

10주차 : Dijkstra's Algorithm(shortest path) & Huffman code

알고리즘

2015. 10. 29.

충남대학교 컴퓨터공학과 임베디드 시스템 연구실
TA 권진세

Overview

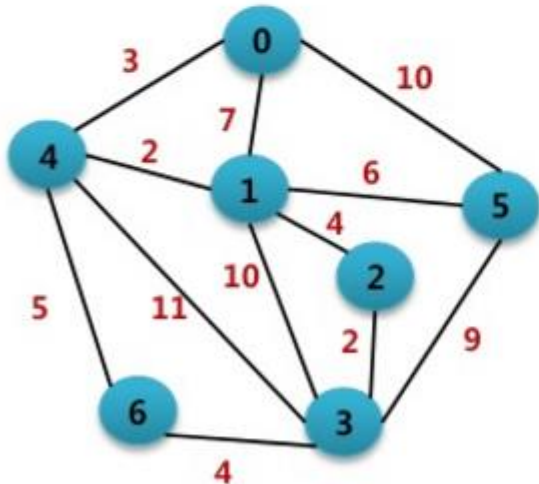
▶ 이번 주 실습 / 과제

- 1) Dijkstra's Algorithm(shortest path) 구현하기
- 2) Huffman code 구현하기

Dijkstra's Algorithm

❖ 최단 경로(Shortest Path)

- 네트워크에서 정점 i 와 정점 j 를 연결하는 경로 중에서 간선들의 가중치 합이 최소가 되는 경로를 찾는 문제
- 간선의 가중치는 비용, 거리, 시간 등을 나타낸다.

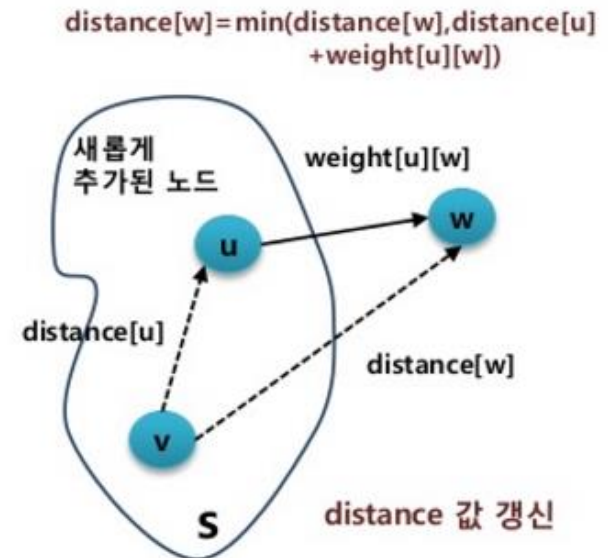
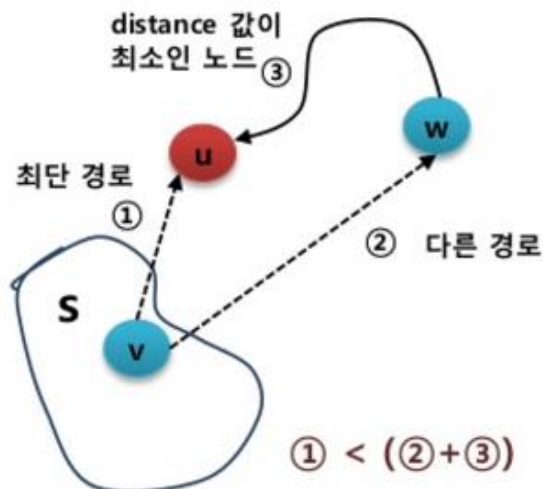
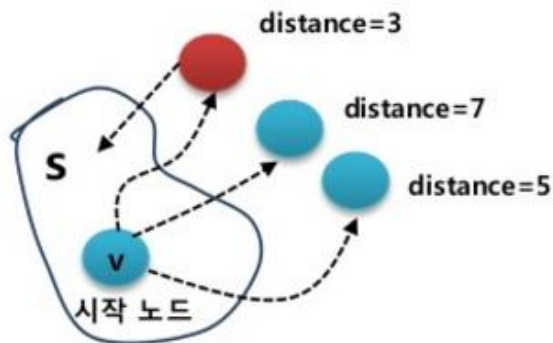


	0	1	2	3	4	5	6
0	0	7	∞	∞	3	10	∞
1	7	0	4	10	2	6	∞
2	∞	4	0	2	∞	∞	∞
3	∞	10	2	0	11	9	4
4	3	2	∞	11	0	∞	5
5	10	6	∞	9	∞	0	∞
6	∞	∞	∞	4	5	∞	0

Dijkstra's Algorithm

❖ Dijkstra의 최단 경로 알고리즘

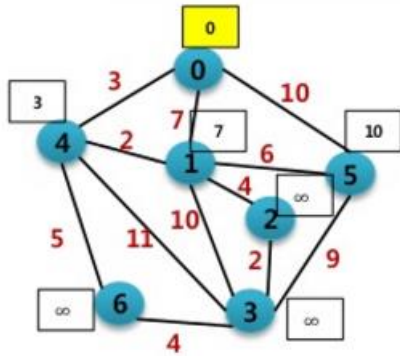
- 네트워크에서 하나의 시작 정점으로부터 모든 다른 정점까지의 최단 경로를 찾는 알고리즘
 - 집합 S: 시작 정점 v로부터의 최단경로가 이미 발견된 정점들의 집합
 - distance 배열: 최단 경로를 알려진 정점만을 통하여 각 정점까지 가는 최단경로의 길이
 - 매 단계에서 가장 distance 값이 적은 정점을 S에 추가한다



Dijkstra's Algorithm

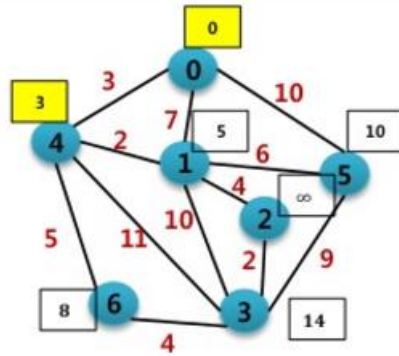
```
// 입력: 가중치 그래프 G, 가중치는 음수가 아님.  
// 출력: distance 배열, distance[u]는 v에서 u까지의 최단 거리이다.  
shortest_path(G, v)  
  
S ← {v}  
for 각 정점 w ∈ G do  
    distance[w] ← weight[v][w];  
while 모든 정점이 S에 포함되지 않으면 do  
    u ← 집합 S에 속하지 않는 정점 중에서 최소 distance 정점;  
    S ← S ∪ {u}  
    for u에 인접하고 S에 있는 각 정점 z do  
        if distance[u] + weight[u][z] < distance[z]  
            then distance[z] ← distance[u] + weight[u][z];
```

Dijkstra's Algorithm



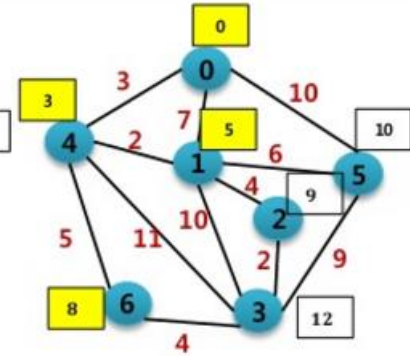
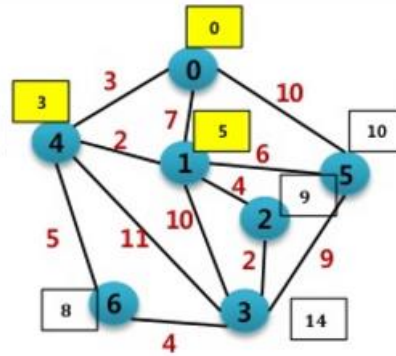
$S=\{0\}$
 distance[] =

0	7	∞	∞	3	10	∞
---	---	----------	----------	---	----	----------



$S=\{0,1\}$
 distance[] =

0	5	9	14	3	10	8
---	---	---	----	---	----	---

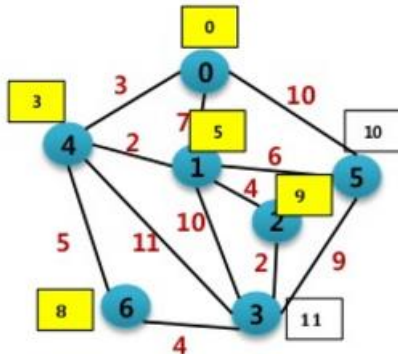


$S=\{0,4\}$
 distance[] =

0	5	∞	14	3	10	8
---	---	----------	----	---	----	---

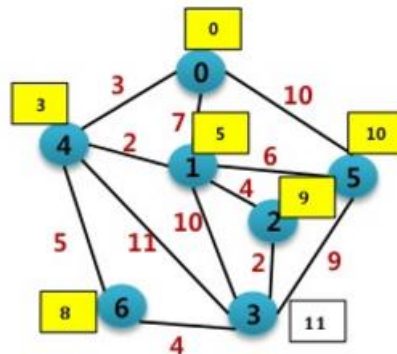
$S=\{0,4,1,6\}$
 distance[] =

0	5	9	14	3	10	8
---	---	---	----	---	----	---



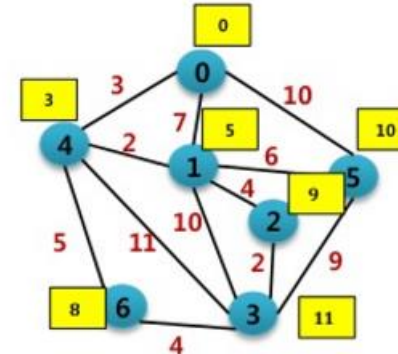
$S=\{0,4,1,6,2\}$
 distance[] =

0	5	9	11	3	10	8
---	---	---	----	---	----	---



$S=\{0,4,1,6,2,5\}$
 distance[] =

0	5	9	11	3	10	8
---	---	---	----	---	----	---



$S=\{0,4,1,6,2,5,3\}$
 distance[] =

0	5	9	11	3	10	8
---	---	---	----	---	----	---

Overview

▶ Huffman code

- *Huffman code*란?
- *Fixed – length code & Variable – length code*
- *Prefix codes (Prefix – free codes)*
- *Full binary tree*

▶ Practice & Homework

Greedy algorithm for Huffman code

Huffman code

▶ Huffman code란?

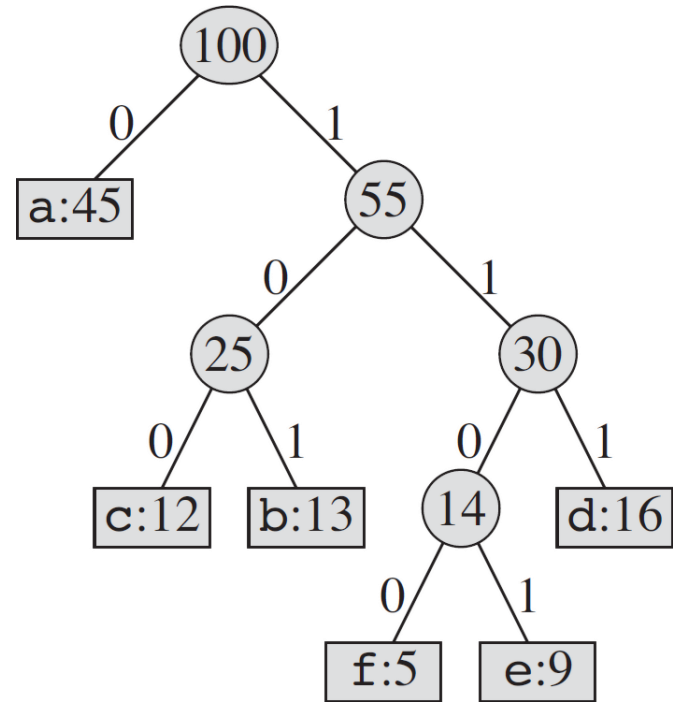
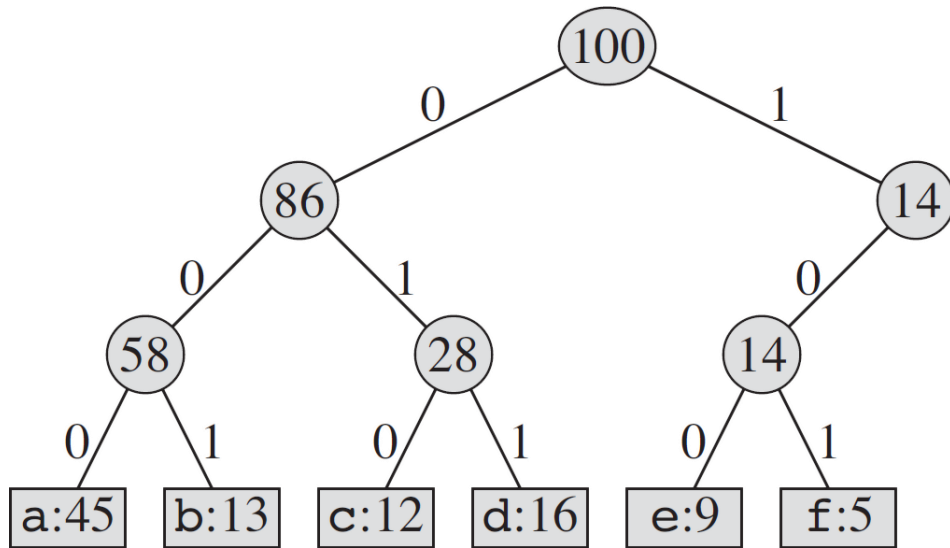
1952년, 데이비드 허프만에 의해 개발된 부호 기술로, JPEG, ZIP 등의 압축 기술에 사용되고 있다.

반복되는 개체마다 고유의 code를 부여하는데

발생빈도가 높은 것일 수록 짧은 코드를 부여하고
낮은 것일 수록 긴 코드를 부여하는 방법을 이용한다.

대상 데이터의 특성에 따라 20~90%의 공간 절약이 가능한,
널리 쓰여지고 있는 압축 기술이다.

Huffman code



	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

$$B(T) = \sum_{c \in C} c.freq \cdot d_T(c)$$

Huffman code

HUFFMAN(C)

```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```

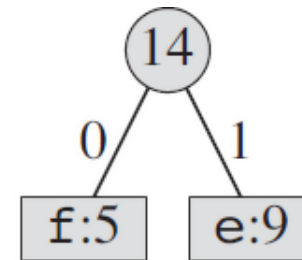
Huffman code

(a) f:5 e:9 c:12 b:13 d:16 a:45

HUFFMAN(C)

```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```

(b) c:12 b:13



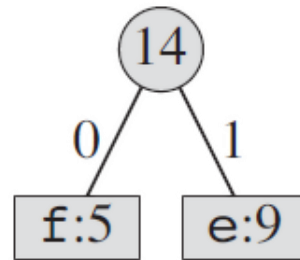
d:16 a:45

Huffman code

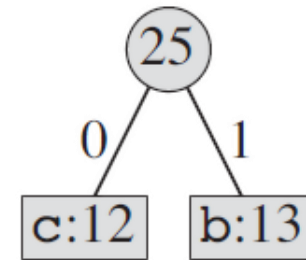
HUFFMAN(C)

```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```

(c)



d:16



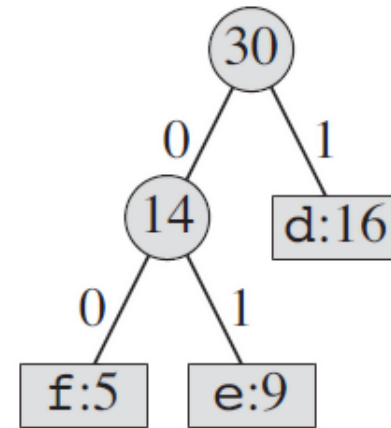
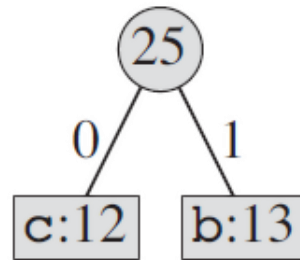
a:45

Huffman code

HUFFMAN(C)

```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```

(d)



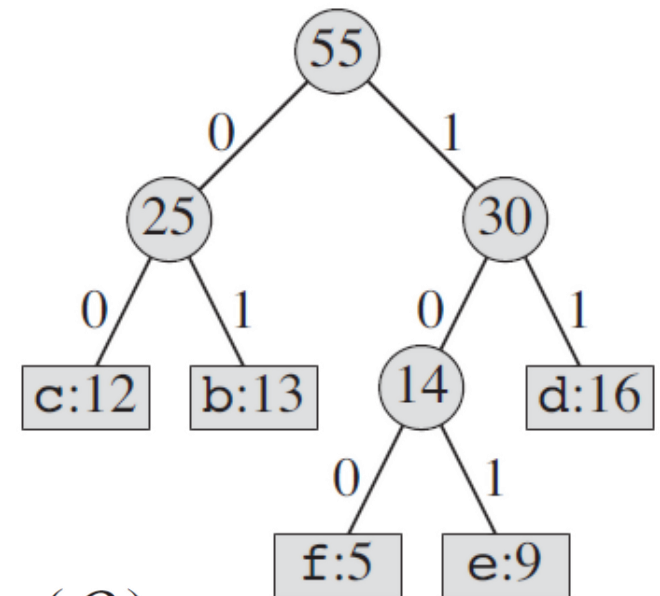
a:45

Huffman code

(e) a:45

HUFFMAN(C)

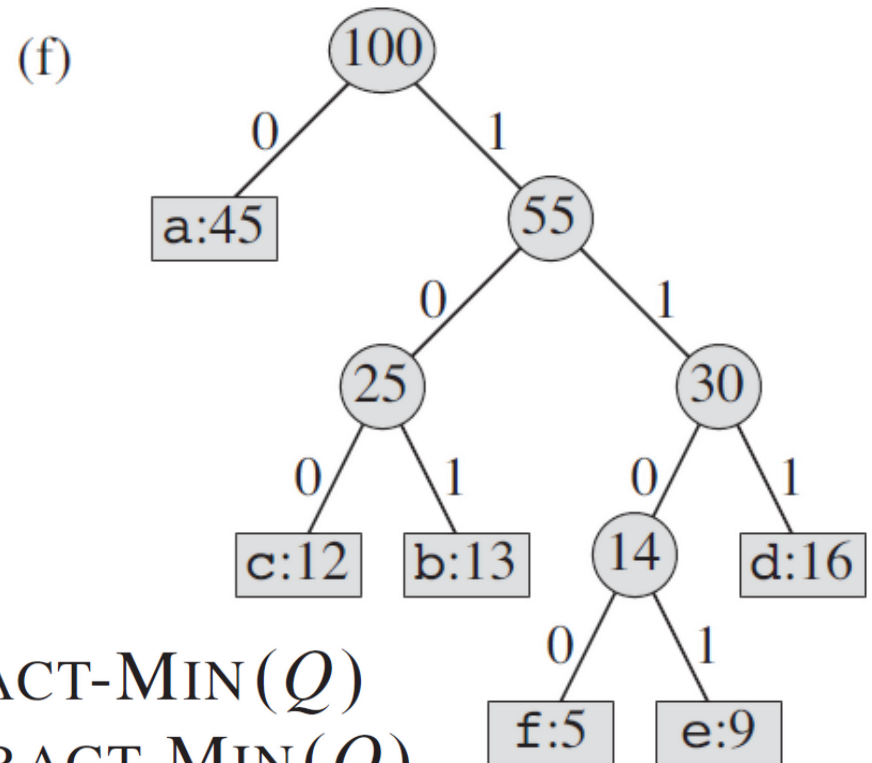
```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```



Huffman code

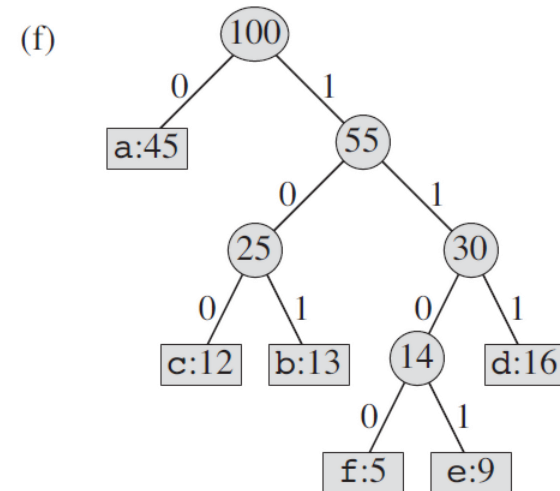
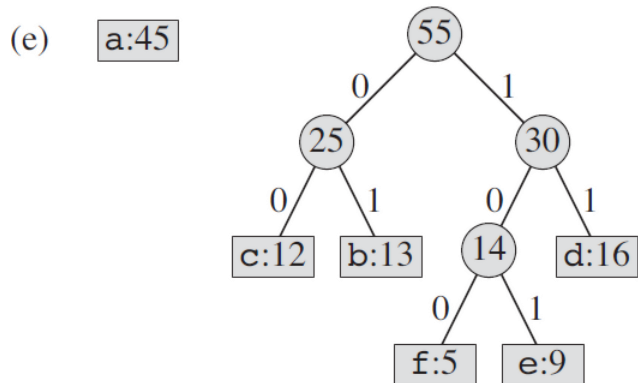
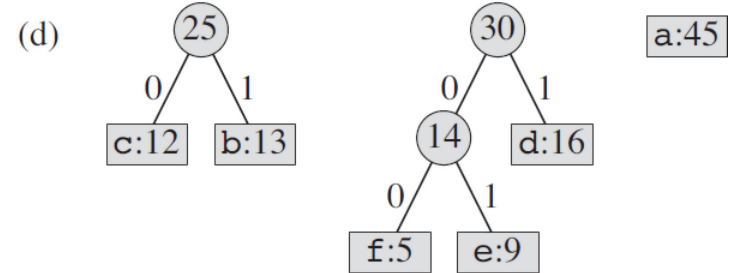
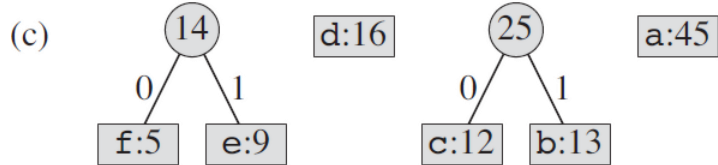
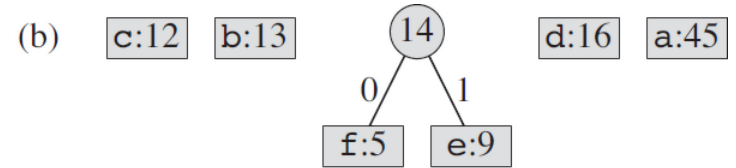
HUFFMAN(C)

```
1   $n = |C|$ 
2   $Q = C$ 
3  for  $i = 1$  to  $n - 1$ 
4      allocate a new node  $z$ 
5       $z.left = x = \text{EXTRACT-MIN}(Q)$ 
6       $z.right = y = \text{EXTRACT-MIN}(Q)$ 
7       $z.freq = x.freq + y.freq$ 
8       $\text{INSERT}(Q, z)$ 
9  return  $\text{EXTRACT-MIN}(Q)$ 
```



Huffman code

(a) f:5 e:9 c:12 b:13 d:16 a:45



Practice

1. Dijkstra's Algorithm 알고리즘 구현

실습 예시)

```
C:\Windows\system32\cmd.exe

Enter the number of nodes:7

Enter the cost matrix:
0 7 0 0 3 10 0
7 0 4 10 2 6 0
0 4 0 2 0 0 0
0 10 2 0 11 9 4
3 2 0 11 0 0 5
10 6 0 9 0 0 0
0 0 0 4 5 0 0

Enter the source matrix:3

Shortest path:
2->0,cost=9
2->1,cost=4
2->3,cost=2
2->4,cost=6
2->5,cost=10
2->6,cost=6
```

	0	1	2	3	4	5	6
0	0	7	∞	∞	3	10	∞
1	7	0	4	10	2	6	∞
2	∞	4	0	2	∞	∞	∞
3	∞	10	2	0	11	9	4
4	3	2	∞	11	0	∞	5
5	10	6	∞	9	∞	0	∞
6	∞	∞	∞	4	5	∞	0

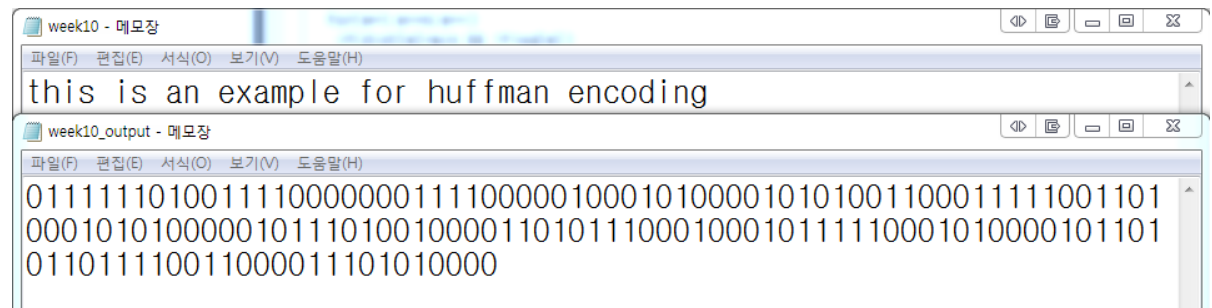
Homework

2. Huffman Code

과제 예시)

Input Data :

Output Data :



```
week10 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
this is an example for huffman encoding

week10_output - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
0111111010011110000000111100000100010100001010100110001111001101
00010101000001011101001000011010111000100010111110001010000101101
011011110011000011101010000
```

< Decoding 참고 >

' ': 000	'g': 010000	'o': 1110
'a': 1000	'h': 1101	'p': 10011
'c': 01101	'i': 0011	'r': 10010
'd': 01100	'l': 010001	's': 1100
'e': 0101	'm': 1111	't': 01111
'f': 0010	'n': 101	'u': 01110
		'x': 01001

Practice / Homework

※ 그 외 실습 과제 수행 중 유의 사항

- 포함내용 : 코드만 제출

※ 201500000_10_dijkstra.c

201500000_10_huffman.c

(두 개의 소스코드를 압축해서 보내세요)

- 제출이름 :

메일 : [알고리즘00반]_201500000_홍길동_10주차

파일 : 201500000_10.zip

- 제출기한 : 2015-11-19 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_t`는 많은경우 `unsigned long int`로 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++)
```

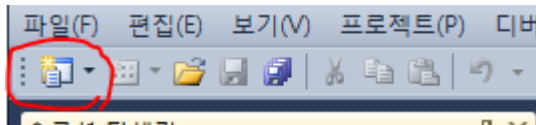
```
    free(array[i]);
```

```
free(array);
```

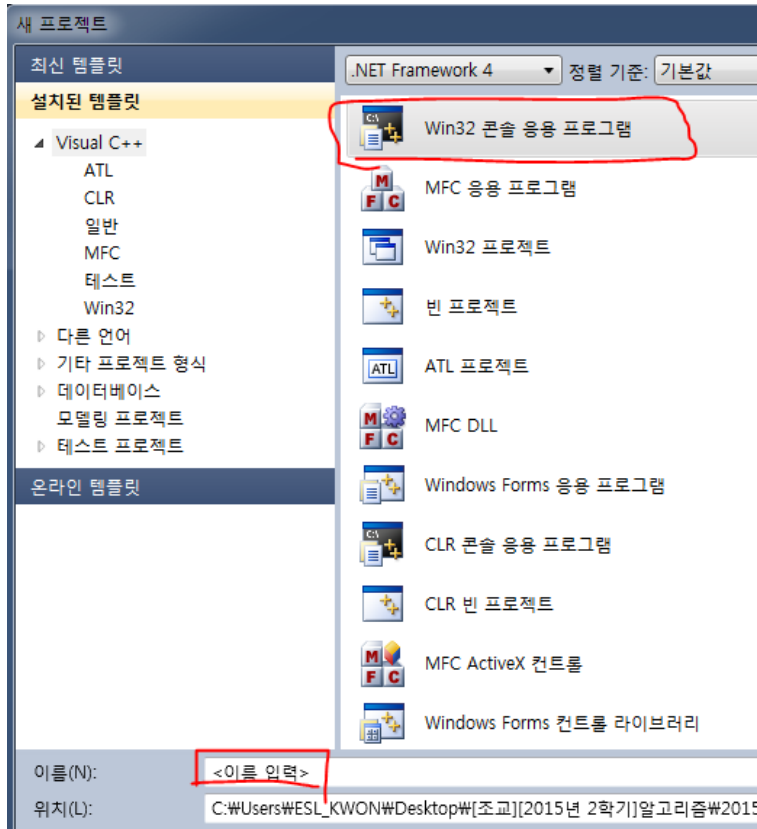
Trouble Shooting

Visual Studio 2010 프로젝트 생성

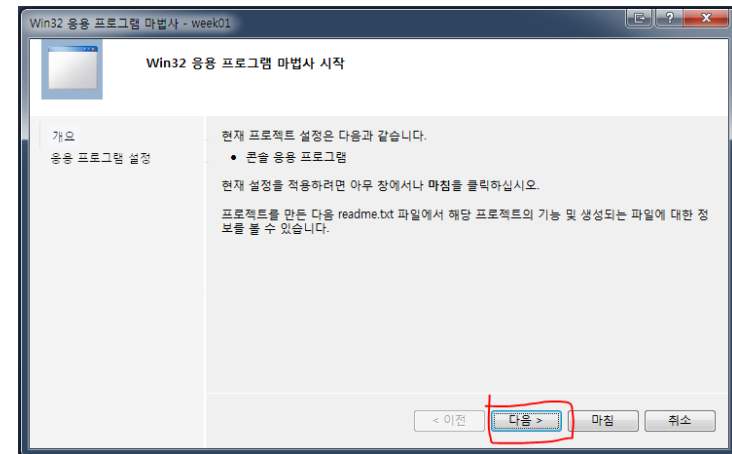
1.



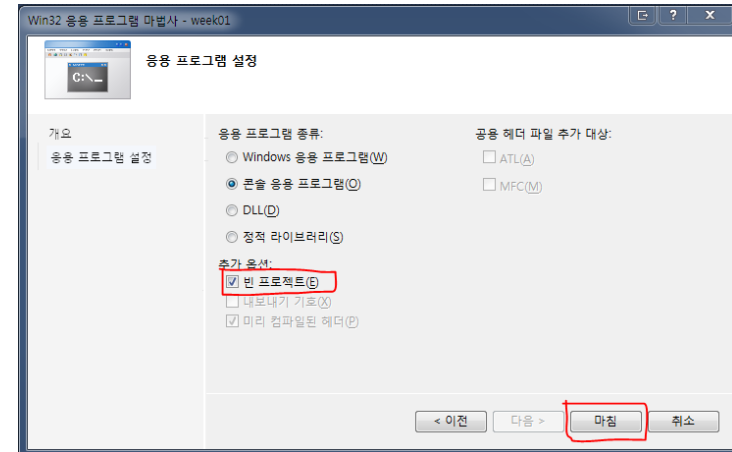
2.



3.

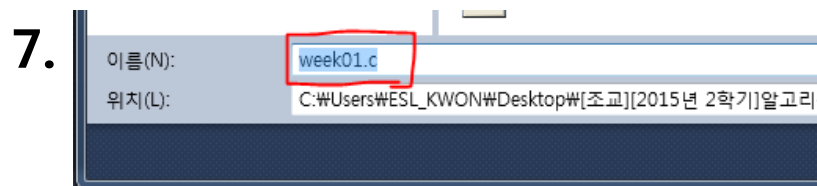
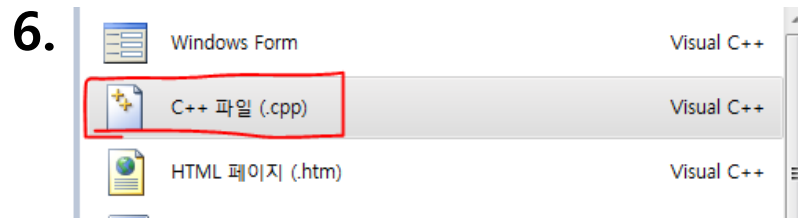
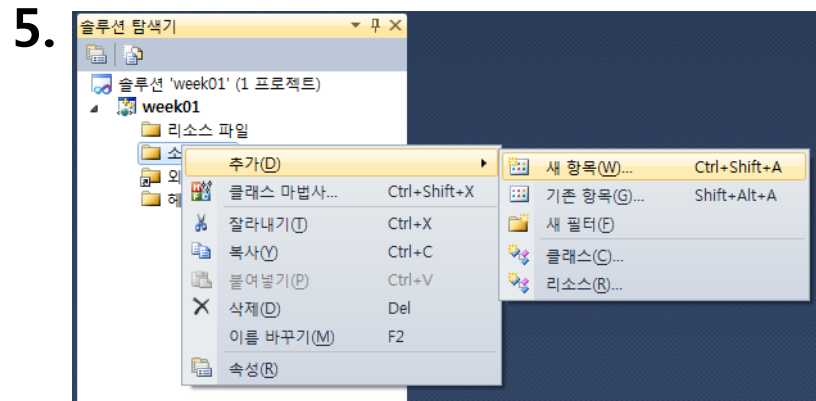


4.



Trouble Shooting

Visual Studio 2010 프로젝트 생성



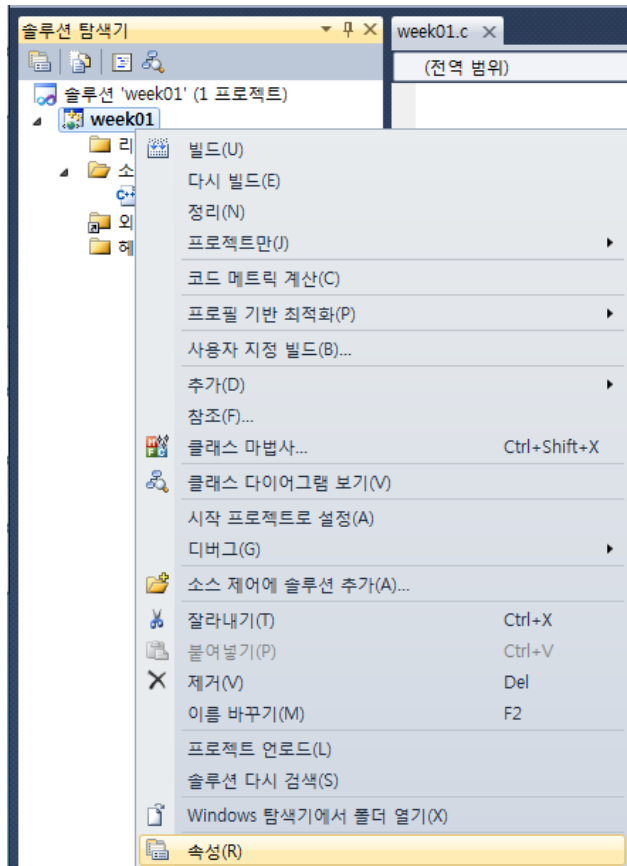
반드시 .c 로 이름 변경!!

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.

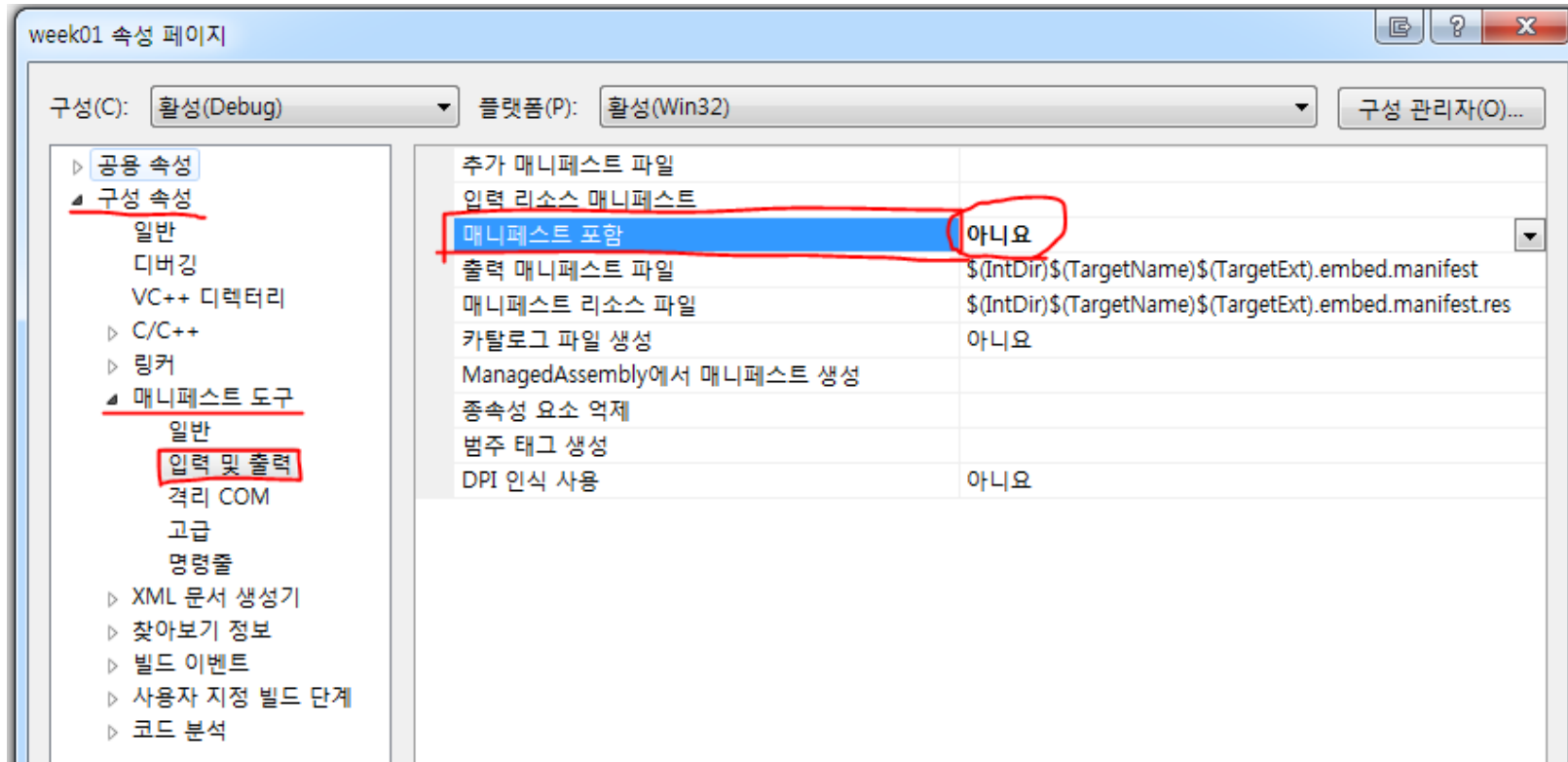


< 프로젝트 속성열기

Trouble Shooting

Visual Studio 2010 매니페스트 오류 해결

2.



구성 속성 -> 매니페스트 도구 -> 입력 및 출력 ->
매니페스트포함 : "아니요 "

Trouble Shooting

Visual Studio 2010 매니페스트 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>