

# MA/CSSE 473

## Day 33

Optimal BSTs



# Exam scores

- This Exam

|     |
|-----|
| 100 |
| 99  |
| 96  |
| 95  |
| 93  |
| 92  |
| 91  |
| 90  |
| 88  |
| 86  |
| 82  |
| 78  |
| 73  |
| 70  |
| 69  |
| 69  |
| 69  |

Exam Average

|      |
|------|
| 99.5 |
| 97   |
| 90.5 |
| 88   |
| 87.5 |
| 86   |
| 84   |
| 84   |
| 81   |
| 78   |
| 77   |
| 75.5 |
| 73.5 |
| 71   |
| 69.5 |
| 67.5 |
| 62.5 |



# MA/CSSE 473 Day 33

- HW 13 due Tomorrow
- Don't forget the Boyer-Moore demonstration animation, due Nov 20
- **Student Questions**
- Optimal BSTs
- Greedy algorithms



# Recap: Optimal Binary Search Trees

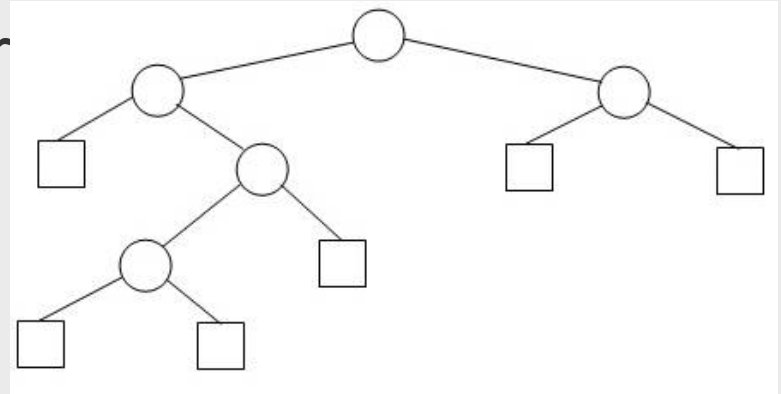
- Suppose we have  $n$  distinct data items  $x_1, x_2, \dots, x_n$  (arranged into increasing order) that we wish to arrange into a Binary Search Tree.
- This time the expected number of probes for an unsuccessful search depends on the shape of the tree and where the search ends up.
- Let  $y$  be the value we are searching for
- Let  $p_i$  be the probability that  $y$  is item  $x_i$
- For  $i = 1, \dots, n-1$  let  $q_i$  be the probability that  $x_i < y < x_{i+1}$
- Similarly, let  $q_0$  be the probability that  $y < x_1$ , and  $q_n$  the probability that  $y > x_n$
- Note that 
$$\sum_{i=1}^n p_i + \sum_{i=0}^n q_i = 1$$

but we can (and will) also just use frequencies when finding the optimal tree (and divide by their sum to get probabilities).



# Recap: Extended binary search tree

- Formally, an Extended Binary Search Tree (EBT) consists of
  - an external node, or
  - an (internal) root node and two EBTs  $T_L$  and  $T_R$



- In diagram, Circles = internal nodes,  
Squares = external nodes
- It's an alternative way of viewing a binary tree
- The external nodes stand for places where an unsuccessful search can end or where an element can be inserted
- An EBT with  $n$  internal nodes has  $n + 1$  external nodes



# How many possible BST's

- Given distinct items  $x_1 < x_2 < \dots < x_n$ , how many different Binary Search Trees can be constructed?
- Figure it out for  $n=2, 3, 4$
- Write the recurrence relation
- Solution is the **Catalan number**  $c(n)$

$$c(n) = \binom{2n}{n} \frac{1}{n+1} = \frac{(2n)!}{n!(n+1)!} \approx \frac{4^n}{n^{3/2} \sqrt{\pi}}$$

- Verify for  $n = 2, 3, 4, 5$



# What not to measure

- Before, we introduced the notions of external path length and internal path length
- These do not take into account the frequencies.



# What contributes to the expected number of probes?

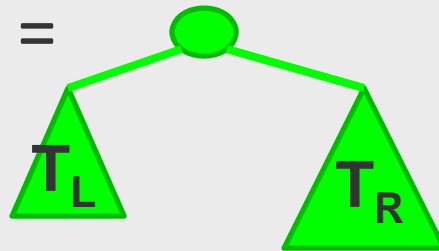
- Frequencies, depth of node
- For successful search, number of probes is one more than depth of the corresponding internal node
- For unsuccessful, number of probes is equal to depth of the corresponding external node



# Weighted Path Length

$$\sum_{i=1}^n p_i [1 + \text{depth}(x_i)] + \sum_{i=0}^n q_i [\text{depth}(y_i)]$$

- If we divide this by  $\sum p_i + \sum q_i$  we get the average search time.
- We can also define it recursively:
- $W(\square) = 0$ . If  $T =$



$W(T) = W(T_L) + W(T_R) + \sum p_i + \sum q_i$ , where the summations are over all  $p_i$  and  $q_i$  for nodes in  $T$

- It can be shown by induction that these two definitions are equivalent (good practice problem).



# Example

- Frequencies of vowel occurrence in English
- : A, E, I, O, U
- p's: 32, 42, 26, 32, 12
- q's: 0, 34, 38, 58, 95, 21
- Draw a couple of trees (with E and I as roots), and see which is best. (sum of p's and q's is 390).



# Strategy

- We want to minimize the weighted path length
- Once we have chosen the root, the left and right subtrees must themselves be optimal EBSTs
- We can build the tree from the bottom up, keeping track of previously-computed values



# Intermediate Quantities

- Cost: Let  $C_{ij}$  (for  $0 \leq i \leq j \leq n$ ) be the cost of an optimal tree (not necessarily unique) over the frequencies  $q_i, p_{i+1}, p_{i+1}, \dots, p_j, q_j$ . Then
- $C_{ii} = 0$ , and 
$$C_{ij} = \min_{i < k \leq j} (C_{i,k-1} + C_{kj}) + \sum_{t=i}^j q_t + \sum_{t=i+1}^j p_t$$
- This is true since the subtrees of an optimal tree must be optimal
- To simplify the computation, we define
- $W_{ii} = q_i$ , and  $W_{ij} = W_{i,j-1} + p_j + q_j$  for  $i < j$ .
- Note that  $W_{ij} = q_i + p_{i+1} + \dots + p_j + q_j$ , and so
- $C_{ii} = 0$ , and 
$$C_{ij} = W_{ij} + \min_{i < k \leq j} (C_{i,k-1} + C_{kj})$$
- Let  $R_{ij}$  be a value of  $k$  that minimizes  $C_{i,k-1} + C_{kj}$  in the above formula



# Code

```
# initialize the main diagonal
for i in range(n + 1):
    R[i][i] = i
    W[i][i] = q[i]
    drawSquare(i, i, W[i][i], C[i][i], R[i][i], win, inden)
# Now populate each of the n upper diagonals:
for d in range(1, n+1): # fill in this diagonal
    # The previous diagonals are already filled in.
    for i in range(n - d + 1):
        j = i + d; # on the dth diagonal, j - i = d
        opt = i + 1 # until we find a better one
        for k in range(i+2, j+1):
            if C[i][k-1]+C[k][j] < C[i][opt-1]+C[opt][j]:
                opt = k
        R[i][j] = opt
        W[i][j] = W[i][j-1] + p[j] + q[j]
        C[i][j] = C[i][opt-1] + C[opt][j] + W[i][j]
        drawSquare(i, j, W[i][j], C[i][j], R[i][j], win, i
```

# Results

|        |         |          |          |          |          |
|--------|---------|----------|----------|----------|----------|
| R00: 0 | R01: 1  | R02: 2   | R03: 2   | R04: 3   | R05: 4   |
| W00: 0 | W01: 66 | W02: 146 | W03: 230 | W04: 357 | W05: 390 |
| C00: 0 | C01: 66 | C02: 212 | C03: 418 | C04: 754 | C05: 936 |
|        | R11: 1  | R12: 2   | R13: 3   | R14: 3   | R15: 4   |
|        | W11: 34 | W12: 114 | W13: 198 | W14: 325 | W15: 358 |
|        | C11: 0  | C12: 114 | C13: 312 | C14: 624 | C15: 798 |
|        |         | R22: 2   | R23: 3   | R24: 4   | R25: 4   |
|        |         | W22: 38  | W23: 122 | W24: 249 | W25: 282 |
|        |         | C22: 0   | C23: 122 | C24: 371 | C25: 532 |
|        |         |          | R33: 3   | R34: 4   | R35: 4   |
|        |         |          | W33: 58  | W34: 185 | W35: 218 |
|        |         |          | C33: 0   | C34: 185 | C35: 346 |
|        |         |          |          | R44: 4   | R45: 5   |
|        |         |          |          | W44: 95  | W45: 128 |
|        |         |          |          | C44: 0   | C45: 128 |
|        |         |          |          |          | R55: 5   |
|        |         |          |          |          | W55: 21  |
|        |         |          |          |          | C55: 0   |

**How to  
construct the  
optimal tree?**

**Analysis of the  
algorithm?**

- Constructed by diagonals, from main diagonal upward
- What is the optimal tree?

