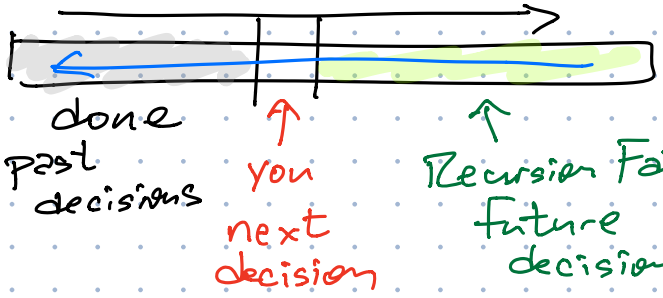
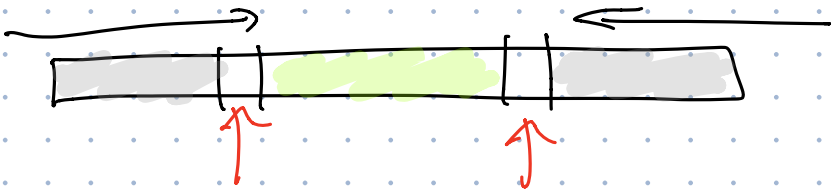


sequence(s)

palindromes



---

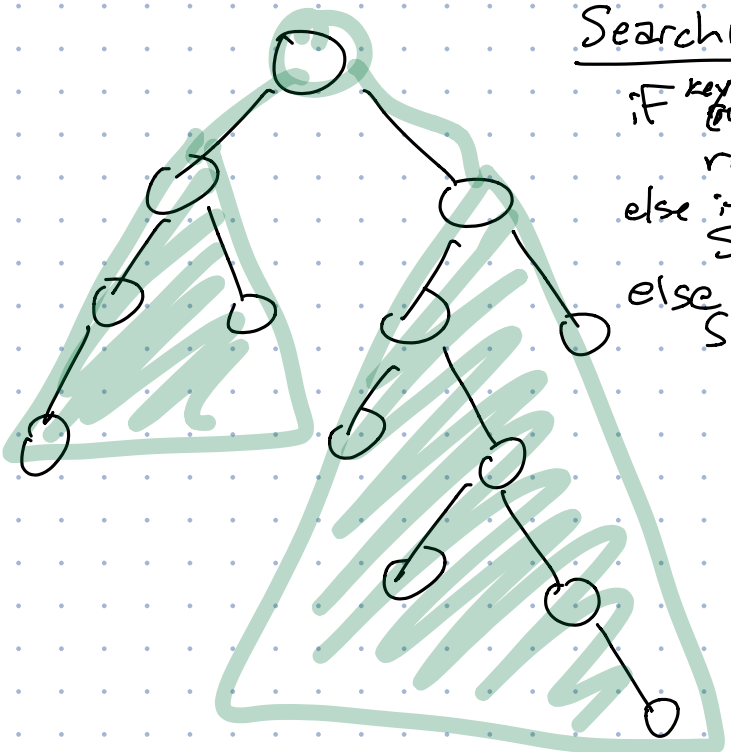
n keys → binary search tree (w/log 1..n)

known frequencies

f[1..n]

f[i] = #searches for key i

Build BST minimizing total cost of all searches.



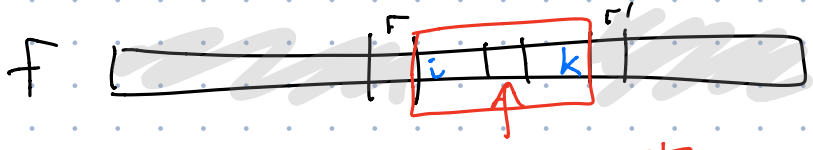
```
Search(x,T):
  if key(root(T)) = x
    return root(T)
  else if key(root(T)) < x
    Search(x, right(T))
  else
    Search(x, left(T))
```

Total Cost

total # searches

index of the root

$$Cost(T, f[1..n]) = \sum_{i=1}^{n} f[i] + Cost(left(T), f[1..r-1]) + Cost(right(T), f[r+1..n])$$



Where is the root for this interval?

Fix Frequency array $F[1..n]$

For any indices i and k, define

OptCost(i,k) = Total cost of best BST for keys $i..k$ with freq $f[i..k]$

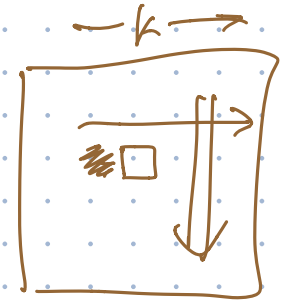
$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ \sum_{j=i}^{k} f[j] + \min_{i \le r \le k} \{ OptCost(i, r-1) + OptCost(r+1, k) \} & \text{otherwise} \end{cases}$$

Freq: 10 8 | 6 4 3 2 2 1

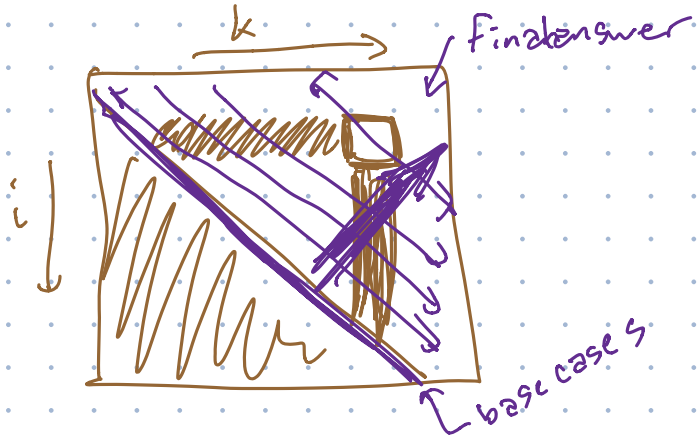
split the # accesses as evenly as possible

$$\sum_{j=i}^{k} f[i] = \begin{cases} 0 & \text{if } i > k \\ \sum_{j=i}^{k-1} F[i] + F[k] & \text{o/w} \end{cases}$$

```
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]
```



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




```
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) time

what is the right value of r?

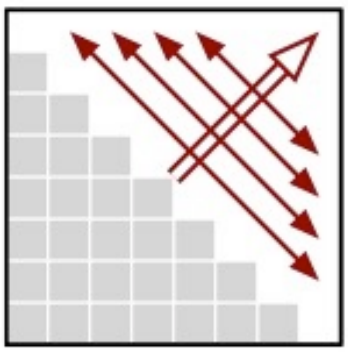
$O(n^2)$

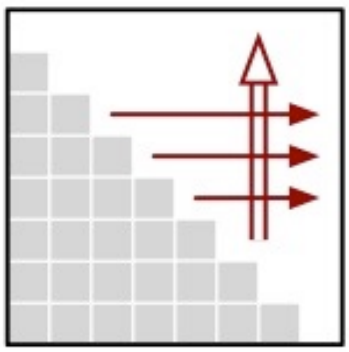
```
OptimalBST(f[1..n]):
  InitF(f[1..n])
  for i ← 1 to n + 1
      OptCost[i, i − 1] ← 0
  for d ← 0 to n − 1
      for i ← 1 to n − d
          ComputeOptCost(i, i + d)
  return OptCost[1, n]
```

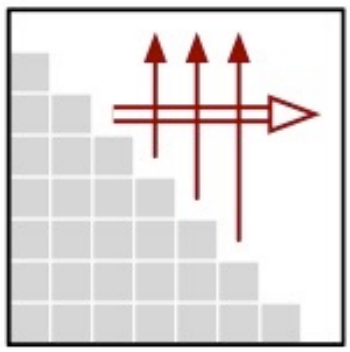
O(n)

$O(n^3)$ time

This is not best possible.







```
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        ⟨⟨...or whatever⟩⟩
          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]
```

$$OptCost(i,k) = \begin{cases} 0 & \text{if } i > k \\ F[i,k] + \min\limits_{i \le r \le k} \left\{ \begin{array}{l} OptCost(i, r-1) \\ \quad + OptCost(r+1, k) \end{array} \right\} & \text{otherwise} \end{cases}$$

Space $\approx O(n^{\#\text{vars on left}})$

time $\approx O(n^{\#\text{vars on right}})$