Optimal Binary Search Tree

Input: $F[1..n]$

Goal: Build BST_T to minimize $\sum_{j=1}^{n} f[j] \cdot cost(T, j)$

OptCost(i,k) = total cost of all searches in optimal BST for keys i..k.

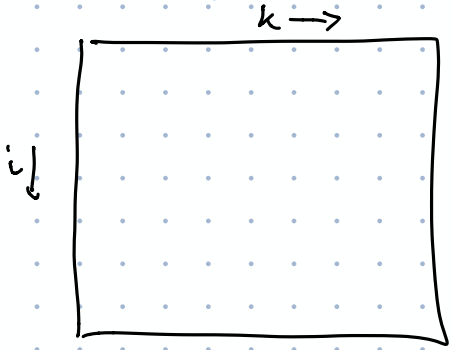
$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ \sum_{j=i}^{k} f[j] + \min_{i \le r \le k} \{ OptCost(i, r-1) + OptCost(r+1, k) \} & \text{otherwise} \end{cases}$$

left

right

root

times we touch the root of opt BST(i..k)



$F[1..n, 1..n]$ $\quad F[i,k] = \sum_{j=i}^{k} f[j]$

```
INITF(f[1..n]):
  for i ← 1 to n
      F[i, i−1] ← 0
      for k ← i to n
          F[i,k] ← F[i,k−1] + f[k]
```

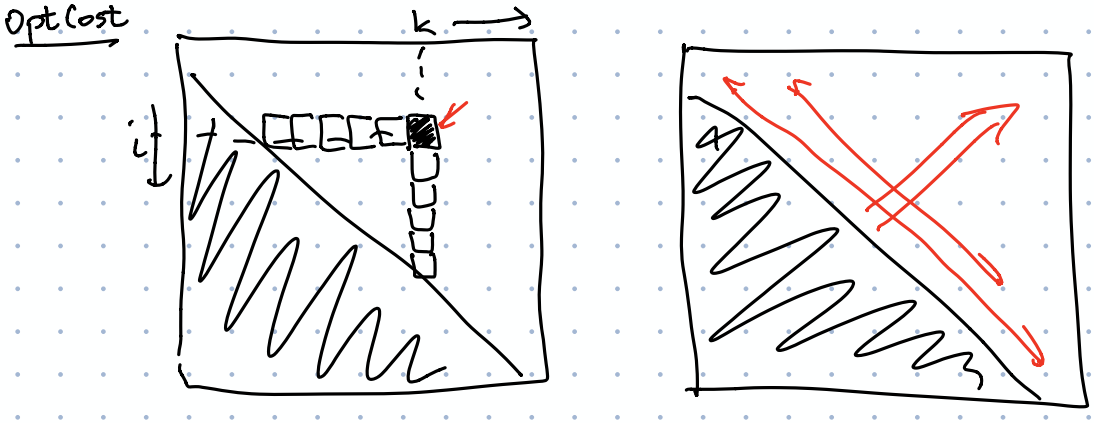
$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ F[i,k] + \min_{i \le r \le k} \{ OptCost(i, r-1) + OptCost(r+1, k) \} & \text{otherwise} \end{cases}$$

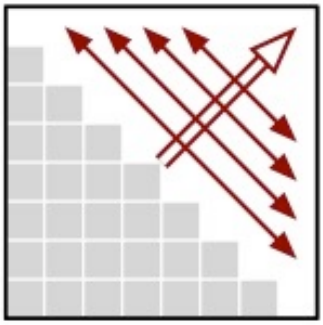
2 vars → $O(n^2)$ space

3 vars → $O(n^3)$ time

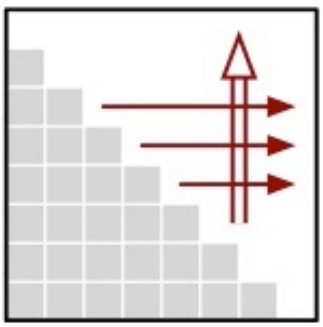
```
COMPUTEOPTCOST(i, k):
  OptCost[i, k] ← ∞
  for r ← i to k
      tmp ← OptCost[i, r − 1] + OptCost[r + 1, k]
      if OptCost[i, k] > tmp
          OptCost[i, k] ← tmp
  OptCost[i, k] ← OptCost[i, k] + F[i, k]
```

O(n)

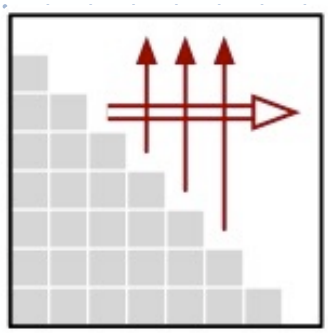




```
OptimalBST(f[1..n]):
  InitF(f[1..n])
  for i ← 1 to n+1
      OptCost[i, i−1] ← 0
  for d ← 0 to n−1
      for i ← 1 to n−d
          ComputeOptCost(i, i+d)
  return OptCost[1, n]
```

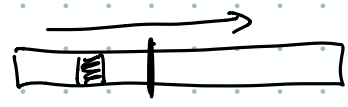


```
OptimalBST2(f[1..n]):
  InitF(f[1..n])
  for i ← n+1 downto 1
      OptCost[i, i−1] ← 0
      for j ← i to n
          ComputeOptCost(i, j)
  return OptCost[1, n]
```

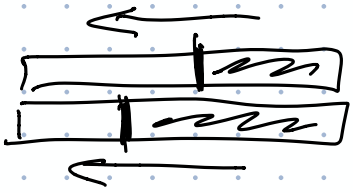


```
OptimalBST3(f[1..n]):
  InitF(f[1..n])
  for j ← 0 to n+1
      OptCost[j+1, j] ← 0
      for i ← j downto 1
          ComputeOptCost(i, j)
  return OptCost[1, n]
```

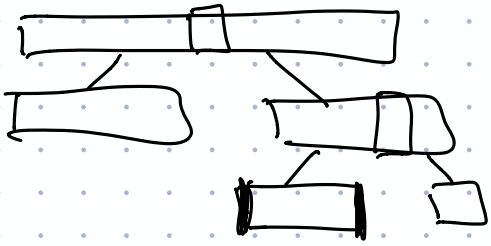
Longest Inc Subsequence



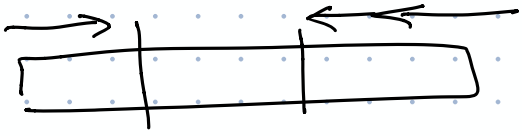
Edit Distance



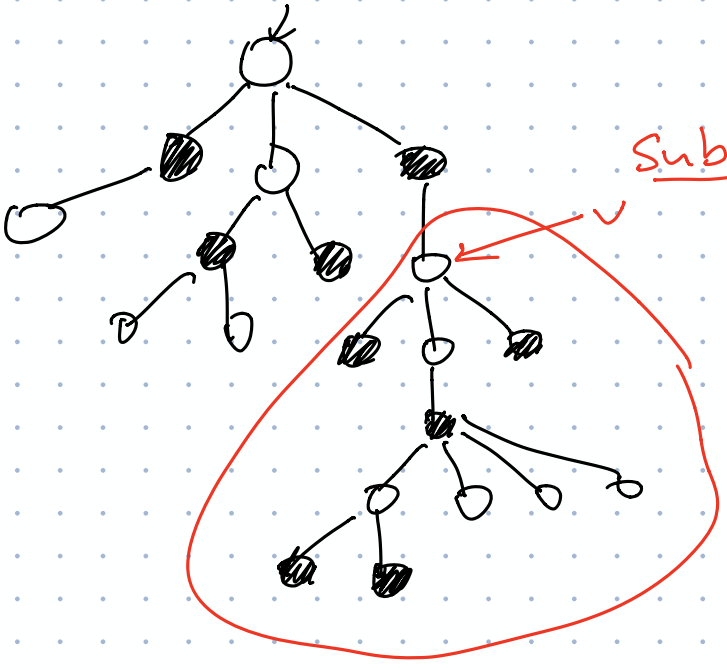
Opt BST



Longest Palindrome subsequence



Max Ind Set in Trees



Subproblem:
Is v part of the MIS of the subtree rooted at v
given the status of v's parent?

MIS(v, p) = size of largest I.S. of v's subtree
- excluding v if p = TRUE
- possibly including v if p = FALSE

node, boolean

w ↓ v = w is a child of v

$$MIS(v, \text{TRUE}) = \sum_{w \downarrow v} MIS(w, \text{FALSE})$$

$$MIS(v, \text{FALSE}) = \max \left\{ 1 + \sum_{w \downarrow v} MIS(w, \text{TRUE}),\ \sum_{w \downarrow v} MIS(w, \text{FALSE}) \right\}$$

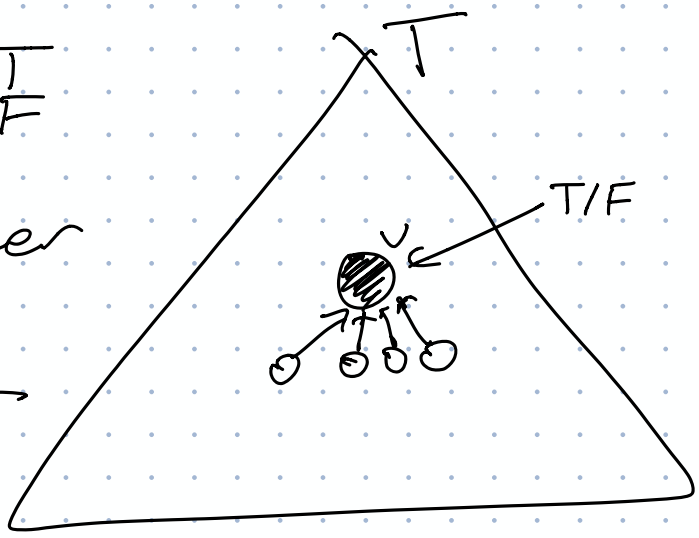
MIS(root, F) ← top-level call.

Memoize this into the input tree itself!

add v.MIST
fields v.MISF

eval in postorder

O(n) time



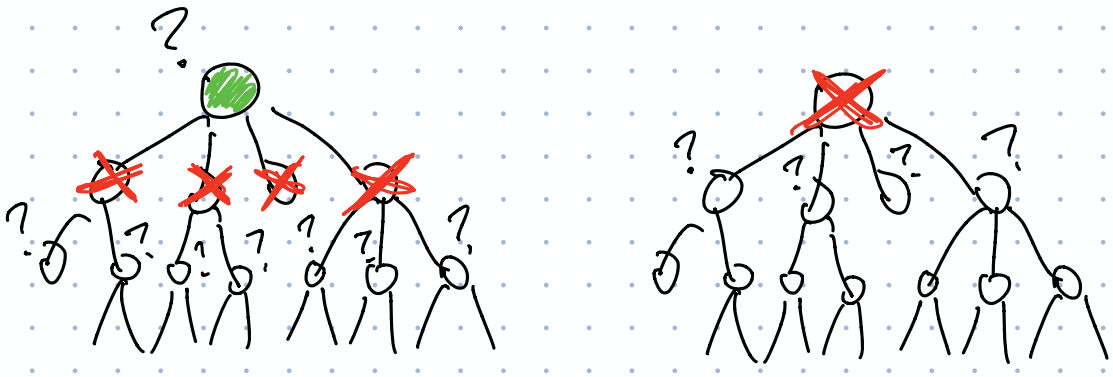
MIS(v) = size of largest ind. set in subtree rooted at v.

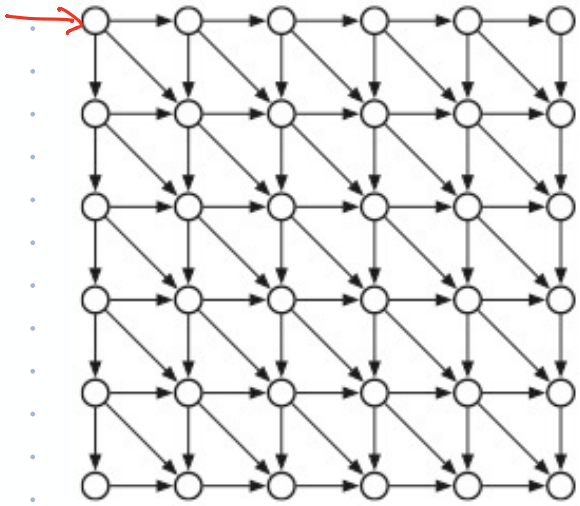
$$MIS(v) = \max\left\{\sum_{w \downarrow v} MIS(w),\ 1 + \sum_{w \downarrow v}\sum_{x \downarrow w} MIS(x)\right\}$$

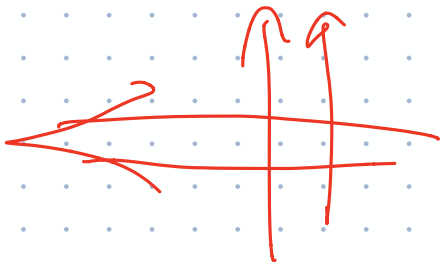
omit v — keep v

```
TREEMIS(v):
  skipv ← 0
  for each child w of v
      skipv ← skipv + TREEMIS(w)
  keepv ← 1
  for each grandchild x of v
      keepv ← keepv + x.MIS
  v.MIS ← max{keepv, skipv}
  return v.MIS
```

O(n)







```
DFS(v):
    if v is unmarked
        mark v
        PreVisit(x)
        for all edges v→w
            DFS(w)

        PostVisit(x)
```

```
Memoize(x):
    if value[x] is undefined
        initialize value[x]

        for all subproblems y of x
            Memoize(y)
            update value[x] based on value[y]
        finalize value[x]
```

```
DynamicProgramming(G):
    for all subproblems x in postorder
        initialize value[x]
        for all subproblems y of x
            update value[x] based on value[y]
        finalize value[x]
```