Design and Analysis of Algorithm

Lecture-18:
Dynamic Programming

Contents

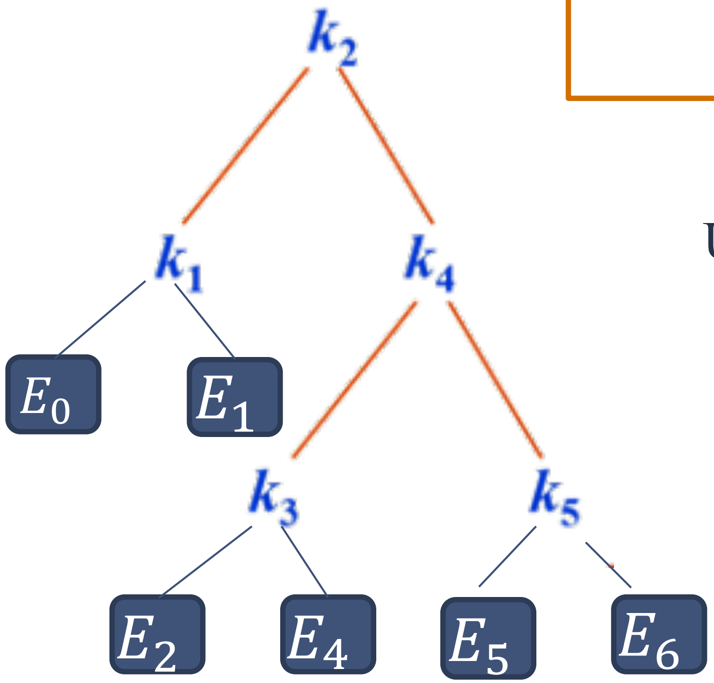
Search Case

- The case of search are two situations, one is success, and the other, without saying, is failure.

If a successful search terminates at an internal node at level L

then, the expected cost contribution from the internal node for a_j is

$p(a_j) \ast level(a_j)$



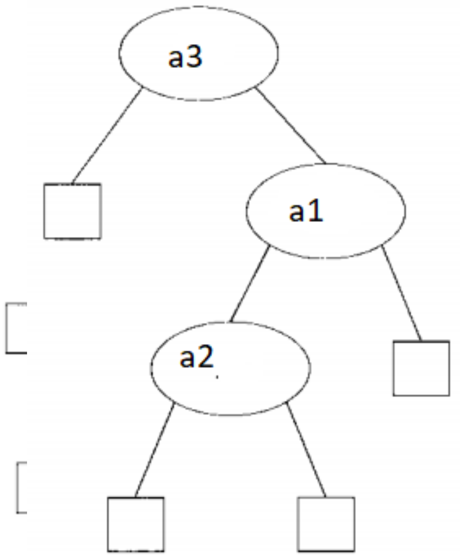
Unsuccessful searches terminate at external nodes

the cost contribution of the external node is $q(i) \ast (level(Ei) - 1)$

Example

- The possible binary search trees for the keys (a_1, a_2, a_3) are given with following probabilities:

With p(l)= .05,p(2) = .1,p(3) = .05, q(0)= .15,q(l) = .1,q(2)= .05and q(3)= .05



(a) $3\times0.5 + 2\times0.1 + 1\times.05 + 3\times0.15 + 3\times0.1 + 2\times.05 + 1\times.05 = 2.65$

(b) 1.9

(c) 1.6

```
Algorithm OBST(p, q, n)
// Given n distinct identifiers a_1 < a_2 < ··· < a_n and probabilities
// p[i], 1 ≤ i ≤ n, and q[i], 0 ≤ i ≤ n, this algorithm computes
// the cost c[i, j] of optimal binary search trees t_ij for identifiers
// a_{i+1}, ..., a_j. It also computes r[i, j], the root of t_ij.
// w[i, j] is the weight of t_ij.
{
    for i := 0 to n − 1 do
    {
        // Initialize.
        w[i, i] := q[i]; r[i, i] := 0; c[i, i] := 0.0;
        // Optimal trees with one node
        w[i, i + 1] := q[i] + q[i + 1] + p[i + 1];
        r[i, i + 1] := i + 1;
        c[i, i + 1] := q[i] + q[i + 1] + p[i + 1];
    }
    w[n, n] := q[n]; r[n, n] := 0; c[n, n] := 0.0;
    for m := 2 to n do  // Find optimal trees with m nodes.
        for i := 0 to n − m do
        {
            j := i + m;
            w[i, j] := w[i, j − 1] + p[j] + q[j];
            // Solve 5.12 using Knuth's result.
            k := Find(c, r, i, j);
                // A value of l in the range r[i, j − 1] ≤ l
                // ≤ r[i + 1, j] that minimizes c[i, l − 1] + c[l, j];
            c[i, j] := w[i, j] + c[i, k − 1] + c[k, j];
            r[i, j] := k;
        }
    write (c[0, n], w[0, n], r[0, n]);
}
```

```
Algorithm Find(c, r, i, j)
{
    min := ∞;
    for m := r[i, j − 1] to r[i + 1, j] do
        if (c[i, m − 1] + c[m, j]) < min then
        {
            min := c[i, m − 1] + c[m, j]; l := m;
        }
    return l;
}
```

```
Algorithm OBST(p, q, n)
// Given n distinct identifiers a_1 < a_2 < ··· < a_n and probabilities
// p[i], 1 ≤ i ≤ n, and q[i], 0 ≤ i ≤ n, this algorithm computes
// the cost c[i, j] of optimal binary search trees t_ij for identifiers
// a_{i+1}, ..., a_j. It also computes r[i, j], the root of t_ij.
// w[i, j] is the weight of t_ij.
{
    for i := 0 to n − 1 do
    {
        // Initialize.
        w[i, i] := q[i]; r[i, i] := 0; c[i, i] := 0.0;
        // Optimal trees with one node
        w[i, i + 1] := q[i] + q[i + 1] + p[i + 1];
        r[i, i + 1] := i + 1;
        c[i, i + 1] := q[i] + q[i + 1] + p[i + 1];
    }
    w[n, n] := q[n]; r[n, n] := 0; c[n, n] := 0.0;
    for m := 2 to n do  // Find optimal trees with m nodes.
        for i := 0 to n − m do
        {
            j := i + m;
            w[i, j] := w[i, j − 1] + p[j] + q[j];
            // Solve 5.12 using Knuth's result.
            k := Find(c, r, i, j);
                // A value of l in the range r[i, j − 1] ≤ l
                // ≤ r[i + 1, j] that minimizes c[i, l − 1] + c[l, j];
            c[i, j] := w[i, j] + c[i, k − 1] + c[k, j];
            r[i, j] := k;
        }
    write (c[0, n], w[0, n], r[0, n]);
}
```

Let n = 4 keys be (a_1, a_2, a_3, a_4)

p(1:4) = (3,3,1,1)

q(0 :4) = (2,3,1,1,1)

$c_{02} = 12 + 0 + 7 = 19$

	0	1	2	3	4
0	$w_{00} = 2$ $c_{00} = 0$ $r_{00} = 0$	$w_{11} = 3$ $c_{11} = 0$ $r_{11} = 0$	$w_{22} = 1$ $c_{22} = 0$ $r_{22} = 0$	$w_{33} = 1$ $c_{33} = 0$ $r_{33} = 0$	$w_{44} = 1$ $c_{44} = 0$ $r_{44} = 0$
1	$w_{01} = 8$ $c_{01} = 8$ $r_{01} = 1$	$w_{12} = 7$ $c_{12} = 7$ $r_{12} = 2$	$w_{23} = 3$ $c_{23} = 3$ $r_{23} = 3$	$w_{34} = 3$ $c_{34} = 3$ $r_{34} = 4$	
2	$w_{02} = 12$ $c_{02} = 19$ $r_{02} = 1$	$w_{13} = 9$ $c_{13} = 12$ $r_{13} = 2$	$w_{24} = 5$ $c_{24} = 8$ $r_{24} = 3$		
3	$w_{03} = 14$ $c_{03} = 25$ $r_{03} = 2$	$w_{14} = 11$ $c_{14} = 19$ $r_{14} = 2$			
4	$w_{04} = 16$ $c_{04} = 32$ $r_{04} = 2$				

$w_{02} = 8 + 3 + 1 = 12$

$m = 1\ to\ 2$ / $if\ m = 1$

$min = c_{00} + c_{12} = 7 \quad l = 1$

$if\ \ m = 2$

$c_{01} + c_{22} = 8 > min,$

```
Algorithm Find(c, r, i, j)
{
    min := ∞;
    for m := r[i, j − 1] to r[i + 1, j] do
        if (c[i, m − 1] + c[m, j]) < min then
        {
            min := c[i, m − 1] + c[m, j]; l := m;
        }
    return l;
}
```

Question

Given a binary search tree, check whether it is optimal binary search tree or not.

```
    w[n,n] := q[n]; r[n,n] := 0; c[n,n] := 0.0;
    for m := 2 to n do  // Find optimal trees with m nodes.
        for i := 0 to n − m do
        {
            j := i + m;
            w[i,j] := w[i,j − 1] + p[j] + q[j];
            // Solve 5.12 using Knuth's result.
            k := Find(c, r, i, j);
                // A value of l in the range r[i,j − 1] ≤ l
                // ≤ r[i + 1,j] that minimizes c[i,l − 1] + c[l,j];
            c[i,j] := w[i,j] + c[i,k − 1] + c[k,j];
            r[i,j] := k;
        }
    write (c[0,n], w[0,n], r[0,n]);
}
```

k_1	k_2	k_3	k_4	k_5	d_0	d_1	d_2	d_3	d_4	d_5
0.15	0.1	0.05	0.1	0.2	0.05	0.1	0.05	0.05	0.05	0.1

	0	1	2	3	4	5
0	$w_{oo} = .05$ $c_{00} = 0$ $r_{00} = 0$	$w_{11} = .1$ $c_{11} = 0$ $r_{11} = 0$	$w_{22} = .05$ $c_{22} = 0$ $r_{22} = 0$	$w_{33} = .05$ $c_{33} = 0$ $r_{33} = 0$	$w_{44} = .05$ $c_{44} = 0$ $r_{44} = 0$	$w_{55} = .1$ $c_{55} = 0$ $r_{55} = 0$
1	$w_{o1} = 0.3$ $c_{01} = 0.3$ $r_{01} = 1$	$w_{12} = .25$ $c_{12} = .25$ $r_{12} = 2$	$w_{23} = .15$ $c_{23} = 0.15$ $r_{23} = 3$	$w_{34} = .2$ $c_{34} = 0.2$ $r_{34} = 4$	$w_{45} = .35$ $c_{45} = .35$ $r_{45} = 5$	
2	$w_{o2} = 0.45$ $c_{02} = 0.7$ $r_{02} = 1$	$w_{13} = .35$ $c_{13} = .5$ $r_{13} = 2$	$w_{24} = .30$ $c_{24} = .45$ $r_{24} = 4$	$w_{35} = .5$ $c_{35} = .7$ $r_{35} = 5$		
3	$w_{03} = .55$ $c_{03} = 1.0$ $r_{03} = 2$	$w_{14} = .50$ $c_{14} = .95$ $r_{14} = 2$	$w_{25} = .60$ $c_{25} = .9$ $r_{25} = 5$			
4	$w_{04} = .7$ $c_{04} = 1.45$ $r_{04} = 2$	$w_{15} = .8$ $c_{15} = 1.65$ $r_{15} = 4$				
5	$w_{05} = 1.0$ $c_{05} = 2.2$ $r_{05} = 2$					

```
Algorithm Find(c, r, i, j)
{
    min := ∞;
    for m := r[i, j − 1] to r[i + 1, j] do
        if (c[i, m − 1] + c[m, j]) < min then
        {
            min := c[i, m − 1] + c[m, j]; l := m;
        }
    return l;
}
```