CSE525 Lec7: Backtracking

Debajyoti Bera (M21)

Longest Increasing Subsequence

BENT SQUARING ... are subsequences of SUBSEQUENCEBACKTRACKING

Q. Given integer array A[1 ... n], find its longest increasing subsequence.

Incremental Recursive Systematic

3	1	4	1	5	9	2	6	5	3	5	8

Q. What problem should be (recursively) solved? Assume that A is global.

- **LIS:** Given input i, return LIS of A[i ... n] X
- **LIS1:** Given input prev and i, return LIS of A[i ... n] in which every element is larger than **A[prev]**
- **LIS2:** Given input i, return LIS of A[i ... n] in which the subsequence starts with **A[i]**

Longest Increasing Subsequence

Q. Given integer array A[1 ... n], find its longest <u>increasing</u> subsequence.

Incremental Recursive Systematic

0	1	2	3	4								
$-\infty$	3	1	4	1	5	9	2	6	5	3	5	8

→ sentinel

LIS2(0) = length of LIS of A[0 ... n] in which the subseq. starts with $-\infty$

- **LIS2**(i): Given i, return LIS of A[i ... n] in which the subsequence starts with **A[i]**

$= \boxed{-\infty} \cdot LIS(A[1 \ldots n])$

LIS2(i) = 1 + max { **LIS2**(j) : i < j such that A[i] < A[j] }

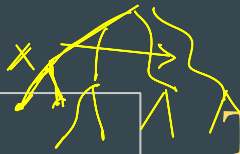
What is no such j exists?

(base) **LIS2**(n) = 1

LIS = ? LIS2(1), LIS2(2), ..., LIS2(n)

1. LIS(A) = max { **LIS2**(i) : all i }
2. LIS(A) = **LIS2**(0) - 1 in which A[0] = ~~-1~~ $-\infty$

Longest Increasing Subsequence



LIS2(3)

LIS2(5) — LIS2(7) — LIS2(8) — LIS2(9)

Time complexity?

$T(k)$: complexity of computing $LIS2[\underbrace{n-k+1 \dots n}_{k}]$

$T(1) = O(1)$

$T(k) \leq \sum_{j=1}^{k-1} T(j) + O(k)$

LIS2(i): Given i, return LIS of A[i … n] in which the subsequence starts with **A[i]**

LIS2(i) = 1 + max { LIS2(j) : i < j such that A[i] < A[j] }

(base) Solve(n) = 1

1. LIS(A) = max { LIS2(i) : all i } $\sum_{k=1}^{n} T(k)$

2. LIS(A) = LIS2(0) - 1 in which A[0] = -1

Time complexity : $T(n+1) + O(1)$

```
LISFIRST(i):
  best ← 0
  for j ← i + 1 to n
      if A[j] > A[i]
          best ← max{best, LISFIRST(j)}
  return 1 + best
```

```
LIS(A[1..n]):
  best ← 0
  for i ← 1 to n
      best ← max{best, LISFIRST(i)}
  return best
```

```
LIS(A[1..n]):
  A[0] ← −∞
  return LISFIRST(0) − 1
```

Longest Increasing Subsequence

Q. Given integer array A[1 ... n], find its longest increasing subsequence.

Incremental Recursive Systematic

1	2	3	4	5	6	7	8	9	10	11	12
3	1	4	1	5	9	2	6	5	3	5	8

- Given input prev and i where prev < i, return LIS of A[i ... n] in which every element of the LIS is larger than **A[prev]**

Solve(prev,i): If prev < i, return length of LIS of A[i ... n] in which every element is larger than **A[prev]**. If prev >= i, undefined.

Solve(1, 10) = ? Solve(1, 11) = ? Solve(1,12) = ?

Q: Write recursive expression to compute Solve(prev,i). Include boundary/base cases.
What happens to Soln. if A[prev] > A[i]? What happens to Soln. if A[prev] < A[i] ?

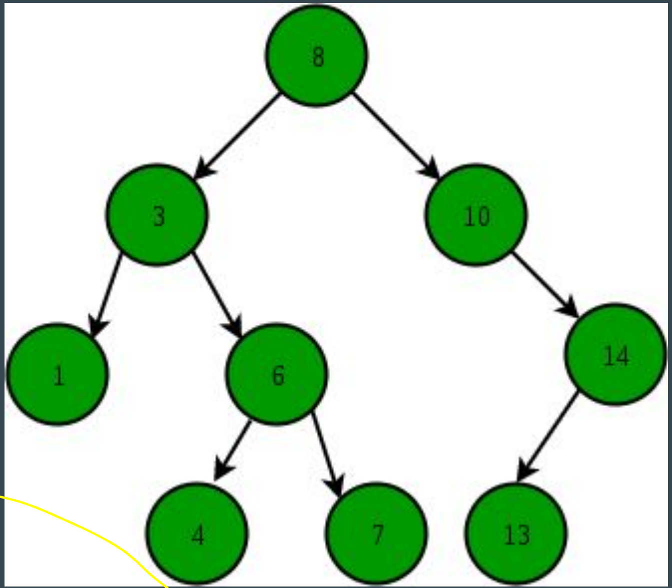
$B(n)\big|_{\text{root is } i} = B(i-1) * B(n-i)$

Binary search trees

Number of binary search trees from

Ex-1	1	3	4	6	7	8	10	13	14	n_1
Ex-2	1	2	3	4	5	6	7	8	9	n_2
Ex-3	20	12	2	130	13	23	34	14	42	n_3

i



B(n) = number of BSTs from n numbers

$[1 \dots n]$

Q: Write a recursive formula for B(n) w/ base case.

What is the first "thing" you draw in a BST ?

Can you "draw" a BST in a recursive manner ?

BST on $1 \dots i-1$

$BST(i+1 \dots n)$

$\equiv BST(1 \dots n-i)$

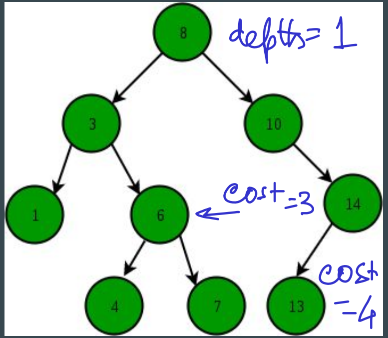
$B(1) = 1$ $\quad B(n) = \sum_{i=1}^{n} B(i-1) * B(n-i) = O(4^n)$

Optimal BST given search frequencies

3

1 4 − 14

key	1 1	3 2	4 3	6 4	7 5	8 6	10 7	13 8	14 9
freq	5	7	2	6	9	1	3	4	2



Given tree T, access Cost(T) = total cost of search in T = $\sum_{i=1}^{n} freq[i] \times depth(i,T)$

root

T.left 3.. ... 4

T.right

3 T.left

r is automatically accessed during access to 1, 2, ... r-1

cost(T.left) +
freq(1) + ... + freq[r-1]

Q: Given T with root r(T), Cost(T) = cost from r(T), cost from T.left, cost from T.right

freq[r] * depth(root) = 1

cost(T.right) +
freq(r+1) + ... + freq(n)

min
r

recursively

$$\min_r \left\{ \sum_{i=1}^{n} freq[i] + Cost(T.left) + Cost(T.right) \right\}$$

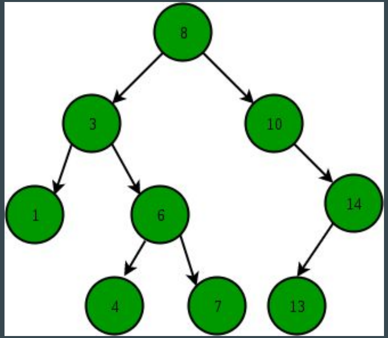
→ independently

does not depend on r

Cost T.left | root = 8

= cost of opt BST on elements 1 - 5, 3 - 7, 4 - 2, 6 - 9

Optimal BST given search frequencies



key	1	3	4	6	7	8	10	13	14
freq	5	7	2	6	9	1	3	4	2

Given tree T, access Cost(T) = total cost of search in T = $\sum_{i=1}^{n} freq[i] \times depth(i, T)$

Q: Given T with root r(T), Cost(T) = cost from r(T), cost from T.left, cost from T.right

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

Q: From key-freq table, construct tree with smallest cost. Solve OptCost(1,n).

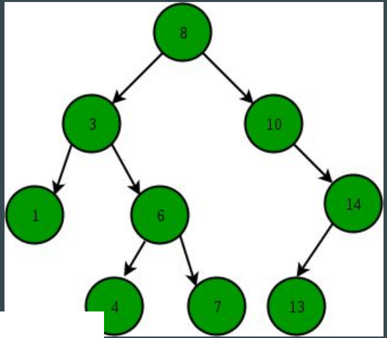
OptCost(i,k) = cost of minimum cost BST using keys { i, i+1, ..., k }

Suppose you knew how to solve OptCost(2,n) / OptCost(3,n) / ... OptCost(1,n-1).

$OptCost(1,n)$ = original problem

Optimal BST given search frequencies

key	1_1	3_2	4_3	6_4	7_5	8_6	10_7	13_8	14_9
freq	5	7	2	6	9	1	3	4	2



T.left | root = r

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

T

$T(j)$ j

T.right | root = r

j r $r-1$ $n-r$

Q: What is the recurrence for time-complexity?

T(i,k) = time-complexity to compute OptCost(i,k)

T(j) = time-complexity to compute OptCost([any range with j elements])

$T(3)$ = cost to compute $OptCost(1..3)$

= ... ([2...4])

$$T(j) = \sum_{t=1}^{j} [T(t-1) + T(j-t)] + O(j) = O(3^n)$$

Q: Solve recurrence. Compare with number of total number of BSTs.

Show that T(n) - T(n-1) = 2T(n-1) + ...

$$T(j) = \sum_{t=1}^{j} \left[T(t-1) + T(j-t) \right] + cj = T(0) + T(j-1) + T(1) + T(j-2) + \cdots + T(j-1) + T(0) + cj$$

$$T(j) = 2\sum_{t=0}^{j-1} T(t) + cj$$

$$O(n^2) = \{ f(n) : \cdots \}$$

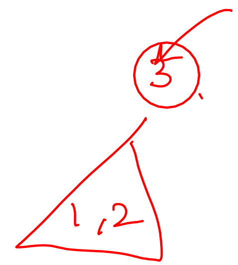
$$T(j-1) = 2\sum_{t=0}^{j-2} T(t) + c(j-1)$$

$$T(j) - T(j-1) = 2T(j-1) + c$$

$$T(j) = 3T(j-1) + c = O(3^n).$$

$$\min_x \left(f(x) + g(x) \right) = \min_x f(x) + \min_x g(x)$$

1	2	3	1.
f_1	f_2	f_3	



$$\text{Cost}(3) + \text{Cost}(4) + \boxed{\text{Cost}(2) + \text{Cost}(1)}$$

$$\text{Cost}(2,1)$$

given 3 ~~is the~~ root

opt. BST ~~for~~ 2, 1

$$f_1 + f_2$$

$$\text{Cost}(T) = \text{Cost}(3) + f_1 + f_2$$

+ ~~access root~~ while acce 2,1