Dynamic Programming

Lecture 8

Timothy Kim
GWU CSCI 6212

Optimal Binary Search Tree

- **Problem**: Given a set of keys and their access frequencies, find the optimal binary tree structure.
- **Input**: Frequencies: $\{p_1, p_2, \ldots, p_n\}$ for items $\{1, 2, \ldots, n\}$
- **Output**:
  - Valid Binary Search Tree with minimal weighted search time:

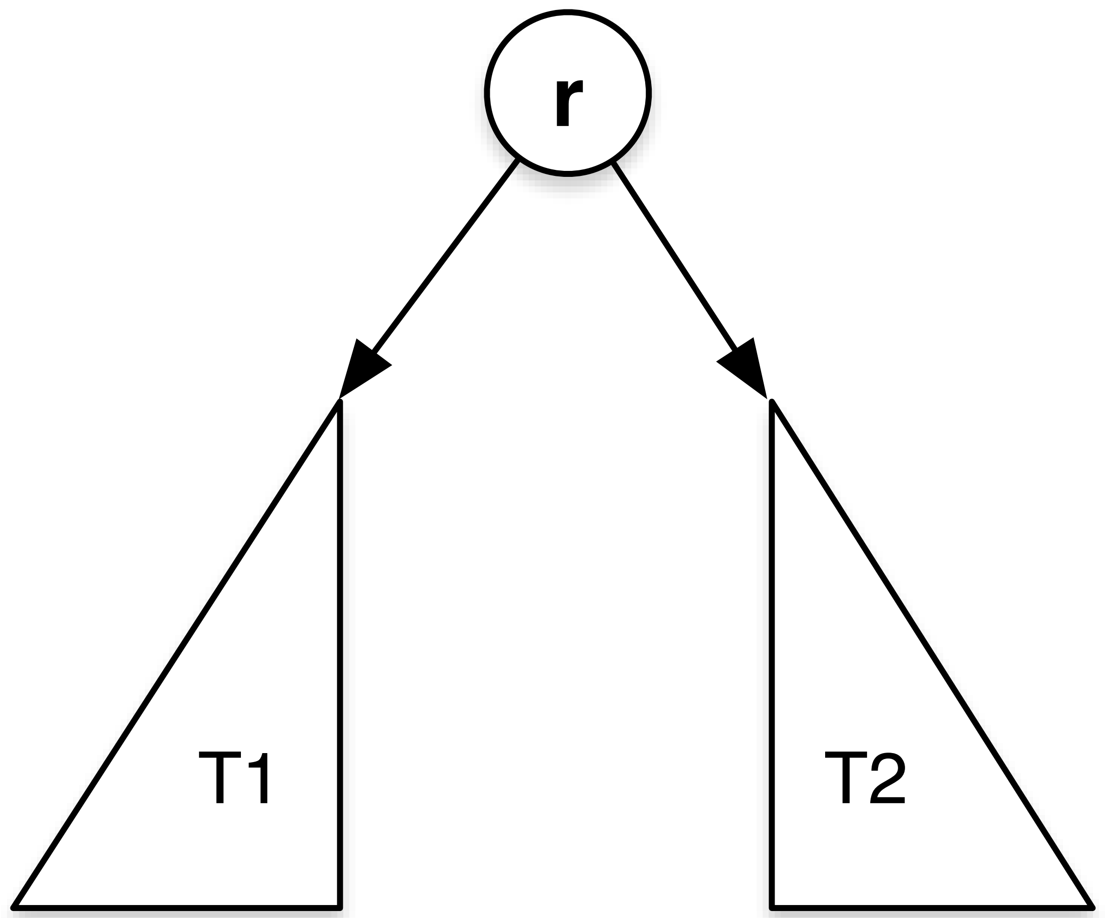
$$C(T) = \sum_{i=\text{items}} p_i \cdot [\text{Search time of } i]$$

  - Search time of i = level of i^{th} item in the tree.

Optimal Substructure

- Assume T is optimal (and thus picked a optimal root)

  - T1 is optimal for $\{1, 2, \ldots, r\text{-}1\}$
  - T2 is optimal for $\{r + 1, r + 2, \ldots, n\}$



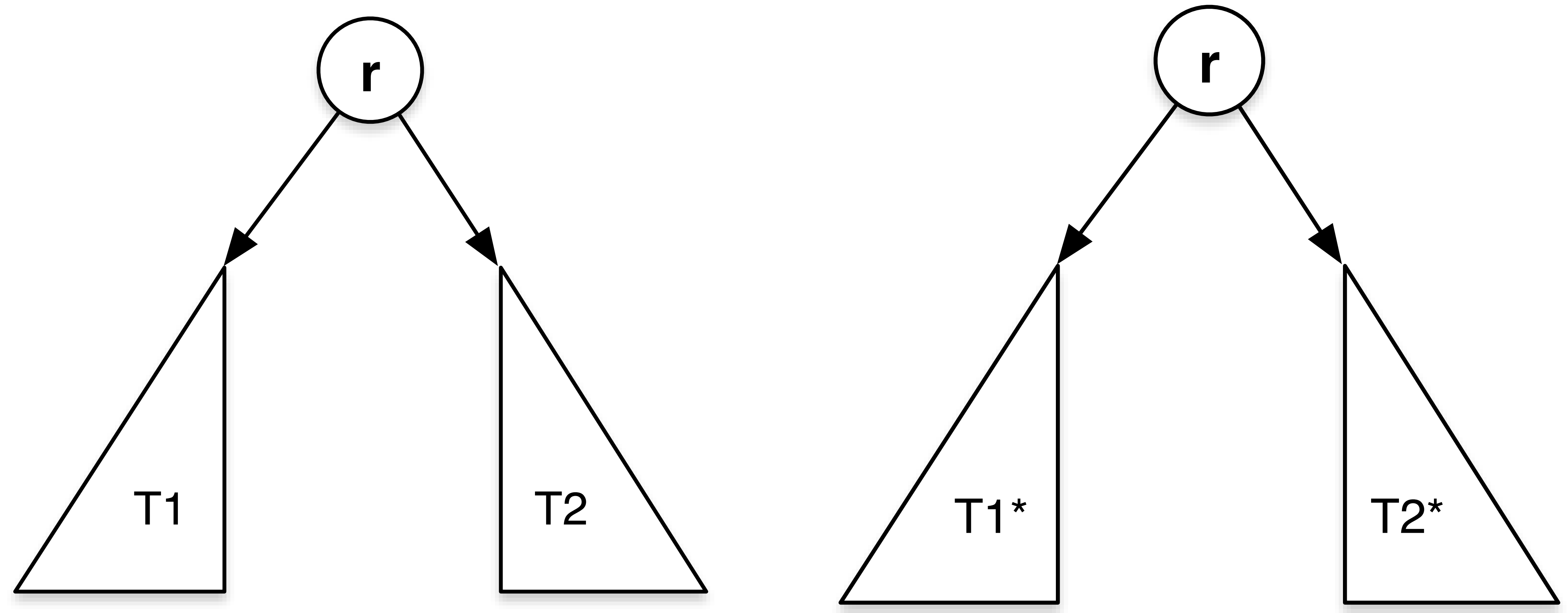
Proof

- Assume that T is optimal yet T_1 is NOT optimal, then

$$\exists T_1^* | C(T_1^*) < C(T_1)$$

- By definition, C of any T is as follows:

$$\begin{aligned}
C(T) &= \sum_{i=1}^{n} p_i \cdot ST_i \\
&= p_r \cdot 1 + \sum_{i=1}^{r-1} p_i \cdot ST_i + \sum_{i=r+1}^{n} p_i \cdot ST_i \\
&= p_r \cdot 1 + \sum_{i=1}^{r-1} p_i \cdot (1 + ST_i \text{ of } T_1) + \sum_{i=r+1}^{n} p_i \cdot (1 + ST_i \text{ of } T_2) \\
&= p_r \cdot 1 + \sum_{i=1}^{r-1} p_i + \sum_{i=1}^{r-1} ST_i \text{ of } T_1 + \sum_{i=r+1}^{n} p_i + \sum_{i=r+1}^{n} ST_i \text{ of } T_2 \\
&= p_r + \sum_{i=1}^{r-1} p_i + \sum_{i=r+1}^{n} p_i + \sum_{i=1}^{r-1} ST_i \text{ of } T_1 + \sum_{i=r+1}^{n} ST_i \text{ of } T_2 \\
&= \sum_{i=1}^{n} p_i + C(T_1) + C(T_2)
\end{aligned}$$



Proof Continued

$$C(T^*) = \sum_{i=1}^{n} p_i + C(T_1^*) + C(T_2^*)$$

- Above statement implies that

$$C(T_1^*) < C(T_1) \rightarrow C(T^*) < C(T)$$

- Which is a contradiction. QED

Subproblem domain

- Naively picking all possible {1, 2, ... i}, {i + 1, ..., n} is not enough.
- Because when you recurse in to the right of the left sub-tree, the start and end index of that tree is not described by the above subproblem.
- Thus
  - S = {i, i + 1, ... , j - 1, j} for all possible i ≤ j values
- Reccurence

$$C_{i,j} = \min_{r=i}^{j}\{\sum_{k-i}^{j} p_k + C_{i,r-1} + C_{r+1,j}\}$$

Algorithm

```
Let A = 2-D Array
for S = 0 to n-1:
  for i = 1 to n:
    j = i + S
    A[i,j] = for r from i to j, min {
      sum of P[k] where k = i to j + A[i, r-1] + A[r+1, j]
    }
return A[i,n]
```

Running Time

- $O(n^2)$: For each sub-problems
- $O(j-i)$: time to compute A[i,j]
- Total running time: $O(n^3)$
- Knuth 71 and Yao 80 solves in $O(n^2)$

Bellman-Ford

- Single source shortest path problem with negative edges.
- Output
    - Either shortest path to all destination from S
    - Or output a negative cycle.

Optimal Substructure

- Idea: restrict on the # of edges that can be used.
- Lemma: Let G=(V,E) be a directed graph with edge length C_e and source vertex S. For ever v in V, i in {1,2,3, ... } let P = shortest path s to v with at most i edges.
  - Case 1: If P has ≤ (i-1) edges it is the shortest path with ≤ i-1 edges for path (s,v)
  - Case 2: If P has i edges with last hop (w,v) then P' is the shortest path with ≤ i - 1 edges for s to w.

Proof of lemma

- Case 1: Proof by contradiction (simple)
- Case 2:
  - Let Q be the path shorter than P'.
  - Then Q + (w,v) is shorter than P because P = P' + (w,v)
  - Which is a contradiction, thus P' must be the shortest.

Recurrence

- $C_{i,v}$ = minimum length of a s to v path with ≤ i edges
- $L_{i,v} = \min \{ L_{i-1,v} , \min \{ L_{i-1,w} + C_{w,v} \}$ for all edges, (w,v) }
- How many times do we iterate?
- If there are no negative cycles, then
    - shortest path will not have cycles
    - you only require ≤ n - 1 edges
- Subproblem domain: $i = \{1, 2, ... , n - 1\}$

Algorithm

```
A = 2-D Array  (i,v)
A[0,s] = 0
A[0,v] = infinity for all v ≠ s
for i = 1 to n-1:
  for each v in V:
    A[i,v] = min { A[i-1, v],
                   min { A[i-1,w] + C[w,v] }
                     for all (w,v) in E
                 }
return all A[n-1,v]
```

Running Time

- $O(n * \text{sum of in-degree of v}) = O(nm)$

Negative Cycle Detection

- Run the for loop one more time
- G has no negative cycle if and only if $A[n\text{-}1,v] = A[n,v]$ for all v in V.