Optimal binary search tree

AVL
red-black
treap
splay
scapegoat
BB[α]

Worst-case depth — $\Theta(\log n)$

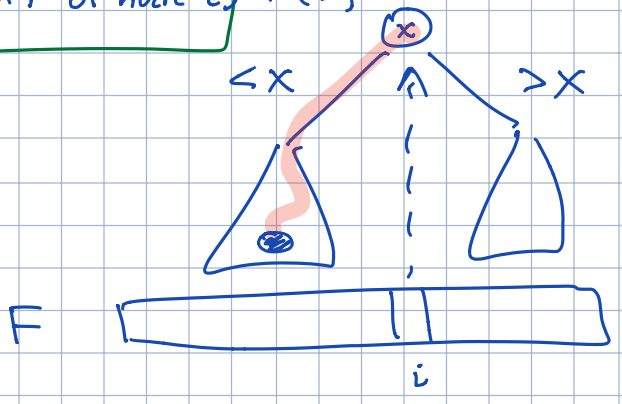
Given $A[1..n]$ search keys — sorted (w.l.o.g. $A[i] = i$)
$F[1..n]$ search frequencies

Guarantee: We will search for $A[i]$ exactly $F[i]$ times.

Build BST T stores $A[1..n]$ minimizes

$$\text{cost}(T) = \sum_{i=1}^{n} (\#\text{ancestors in } T \text{ of node } i) \cdot F[i]$$

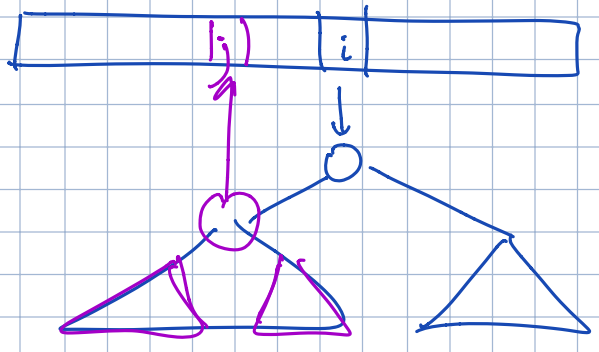
what goes here?



$$\text{cost}(T) = \sum_{i=1}^{r-1} \#\text{anc}_T(i) \cdot F[i] + F[r] + \sum_{i=r+1}^{n} \#\text{anc}_T(i) \cdot F[i]$$

$$= \sum_{i=1}^{r-1} \#\text{anc}_{\text{left}(T)}(i) \cdot F[i] + \sum_{i=1}^{r-1} F[i] + F[r] + \sum_{i=r+1}^{n} \#\text{anc}_{\text{right}(T)}(i) \cdot F[i] + \sum_{i=r+1}^{n} F[i]$$

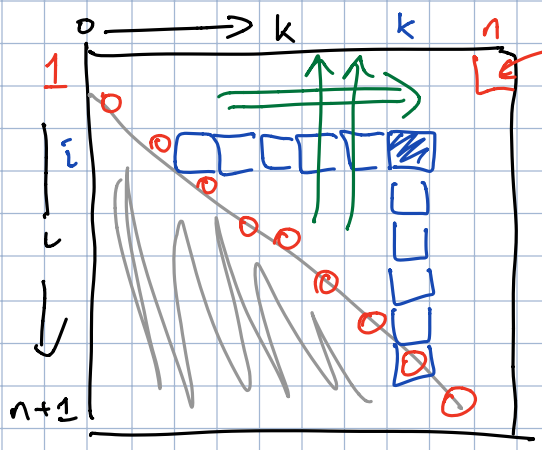
$$= \text{cost}(\text{left}(T)) + \sum_{i=1}^{n} F[i] + \text{cost}(\text{right}(T))$$



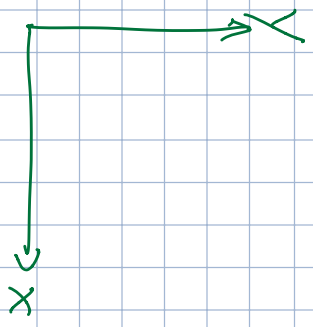
OptCost(i,k) =

Cost of optimal BST for $F[i..k]$.

$$OptCost(i,k) = \begin{cases} 0 & i > k \\ \sum_{j=i}^{k} F[j] + \min\{OptCost(i,j-1) + OptCost(j+1,k) \mid i \le j \le k\} \end{cases}$$



goal $OptCost(1,n)$



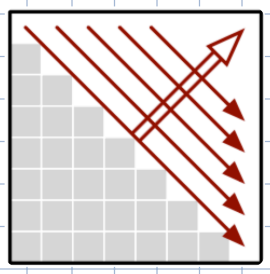

```
InitF(f[1..n]):
  for i ← 1 to n
      F[i, i−1] ← 0
      for k ← i to n
          F[i,k] ← F[i,k−1] + f[k]
```

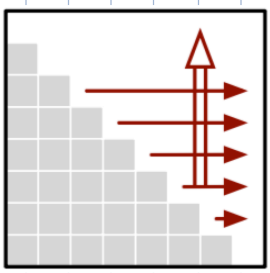
$O(n^2)$

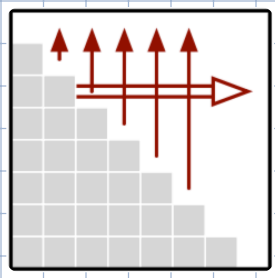
$\sum_{j=i}^{k} F[j]$

```
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]
```

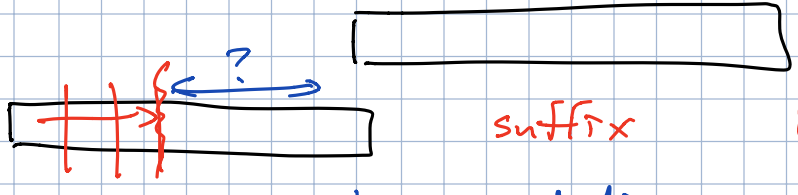
```
OptimalBST2(f[1..n]):
  InitF(f[1..n])
  for i ← n+1 downto 1
      OptCost[i, i−1] ← 0
      for j ← i to n
          ComputeOptCost(i, j)
  return OptCost[1, n]
```

O(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]
```

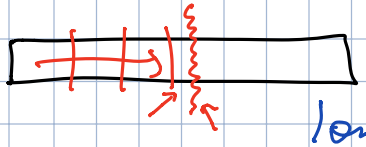
$O(n^3)$ time

General patterns



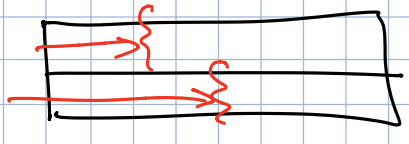
suffix — space $O(n)$ — time $O(n^2)$

Word segmentation



suffix+index $O(n^2)$ $O(n^2)$

longest incr subseq.

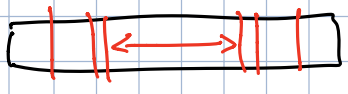


Two suffixes $O(n^2)$ $O(n^2)$

edit distance

substring/interval $O(n^2)$ $O(n^3)$

optimal BST



centered substring $O(n)$ $O(n^2)$

metapalindrome