제 4 장　욕심쟁이 방법 (계속)

- 최적 합병 패턴(호프만 코드)
- 배낭 문제
- 트리 정점 분할

4.5 최적 합병 패턴(호프만 코드)

- 합병(merge)은 두 개의 정렬된 리스트(혹은 파일)를 하나의 큰 정렬된 리스트로 만드는 작업이다. 이때 리스트의 길이가 각각 m, n 일 때, 합병에 걸리는 시간은 O(n+m) 이다.
- 여러 개의 정렬된 리스트들이 존재할 때, 이들을 둘씩 쌍으로 묶어 계속적으로 합병함으로써 전체적인 합병을 이룰 수 있다.
- 예제:

  파일 (x_1, x_2, x_3) 의 길이는 (30, 20, 10)이며 세 개의 정렬된 파일이다.

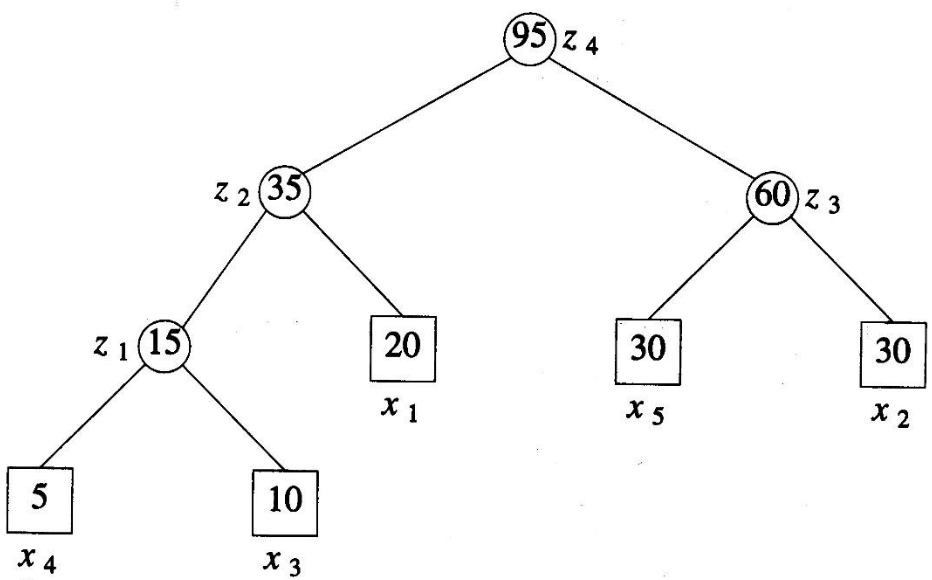
  순서 1: $\underbrace{x_1\text{과 } x_2\text{를 합병한}}_{30 + 20 = 50}$ 후 $\underbrace{\text{이 결과와 } x_3 \text{ 를 합병}}_{50 + 10 = 60}$했을 때 $\underbrace{\text{총 시간}}_{50+60=110}$

  순서 2: $\underbrace{x_2\text{와 } x_3\text{을 합병한}}_{20 + 10 = 30}$ 후 $\underbrace{\text{이 결과와 } x_1 \text{ 을 합병}}_{30 + 30 = 60}$했을 때 $\underbrace{\text{총 시간}}_{30+60=90}$

  → 합병하는 순서에 따라 전체 합병 시간이 서로 다를 수 있다.
- 이것은 n 개의 정렬된 파일들을 합병하는 가장 적은 시간이 소요되는 최적의 방법을 결정하는 문제이다.

욕심쟁이 방법

- 어떤 파일이 합병되면, 그 합병된 파일이 또 다른 합병에 사용될 때마다 원래 파일의 길이는 합병 시간에 영향을 미친다.
- 욕심쟁이 전략: 각 단계에서 가장 작은 크기의 파일들을 먼저 합병한다. 2원 합병인 경우, 2개의 가장 작은 파일을 먼저 합병하고, k-원 합병인 경우, k 개의 가장 작은 파일들을 먼저 합병한다.
- 예: 길이가 (20, 30, 10, 5, 30) 인 5 개 파일의 이진 합병 트리



이진 합병 트리와 최적 합병 패턴

- 이진 합병 트리에서 사각형 노드는 외부 노드, 원형 노드는 내부 노드이다.
- 가중 외부 경로 길이(weighted external path length):

  $\Sigma_{1\le i\le n} d_i q_i$, d_i 는 루트 노드에서 파일 x_i의 외부 노드 까지의 거리

  q_i 는 파일 x_i의 길이
- 가중 외부 경로 길이는 이진 합병 트리에 대한 레코드 이동의 총 수와 같다(총 합병에 필요한 시간).
- 최적 2-원 합병 패턴은 최소 가중 외부 경로 길이를 갖는 이진 합병 트리에 대응한다.

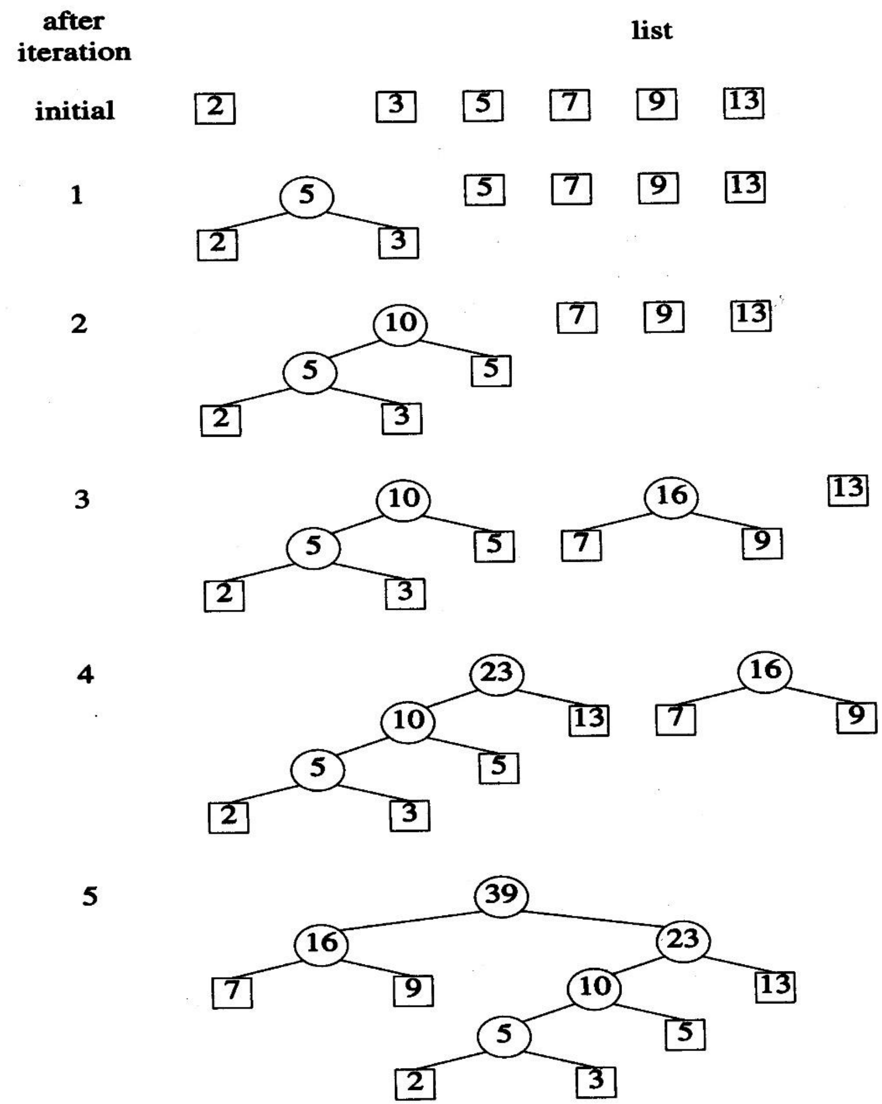
```
struct treenode {        // C++ 버전
        struct treenode *lchild, *rchild;
        int weight;
};
typedef struct treenode Type;

  Type *Tree(int n)
  // list is a global list of n single node
  // binary trees as described above.
  {
     for (int i=1; i<n; i++) {
        Type *pt = new Type;
        // Get a new tree node.
        pt -> lchild = Least(list); // Merge two trees with
        pt -> rchild = Least(list); // smallest lengths.
        pt -> weight = (pt->lchild)->weight + (pt->rchild)->weight;
        Insert(list, *pt);
     }
     return (Least(list)); // Tree left in l is the merge tree.
  }
```

- 시간 복잡도: Least와 Insert를 일반적인 배열로 처리시에는 $O(n^2)$,
  최소 히프로 처리시에는 O(n log n) 이다.

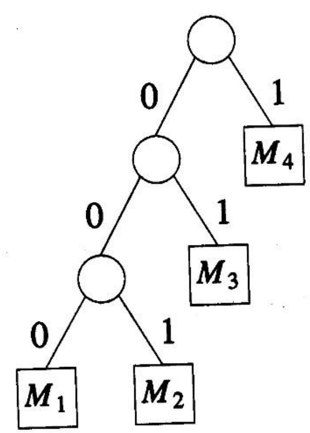
예제

- 리스트의 길이:
  (2, 3, 5, 7, 9, 13)



Huffman 코드

- 최소 가중 외부 경로 길이를 갖는 이진 트리를 메시지 M_1,M_2, …, M_n 에 대한 최적의 코드 집합을 얻는데 사용할 수 있다.
- 코드(code)는 대응된 메시지의 전달을 위해 사용되는 이진 문자열(binary string)이다. 메시지를 수신한 곳에서는 복호 트리(decode tree)를 사용하여 코드를 해독한다.
- 복호 트리는 외부 노드들이 메시지를 표현하는 이진 트리이다.



M_1: 000 (a)

M_2: 001 (b)

M_3: 01 (c)

M_4: 1 (d)

100101110001 ←→ dbcddad

- 코드 단어를 복호하는 비용은 그 코드에 포함된 비트들의 수에 비례한다. 비트들의 수는 루트 노드에서 대응되는 외부 노드까지의 거리와 같다.
- q_i 를 메시지 M_i 가 전송되는 상대적인 빈도라고 하면,
  평균 복호 시간(expected decode time)은
  $\Sigma_{1 \le i \le n} q_i d_i$, d_i 는 루트 노드에서 메시지 M_i 의 외부 노드 까지의 거리
- 평균 복호 시간은 최소 가중 외부 경로 길이를 갖는 복호 트리에 의해 얻어지는 코드 단어들을 선택함으로써 최소화할 수 있다.
- 평균 복호 시간을 최소화하는 코드는 메시지의 평균 길이도 최소화한다.

Huffman 트리의 구성

- 암호(encode) 트리: 복호 트리와 동일
- 예: 가중 규칙(사용 빈도수)가 2, 3, 5, 7, 9, 13 일 때

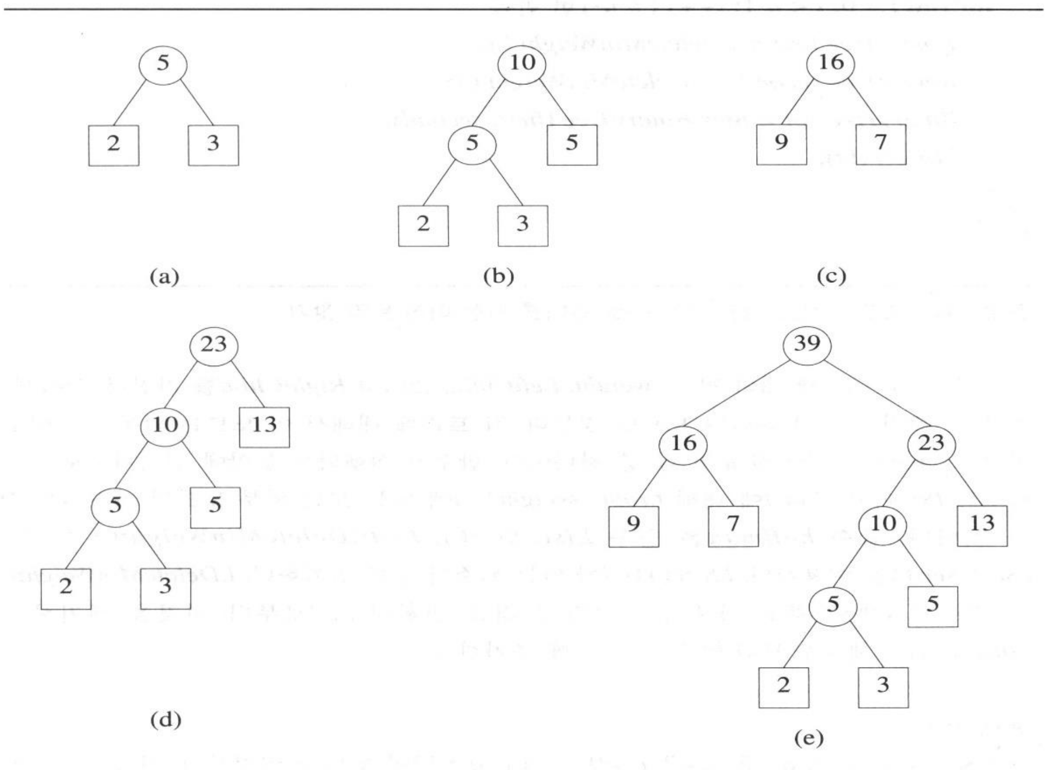


그림 7.28 : Huffman 트리의 구성

4.6 배낭 문제(Knapsack Problem)

- 문제: n 개의 물건들과 1 개의 배낭이 주어지고, 물건 i 는 무게 w_i 를 가지며, 물건 i 의 일부분 x_i ($0 \le x_i \le 1$) 를 배낭에 넣는다면 $p_i \cdot x_i$ 만큼의 이익을 얻는다. 배낭의 용량이 m 일때, 최대 이익을 취할 수 있도록 물건들을 배낭에 채워라.
- 다음과 같이 기술할 수 있다.

  maximize $\Sigma_{1 \le i \le n}\, p_i \cdot x_i$, (4.1)

  subjet to $\Sigma_{1 \le i \le n}\, w_i \cdot x_i \le m$ and (4.2)

  $0 \le x_i \le 1,\ 1 \le i \le n$ (4.3)
- 무게와 이익은 모두 양수 값이다.
- 가능 해는 위의 (4.2)과 (4.3)을 만족하는 집합 $(x_1, x_2, \ldots, x_n)$이다.
- 최적 해는 (4.1)의 식을 최대로 하는 집합 $(x_1, x_2, \ldots, x_n)$이다.

배낭 문제의 예

- $n = 3$, $m = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$,
  $(w_1, w_2, w_3) = (18, 15, 10)$
  이 문제에 대한 네 가지 가능 해들은 다음과 같다.

	(x_1, x_2, x_3)	$\Sigma w_i \cdot x_i$	$\Sigma p_i \cdot x_i$
1.	(1/2, 1/3, 1/4)	16.5	24.5
2.	(1, 2/15, 0)	20	28.2
3.	(0, 2/3, 1)	20	31
4.	(0, 1, 1/2)	20	31.5

위 가능 해들 중에 4번 해가 최대의 이익을 만들어 내고 있으며, 실제 최적 해가 된다.

배낭 문제의 성질

- 보조 정리:
  모든 무게들의 합이 ≤ m 인 경우에, $x_i = 1$, $1 \le i \le n$ 이 최적 해이다.
- 보조 정리:
  모든 최적 해는 배낭을 정확히 꽉 채운다.
- 배낭 문제에서는 물건들의 부분 집합을 선택하는 것이며(부분 집합 형), 각 물건에 대한 x_i 의 선택을 해야한다.
  특히 $x_i = 0$은 해당 물건을 선택하지 않음을 뜻한다.

욕심쟁이 전략

- 전략 1:

  가장 큰 이익을 갖는 물건을 우선 포함시킨다. 고려중인 물건이 꼭 맞지 않는다면, 이 물건의 일부분으로 배낭을 채운다.

  예: 예제 4.1에서

  2번째 해: (1, 2/15, 0) 20 28.2 → 최적 해 아님

- 전략 2:

  배낭의 용량을 가능한 한 천천히 채우도록 한다.

  3번째 해: (0, 2/3, 1) 20 31 → 최적 해 아님

- 전략 3:

  이익이 증가하는 비율과 용량이 채워지는 비율 사이에 균형을 맞춘다. 단위 용량 당 최대 이익을 갖는 물건을 우선 포함시킨다.

  즉 p_i/w_i 값이 큰 것을 먼저 고려한다.

  4번째 해: (0, 1, 1/2) 20 31.5 → 최적 해

배낭문제에 대한 최적의 욕심쟁이 방법 (lec7-1)

- $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$ 의 순서로 입력된다.

```
public void GreedyKnapsack(float m, int n)
  // p[1:n] and w[1:n] contain the profits and weights
  // respectively of the n objects ordered such that
  // p[i]/w[i] >= p[i+1]/w[i+1]. m is the knapsack
  // size and x[1:n] is the solution vector.
  {
     for (int i=1; i<=n; i++) x[i] = 0.0; // Initialize x.
     float U = m;
     for (i=1; i<=n; i++) {
        if (w[i] > U) break;
        x[i] = 1.0;
        U -= w[i];
     }
     if (i <= n) x[i] = U/w[i];
  }
```

- 두 개의 for 문이 각각 n 번 반복하므로 O(n) 시간 걸리며, 단위 용량당 이익의 크기 순서를 얻기 위해 정렬하는 시간까지 포함하면 O(n log n) 시간이 걸린다.

- 정리:
  $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$ 이면, 프로그램 GreedyKnapsack은 배낭 문제의 주어진 사례에 대한 최적 해를 생성한다.
- 0/1 배낭 문제:
  maximize $\Sigma_{1 \leq i \leq n}\ p_i \cdot x_i$,
  subjet to $\Sigma_{1 \leq i \leq n}\ w_i \cdot x_i \leq m$ and
  $x_i = 0$ or 1, $1 \leq i \leq n$

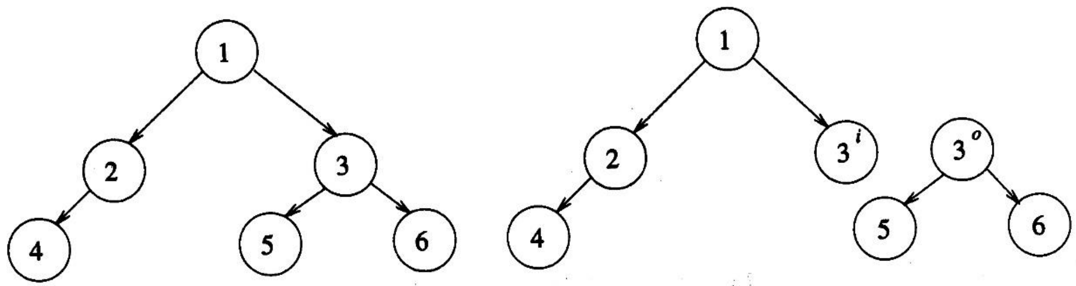
이 문제에서는 앞의 프로그램이 최적의 해를 구하지 못한다.

예: 앞의 예제에서

프로그램을 적용하면 (0, 1, 0) 과 24의 이익을 얻을 수 있으나
최적 해는 (1, 0, 0)으로 25의 이익을 얻을 수 있다.

4.7 트리 정점 분할(Tree Vertex Splitting Problem: TVSP)

- 간선에 실수의 가중치를 갖는 가중 방향 트리(weighted directed tree)에서 (그래프의 특별한 형태로 취급), 특정한 정점에서 트리를 분할한다.
- 예:



- 전기선의 전압 손실이나 송유관의 유압 손실을 어떤 한계(한계값) 이상으로 감당할 수 없을 때 중간 지점에서 승압기를 설치해야 한다. 가능한 한 적은 수의 승압기를 설치하여 이 문제를 해결하고자 할 때 최적의 위치를 결정하기 위해 이 문제의 알고리즘을 이용할 수 있다.
- 용어: 시작 정점, 도착 정점, 한계값(δ 로 표기)
- 임의 경로 P에 대해 그 경로의 지연(delay)은, d(P)로 표기, 그 경로 상의 가중치들의 합으로 정의된다.
- 트리 T의 지연, d(T)는 모든 경로 지연들 중의 최대 값이다.

- T = (V, E, w)는 간선상에 가중치 w를 갖는 방향 트리이다.
- T/X 는 X에 속한 각 정점 u를 두 노드 u^i와 u^o로 분할하고 모든 간선 $<u, j> \in E$ ($<j, u> \in E$)를 $<u^o, j>$ ($<j, u^i>$)로 대치함으로써 만들어지는 포리스트(forest)로 정의한다.

  예: 앞 그림에서 X = { 3 }일 때 분할된 트리는 T/X를 나타낸다.
- TVSP는 한계값 δ 에 대하여 d(T/X) ≤ δ 를 만족하는 가장 작은 크기의 집합 X ⊆ V 를 구하는 것이다.
- 최대 간선 가중치가 δ 이하일 때 TVSP 문제가 해를 가지며, 이 문제는 자연스럽게 부분 집합형에 해당한다.

TVSP 해결 방법

- 간단한 해결 방법:

  가중 트리 T = (V, E, w)와 한계값 δ 가 주어졌을 때 $d(T/X) \le \delta$ 을 만족하는 V의 부분 집합 X 는 가능 해이다. 주어진 X에 대해 d(T/X)를 O(|V|) 시간에 계산할 수 있다. 따라서 V의 모든 가능한 부분 집합 X에 대해 d(T/X)를 계산한 후, 최소의 원소를 갖는 X를 결정할 수 있다.

  → $2^{|V|}$개의 부분 집합이 존재하므로 엄청난 시간이 필요

- 욕심쟁이 방법:

  각 노드 $u \in V$에 대하여 u 로부터 u 의 부분 트리(subtree)에 속한 다른 임의의 노드까지의 최대 지연 d(u)를 계산한다.

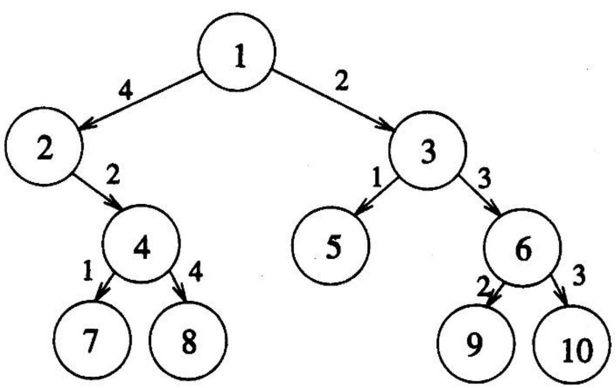
  $d(u) = \max_{k \in C(u)}\{d(k) + w(u, k)\}$,

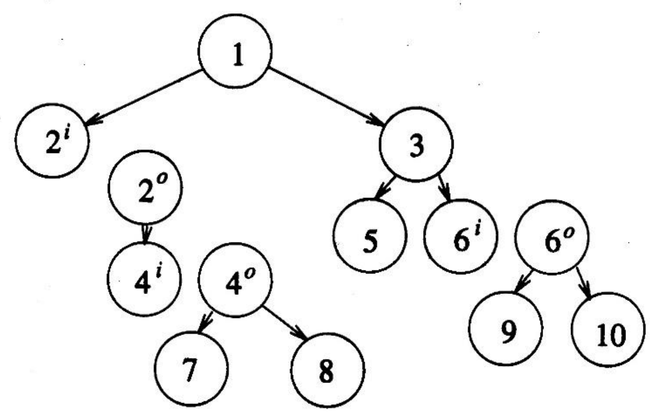
  C(u)는 u의 자식 노드들의 집합, u 가 리프 노드이면 d(u) = 0

  만약 u가 $d(u)+w(v,u) > \delta$ 가 되는 부모 노드 v 를 가진다면, 노드 u를 분할하고, d(u)를 0 으로 놓는다. 계산은 리프 노드부터 루트 노드로 진행한다.

욕심쟁이 방법의 예

- δ = 5 일때 다음 그림에서 d(7) = d(8) = d(5) = d(9) = 0.
  d(4) = 4 이고 d(4)+w(2,4) = 6 > δ 이므로 노드 4는 분할되고 d(4) = 0.
  d(2) = 2 이고 d(2)+w(1,2) = 6 > δ 이므로 노드 2는 분할되고 d(2) = 0.
  d(6) = 3 이고 d(6)+w(3,6) = 6 > δ 이므로 노드 6은 분할되고 d(6) = 0.
  d(3) = 3, d(1) = 5.
  따라서 최적의 위치를 나타내는 X = {4, 2, 6} 이다.





트리 정점 분할 알고리즘(lec7-2)

```
void TVS(Tree T, int delta)
// Determine and output the nodes to be split.
// w is the weighting adjacent matrix for the edges.
{
   if ( T != NIL ) {
        d[T] = 0 ;
        for (each child v of T) {
             TVS(v, delta);
             d[T] = max{ d[T], d[v]+w[T][v] };
        }
        if ((T is not the root) && (d[T]+w[parent(T)][T)>delta)) {
             System.out.println(T);
             d[T] = 0 ;
        }
    }
}
```

- 프로그램 TVS는 트리의 노드를 후위 순서(postorder)로 방문한다.
- TVS는 순환 호출로 호출되며 각 노드 T에서 단지 한번만 호출되고, d의 계산과 분할 결정은 모두 상수 시간 걸리므로 O(n) 시간 복잡도를 갖는다.
- 정리:

  알고리즘 TVS는 임의의 트리 T에서 어떠한 간선도 δ 보다 큰 가중치를 갖지 않는다면 d(T/U) ≤ δ 를 만족하는 최소 집합 U를 출력한다.

레포트 #3

1. 하나의 영문 파일을 입력받아 각 영문자마다(space 문자 포함) 빈도수를 조사한 후, Huffman 트리(decode 트리)를 생성하라.
2. Huffman 트리를 이용하여 각 문자의 코드를 결정하라.

<제출물>

- 프로그램 소스
- 입력 파일 (위 단계 1에서 사용됨)
- 각 문자의 빈도수와 할당된 코드(단계 1과 2의 결과)

** 3개의 파일에 대해 테스트하라.

** 단순화를 위해 영문 파일은 영어 알파벳의 소문자와 space 문자로만 되어 있다고 가정한다.