 메모리를 할당 할 때 이 크기만큼 더 할당해서 할당 영역의 처음 부분에 길이의 값을 저장해 두고 있음.

- *(ptr - sizeof(size_t))
- **다음과 같은 함수로 만들어 사용 가능**

```
int sizeof_ar(int* S){
    int size;
    size = *(S - sizeof(int));
    return size;
}
```

# APPENDIX 4. 중간 값 찾기

3개의 원소중에 중간 값을 찾는 방법

```
int iPivot;
int ptrCenter = (ptrLeft + ptrRight) / 2;
if(!(ptrLeft < ptrCenter ^ ptrCenter < ptrRight))
    iPivot = ptrCenter;
else if(!(ptrCenter < ptrLeft ^ ptrLeft < ptrRight))
    iPivot = ptrLeft;
else
    iPivot = ptrRight;
```

# APPENDIX 5. quick sort lib func

**Quick sort library function**

**#include <stdlib.h>**

```
int compareX(const void* a, const void* b)
{
    d2_arr *p1 = (d2_arr *)a,  *p2 = (d2_arr *)b;
    return (p1->x - p2->x);
}
int compareY(const void* a, const void* b)
{
    d2_arr *p1 = (d2_arr *)a,  *p2 = (d2_arr *)b;
    return (p1->y - p2->y);
}
```

**qsort(arr, arr size, element size, compare_위에참조 )**

# APPENDIX 6. 각 자료형의 최대크기

**Variant limits 헤더**

**#include <limits.h>**
**=>정수형 변수의 최대값을 전처리 매크로로 저장한 헤더**

**#include <float.h>**
**⇒ ex) double 사이즈의 최대 크기를 알고 싶을 때**
**⇒ printf("%lf",DBL_MAX);**

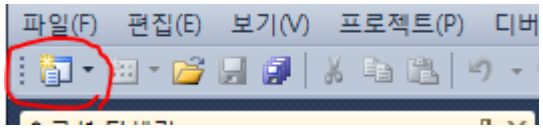
# APPENDIX 7. 2차원 배열 동적 할당

```
int input, i;

int **array = (int**)malloc(sizeof(int *)*input);

for(i=0; i<input; i++)

    array[i] = (int *)malloc(sizeof(int)*input);

//생성된 array[input][input] 을 사용

for(i=0; i<input; i++)

    free(array[i]);

free(array);
```

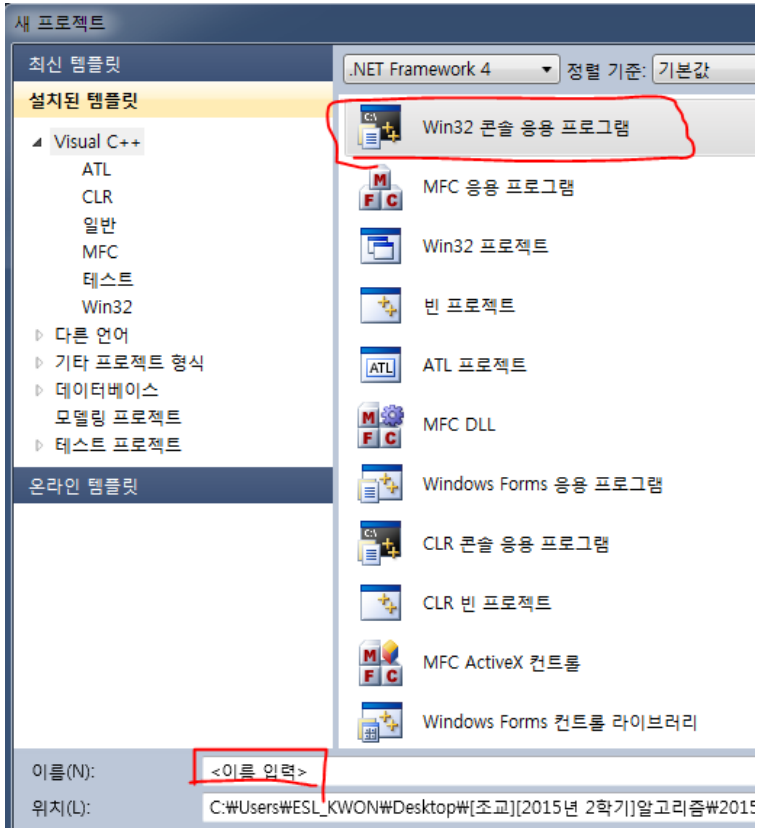
# Trouble Shooting

## Visual Studio 2010 프로젝트 생성

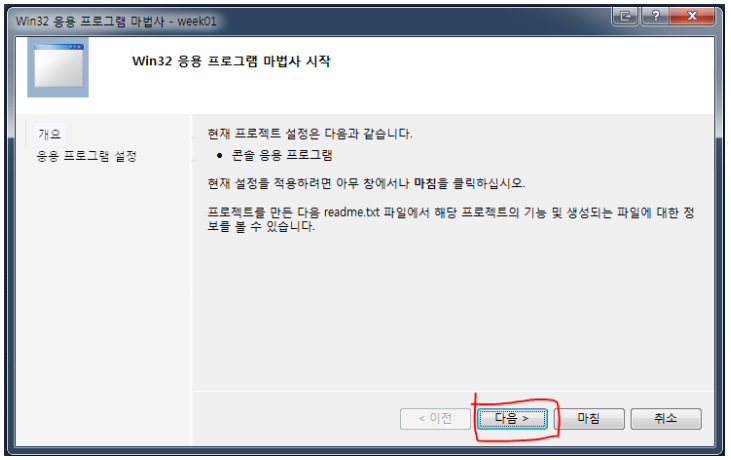
1.



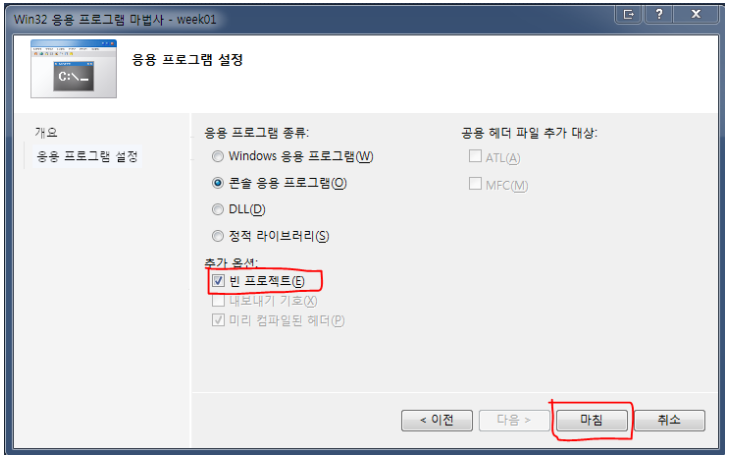

2.




3.



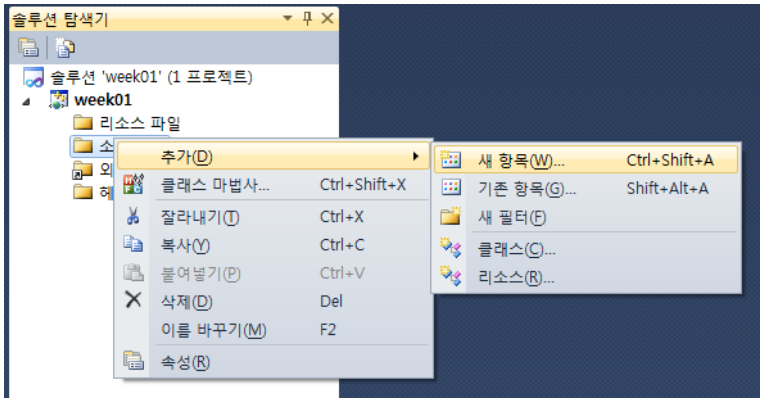

4.



# Trouble Shooting

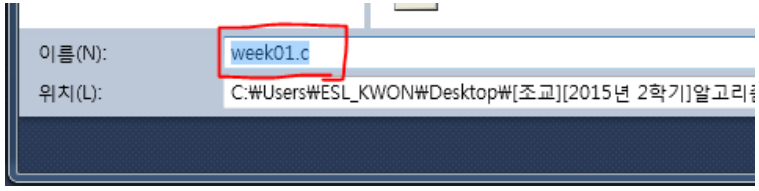
## Visual Studio 2010 프로젝트 생성

5.




6.


7.



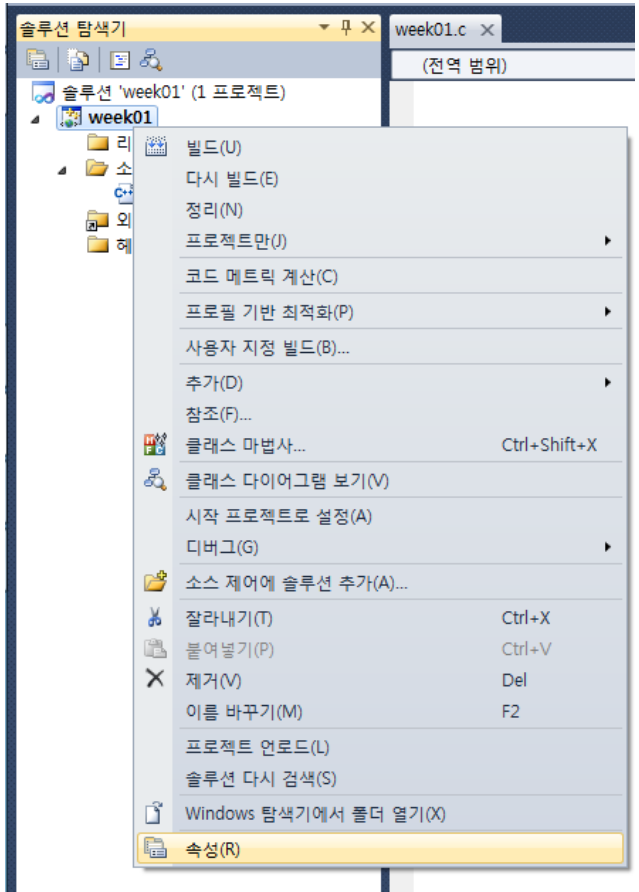

**반드시 .c 로 이름 변경!!**

# Trouble Shooting

## Visual Studio 2010 메니페스트 오류 해결

```
1>LINK : fatal error LNK1123: COFF로 변환하는 동안 오류가 발생했습니다. 파일이 잘못되었거나 손상되었습니다.
1>
1>빌드하지 못했습니다.
```

1.

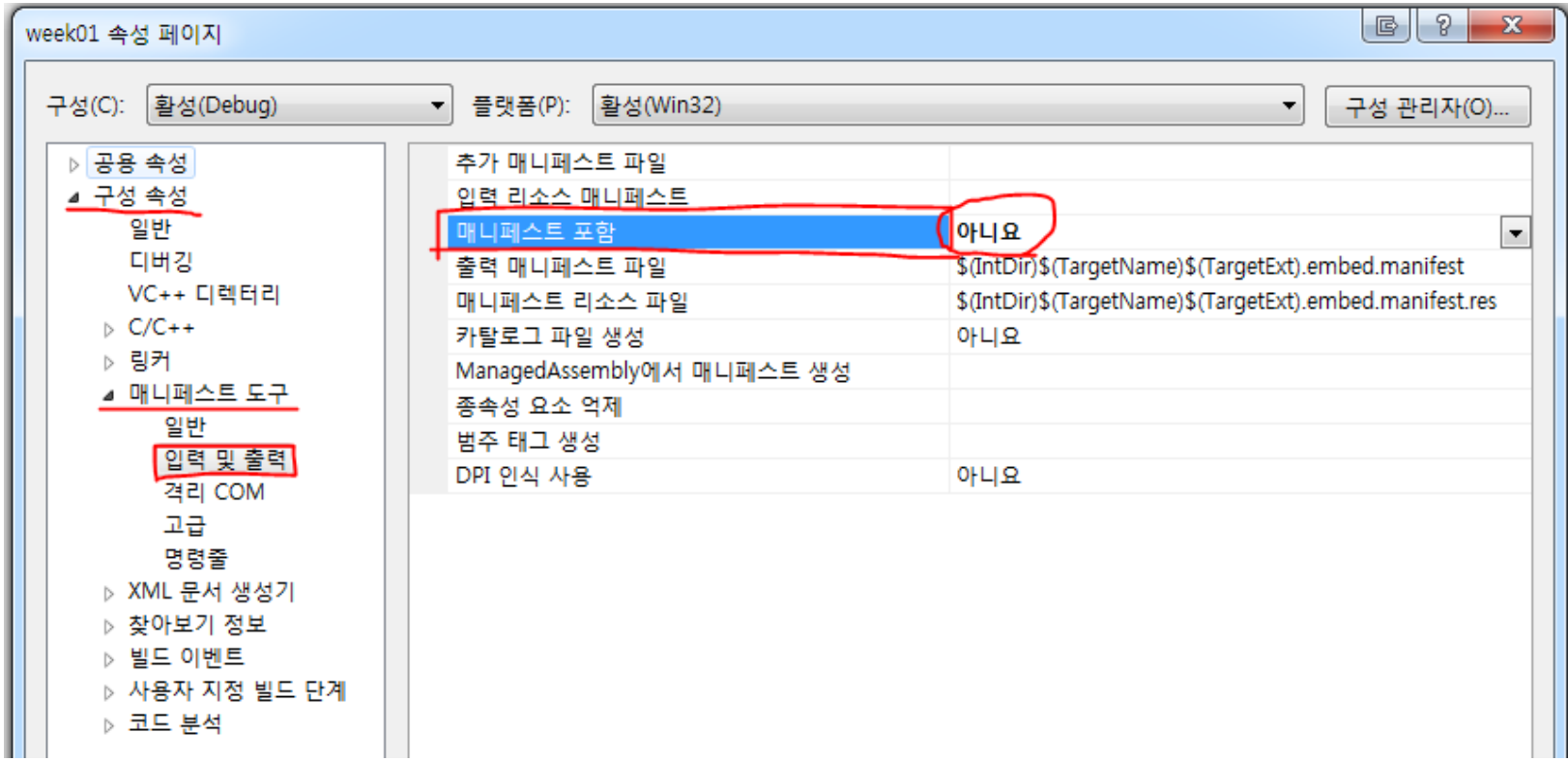



< 프로젝트 속성열기

# Trouble Shooting

**Visual Studio 2010 메니페스트 오류 해결**

**2.**




**구성 속성 -> 매니페스트 도구 -> 입력 및 출력 ->**

**메니페스트포함 : “아니요 "**

# Trouble Shooting

**Visual Studio 2010 메니페스트 오류 해결**
**3. 매니페스트 문제 영구적 해결 방법**

**Visual Studio Service Pack 1 다운로드.**
**( >600MB 오래 걸림… )**

https://www.microsoft.com/en-us/download/confirmation.aspx?id=23691