알고리즘

- HW10: Optimal Binary Search Tree-

제 출 일	2017년 12월 05일
분 반	02반
담당교수	공은배
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 해결 방법

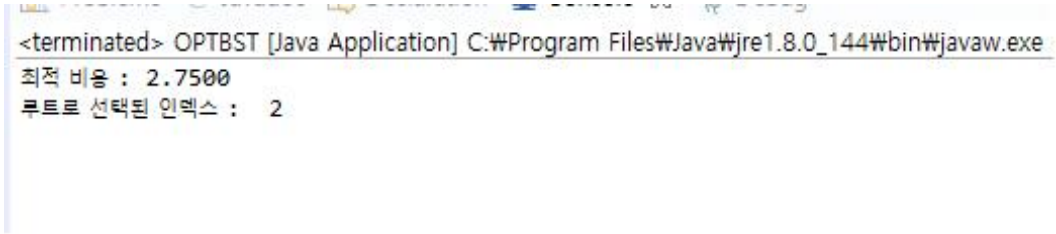
```
double[][] Ei = new double[count+1][count];
double[][] Wi = new double[count+1][count];
int[][] root = new int[count+1][count];
for(int i = 1;i<count+1;i++) {
    Ei[i][i-1]=Qi[i-1];
    Wi[i][i-1]=Qi[i-1];
}//필요한 배열선언후 초기화
int j;
for(int l=1;l<count;l++) {
    for(int i =1;i<count-l+1;i++) {
        j=i+l-1;
        Ei[i][j]=M;//초기화
        Wi[i][j]=Wi[i][j-1]+Pi[j]+Qi[j];//이전에 구한값들을 이용해서 W를 구한다
        for(int r=i;r<=j;r++) {
            double t = Ei[i][r-1]+Ei[r+1][j]+Wi[i][j];//새롭게구한 i~j에서 r을 선택할경우 평균비용이다
            if (t<Ei[i][j]) {//만약 더적을경우 최적값을 찾았기때문에 갱신한다.
                Ei[i][j]=t;
                root[i][j]=r;
            }
        }
    }

}

System.out.printf("최적 비용 : %.4f\n루트로 선택된 인덱스 :  %d",Ei[1][count-1],root[1][count-1]);
```

데이터를 읽은면서 Pi와 Qi에 알맞은 확률을 저장한뒤 필요한 Ei배열과 Wi배열을 만든 후에 초기화를 해준다. 그후 I와 J를 증가시키면서 테이블을 완성하는데 테이블을 완성시킬 때 최단 비용을 성립하는 루트값을 저장하기위해 Root배열을 만들어놓고 반복문을 실행하면 OPT테이블이 완성된다.

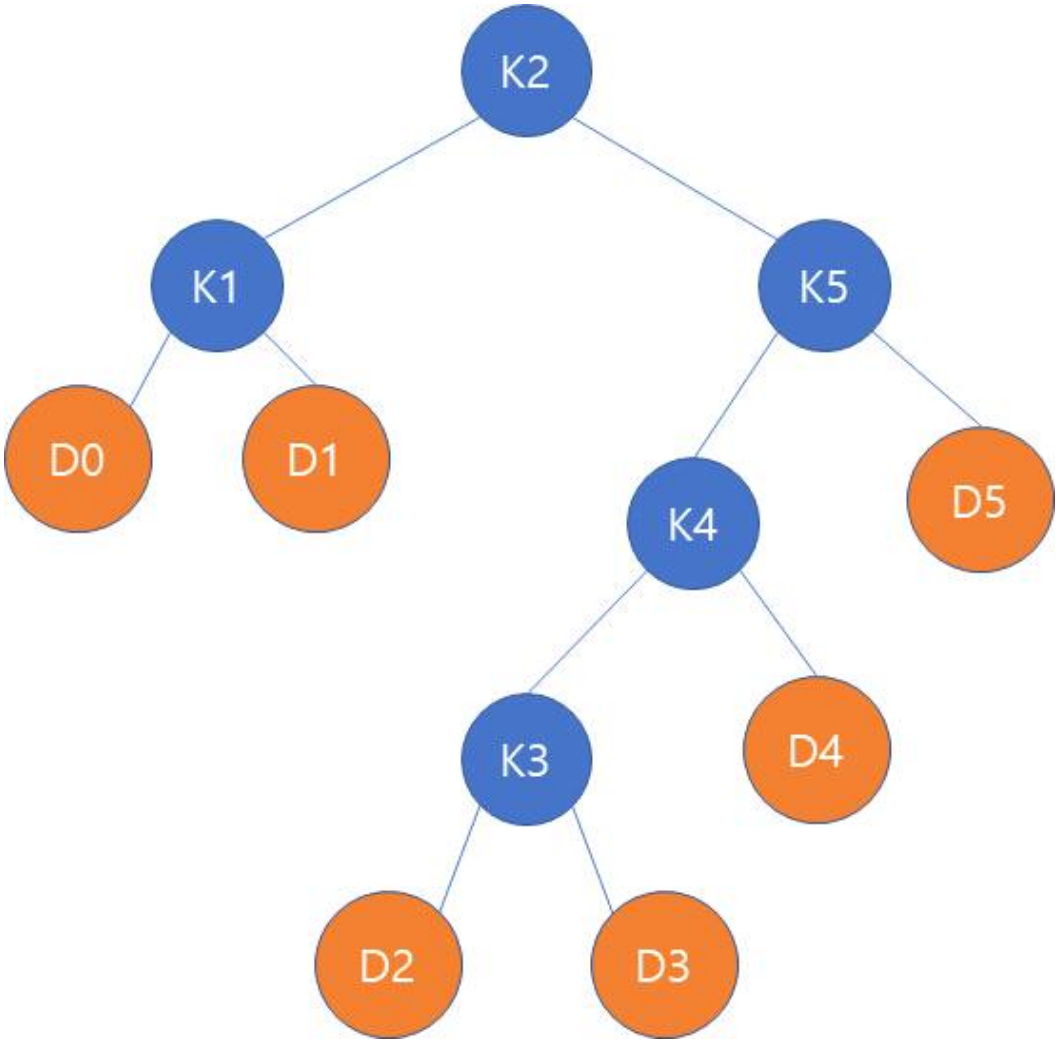
2.실행결과



3. Path

일단 최상단 Root는 1~5의 루트값ㅇ르 저장하고있는 Root[1][5]의값이다
그값은 2로서 최상단 루트는 k2를 선택 왼쪽 child는 자동적으로 k1 선택한다.
오른쪽 child는 k345로 이루어진 것중 최소비용을 구하면되는데 Root[3][5]를 보면 5를 저장하고 있다.
k5를 중심으로 하고 오른쪽은 올수있는건 d5밖에없다 왼쪽에 올 수 있는건 k3,4인데
root[3][4]의 값은 4임으로 k4를 선택 하면 트리가 완성된다.

root	(id=24)
[0]	(id=28)
[1]	(id=29)
[2]	(id=30)
[3]	(id=31)
[0]	0
[1]	0
[2]	0
[3]	3
[4]	4
[5]	5
[4]	(id=32)
[5]	(id=33)
[6]	(id=34)



4. 느낀 점

저번보다 수도코드가 자세히 나와 있어서 쉽게 만들었다. 알고리즘을 이용하여 빨리 풀 수 있는 과제인데 만약 손으로 푼다면 엄청 오래걸릴 것 같다.