Optimal Binary Search Trees

```
    k2
   /  \
 k1    k5
       /
     k4
     /
   k3
```

i	$\text{depth}_T(k_i)$	$\text{depth}_T(k_i) \cdot p_i$
1	1	0.25
1	0	0
3	2	0.1
4	1	0.2
5	2	0.3
		1.10

$$\text{Therefore, } E[\text{search cost}] = 2.10$$

Observerations

- Optimal BST may not have smallest height
- Optimal BST may not have highest-probability key at root

For each, assign keys and compute expected search cost. But there are $\Omega(\frac{4^n}{3^{3/2}})$ different BSTs with n nodes.

Optimal Substructure

- Any subtree of a BST contains key in a contiguous range $k_i, \ldots, k_j$ for some $1 \leq i \leq j \leq n$.
- If T is an optimal BST and T contains subtree T' with keys $k_i, \ldots, k_j$, then T' must be an optimal BST for keys $k_i, \ldots, k_j$.

To find an optimal BST:

1. Examine all candidate roots k_r, for $i \leq r \leq j$
2. Determine all optimal BSTs containing $k_i, \ldots, k_{r-1}$ and containing $k_{r+1}, \ldots, k_j$

When optimal subtree becomes a subtree of a node:

1. Depth of every node in OPT subtree goes up by 1
2. Expected search cost increases by

$$w(i, j) = \sum_{l=i}^{j} p_l$$

If k_r is the root of an optimal BST for $k_i, \ldots, k_j$:

$$\begin{aligned} e[i,j] &= p_r + (e[i,r-1] + w(i,r-1)) + (e[r+1,j] + w(r+1,j)) \\ &= e[i,r-1] + e[r+1,j] + w(i,j) \\ &\qquad \big(\text{because } w(i,j) = w(i,r-1) + p_r + w(r+1,j)\big) \end{aligned}$$

But, we don't know k_r. Hence,

$$e[i,j] = \begin{cases} 0 & \text{if } j = i-1 \\ \underbrace{\min}_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \le j \end{cases}$$