Lecture 6

Instructor: Subrahmanyam Kalyanasundaram

26th August 2019

Plan

- Last class, we saw DELETE in Red-Black Trees

Plan

- Last class, we saw Delete in Red-Black Trees

- Today, order of insertion in BST
- Randomized quicksort
- Traversals of Binary Trees

Course grading scheme

- 60% – Exams (2 or 3)
- 30% – Programming Assignments
- 10% – Attendance and Quizzes

Course grading scheme

- 60% – Exams (2 or 3)
- 30% – Programming Assignments
- 10% – Attendance and Quizzes

Exam on Thursday, 5 Sep

Traversals of a Binary Tree

Pre-order Traversal: Pre-order(v)

Traverse v, then do Pre-order(left(v)), and then do Pre-order(right(v)).

Traversals of a Binary Tree

Pre-order Traversal: PRE-ORDER(v)

Traverse v, then do PRE-ORDER(left(v)), and then do PRE-ORDER(right(v)).

Post-order Traversal: POST-ORDER(v)

Do POST-ORDER(left(v)), then do POST-ORDER(right(v)), and then traverse v.

Traversals of a Binary Tree

Pre-order Traversal: PRE-ORDER(v)

Traverse v, then do PRE-ORDER(left(v)), and then do PRE-ORDER(right(v)).

Post-order Traversal: POST-ORDER(v)

Do POST-ORDER(left(v)), then do POST-ORDER(right(v)), and then traverse v.

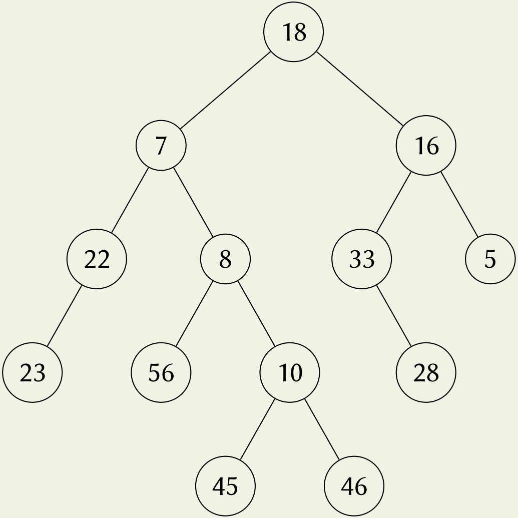
In-order Traversal: IN-ORDER(v)

Do IN-ORDER(left(v)), then traverse v, and then do IN-ORDER(right(v)).

Traversals of a Binary Tree

Pre-order Traversal

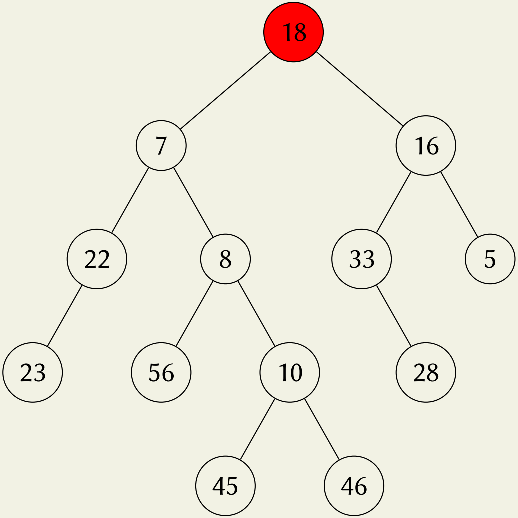
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5



Traversals of a Binary Tree

Pre-order Traversal

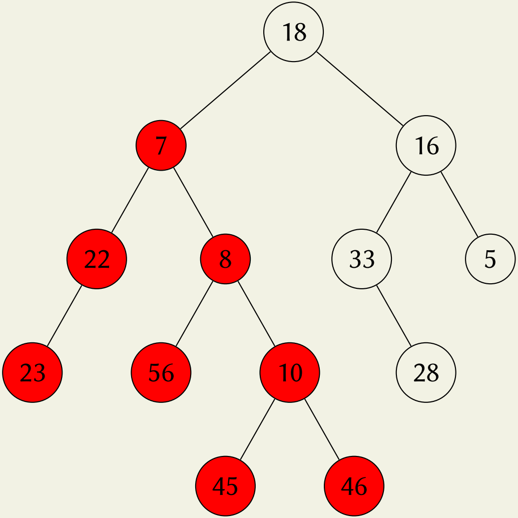
18, 7, 22, 23, 8, 56, 10,
45, 46, 16, 33, 28, 5



Traversals of a Binary Tree

Pre-order Traversal

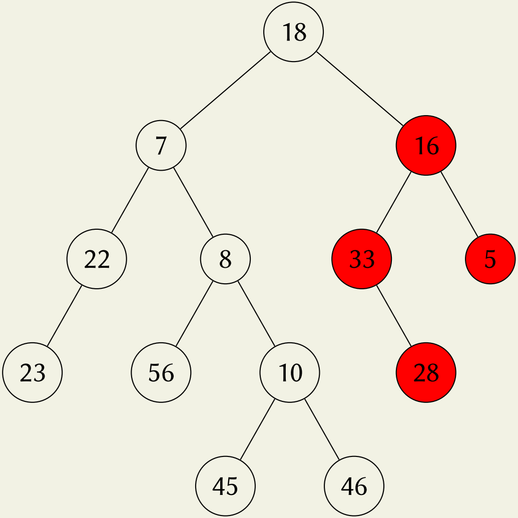
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5



Traversals of a Binary Tree

Pre-order Traversal

18, 7, 22, 23, 8, 56, 10,
45, 46, 16, 33, 28, 5



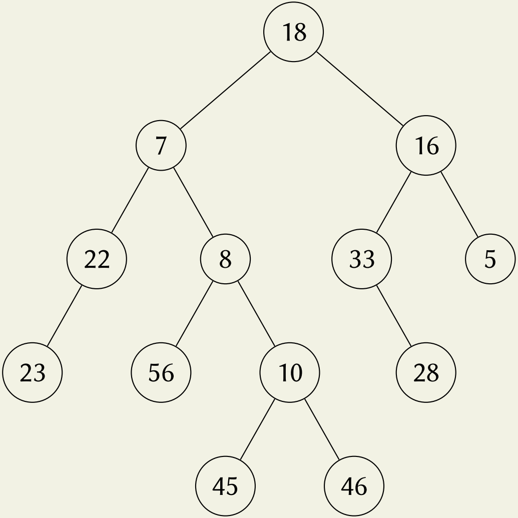
Traversals of a Binary Tree

Pre-order Traversal

18, 7, 22, 23, 8, 56, 10,
45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10,
8, 7, 28, 33, 5, 16, 18



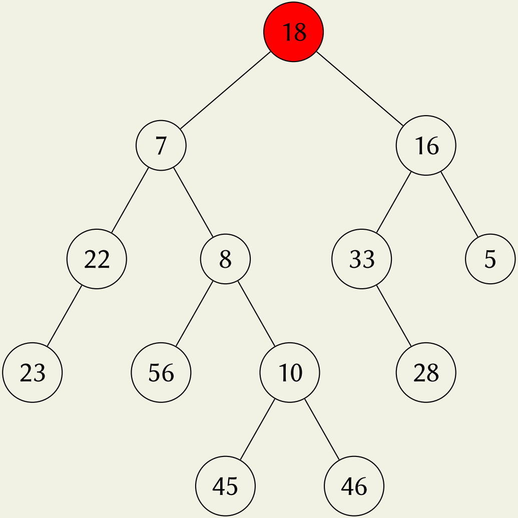
Traversals of a Binary Tree

Pre-order Traversal

18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18



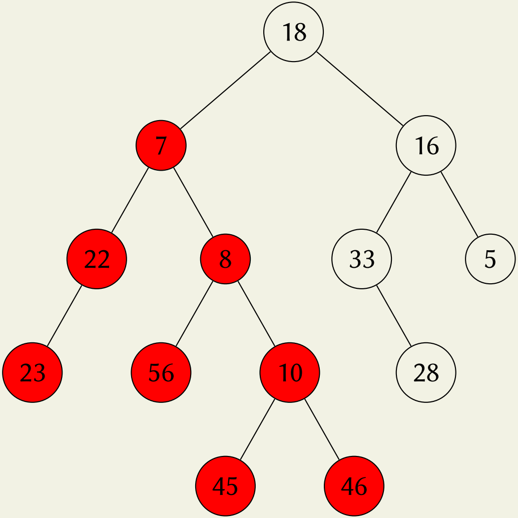
Traversals of a Binary Tree

Pre-order Traversal

18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18



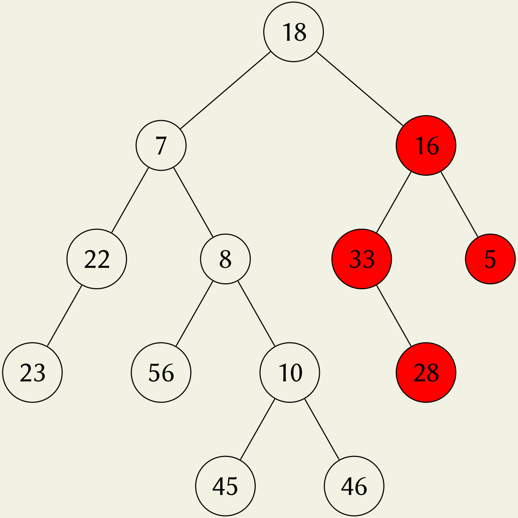
Traversals of a Binary Tree

Pre-order Traversal

18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18



Traversals of a Binary Tree

Pre-order Traversal

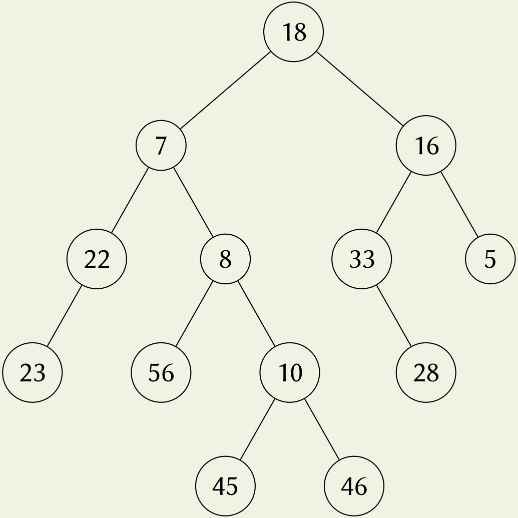
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18

In-order Traversal

23, 22, 7, 56, 8, 45, 10, 46, 18, 33, 28,16, 5



Traversals of a Binary Tree

Pre-order Traversal

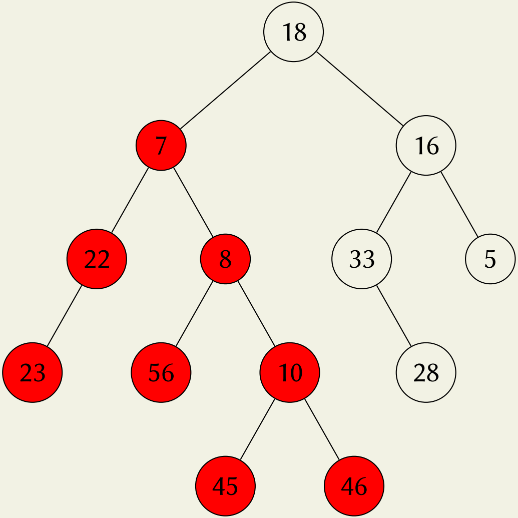
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18

In-order Traversal

23, 22, 7, 56, 8, 45, 10, 46, 18, 33, 28,16, 5



Traversals of a Binary Tree

Pre-order Traversal

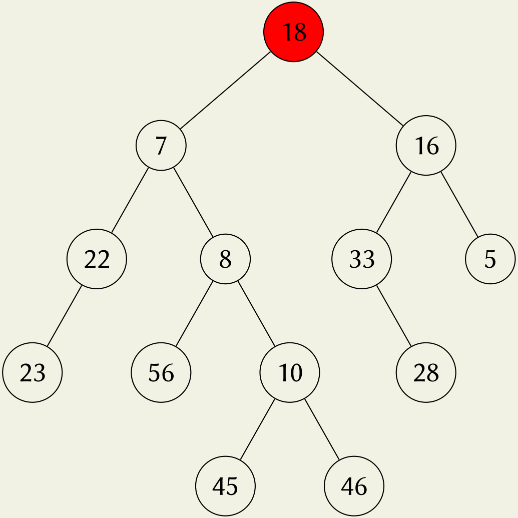
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18

In-order Traversal

23, 22, 7, 56, 8, 45, 10, 46, 18, 33, 28,16, 5



Traversals of a Binary Tree

Pre-order Traversal

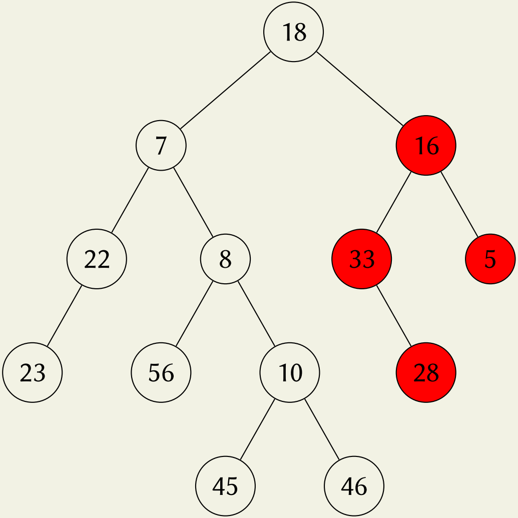
18, 7, 22, 23, 8, 56, 10, 45, 46, 16, 33, 28, 5

Post-order Traversal

23, 22, 56, 45, 46, 10, 8, 7, 28, 33, 5, 16, 18

In-order Traversal

23, 22, 7, 56, 8, 45, 10, 46, 18, 33, 28,16, 5



Question

What does the in-order traversal of a BST look like?

Question

What does the in-order traversal of a BST look like?

BST Sorting Algorithm

- Insert the elements sequentially into a BST
- Do an in-order traversal of the BST

Time Complexity of BST Sort

- Insert the elements sequentially into a BST
- $n \cdot O(h) \leq n \cdot O(n) = O(n^2)$.

Time Complexity of BST Sort

- Insert the elements sequentially into a BST
- $n \cdot O(h) \leq n \cdot O(n) = O(n^2)$.

- Do an in-order traversal of the BST
- $O(n)$.

- Total time complexity is $O(n^2)$ in worst case.

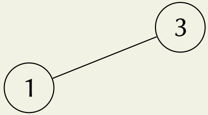
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5

3

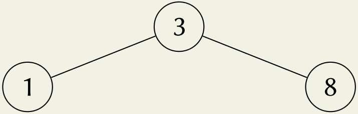
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



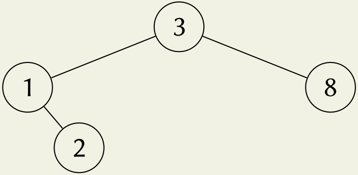
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



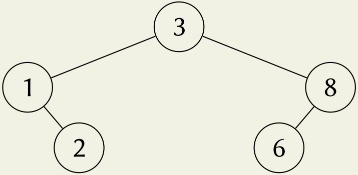
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



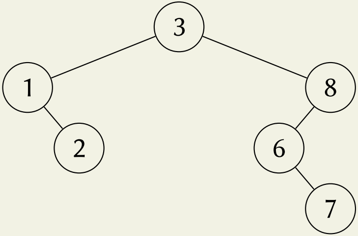
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



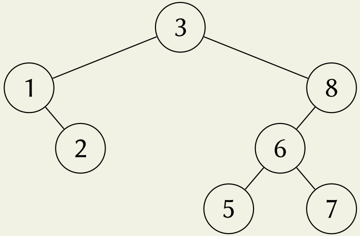
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



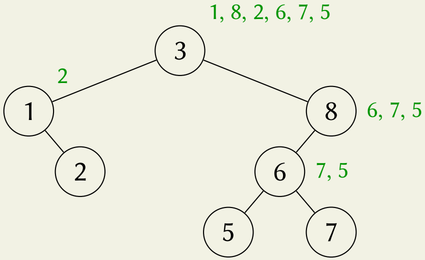
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5



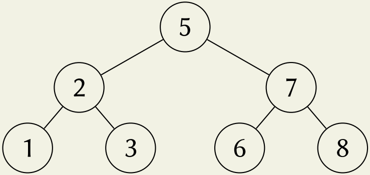
An example

Consider the sequence 3, 1, 8, 2, 6, 7, 5

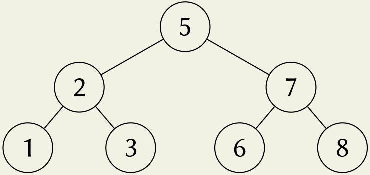


12 Comparisons

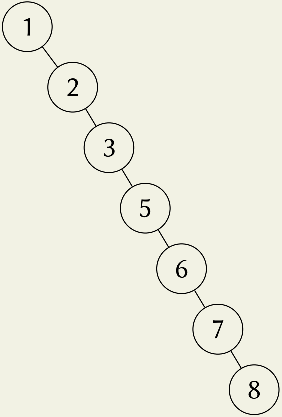
Order 1: 5, 2, 7, 8, 3, 1, 6



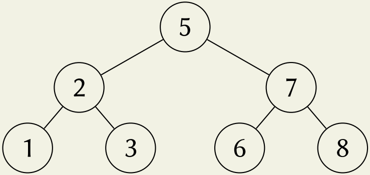
Order 1: 5, 2, 7, 8, 3, 1, 6



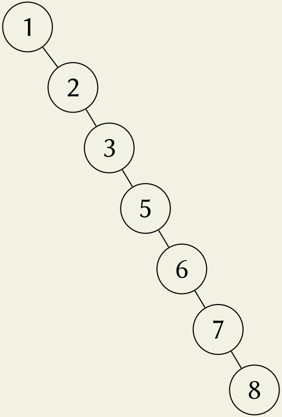
Order 2: 1, 2, 3, 5, 6, 7, 8



Order 1: 5, 2, 7, 8, 3, 1, 6



Order 2: 1, 2, 3, 5, 6, 7, 8



Comparison Count

Order 1: 10
Order 2: 21

A detour: Quicksort

The Goal

Given an array A of n elements, arrange the elements in increasing order.

Quicksort(A, s, t)

1. If $s \geq t$, exit.
2. Choose pivot p from $\{s, s+1, \ldots, t\}$
3. $q =$Partition(A, s, t, p). Partition(A, s, t, p) partitions $A(s, t)$ in place into less than pivot, pivot and greater than pivot. It also returns the correct index of p.
4. Quicksort($A, s, q-1$)
5. Quicksort($A, q+1, t$)

Deterministic Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Deterministically choose pivot p from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

Deterministic Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Deterministically choose pivot p from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

- For instance, pivot p is always the first element.

Deterministic Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Deterministically choose pivot p from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

- For instance, pivot p is always the first element.
- The running time is determined by the number of comparisons.

Deterministic Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Deterministically choose pivot p from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

- For instance, pivot p is always the first element.
- The running time is determined by the number of comparisons.
- Any deterministic pivot rule requires worst case $\Omega(n^2)$ comparisons.
- One can come up with a bad input order for any deterministic pivot rule.

Deterministic Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Deterministically choose pivot p from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

- For instance, pivot p is always the first element.
- The running time is determined by the number of comparisons.
- Any deterministic pivot rule requires worst case $\Omega(n^2)$ comparisons.
- One can come up with a bad input order for any deterministic pivot rule.
- Can randomization help?

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If we choose the median as the pivot, the recurrence for number of comparisons is

$$T(n) = 2T(n/2) + (n-1)$$

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If we choose the median as the pivot, the recurrence for number of comparisons is

$$T(n) = 2T(n/2) + (n - 1)$$

- This solves to $\Theta(n \log n)$

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If pivot gives a $[n/10, 9n/10]$-split, we get the recurrence.

$$T(n) = T(n/10) + T(9n/10) + (n-1)$$

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If pivot gives a $[n/10, 9n/10]$-split, we get the recurrence.

$$T(n) = T(n/10) + T(9n/10) + (n-1)$$

- Even this gives us $\Theta(n \log n)$ number of comparisons.

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If pivot gives a $[n/10, 9n/10]$-split, we get the recurrence.

$$T(n) = T(n/10) + T(9n/10) + (n-1)$$

- Even this gives us $\Theta(n \log n)$ number of comparisons.

- A random pivot is likely to work with probability 0.8.

Why randomize?

- Worst case occurs when we repeatedly choose the smallest/largest number as pivot.
- A good pivot separates the array into two (roughly) equal parts.
- If pivot gives a $[n/10, 9n/10]$-split, we get the recurrence.

$$T(n) = T(n/10) + T(9n/10) + (n-1)$$

- Even this gives us $\Theta(n \log n)$ number of comparisons.

- A random pivot is likely to work with probability 0.8.
- This is still an intuition.

Randomized Quicksort

Quicksort(A, s, t)

- If $s \geq t$, exit.
- Choose pivot p uniformly at random from $\{s, s+1, \ldots, t\}$
- $q =$Partition(A, s, t, p).
- Quicksort($A, s, q-1$)
- Quicksort($A, q+1, t$)

Analysis of Randomized Quicksort

- Let the numbers in A be $z_1 < z_2 < \ldots < z_n$.

Analysis of Randomized Quicksort

- Let the numbers in A be $z_1 < z_2 < \ldots < z_n$.

- Let $X_{i,j}$ denote an indicator random variable for all $1 \leq i < j \leq n$.
- If z_i is compared to z_j during the execution of the algorithm, $X_{i,j} = 1$.
- Otherwise $X_{i,j} = 0$

Analysis of Randomized Quicksort

- Let the numbers in A be $z_1 < z_2 < \ldots < z_n$.

- Let $X_{i,j}$ denote an indicator random variable for all $1 \leq i < j \leq n$.
- If z_i is compared to z_j during the execution of the algorithm, $X_{i,j} = 1$.
- Otherwise $X_{i,j} = 0$

The total no. of comparisons X is given by

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}$$

Analysis of Randomized Quicksort

- Let the numbers in A be $z_1 < z_2 < \ldots < z_n$.

- Let $X_{i,j}$ denote an indicator random variable for all $1 \leq i < j \leq n$.
- If z_i is compared to z_j during the execution of the algorithm, $X_{i,j} = 1$.
- Otherwise $X_{i,j} = 0$

The total no. of comparisons X is given by

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}$$

- Correct because $X_{i,j}$ takes only values from $\{0, 1\}$.
- Also because no two z_i and z_j are compared more than once.

Analysis of Randomized Quicksort

- Need to calculate expected number of comparisons $E(X)$.

Analysis of Randomized Quicksort

- Need to calculate expected number of comparisons $E(X)$.

Linearity of Expectations

$$E(X) = E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E(X_{i,j})$$

Analysis of Randomized Quicksort

- Need to calculate expected number of comparisons $E(X)$.

Linearity of Expectations

$$E(X) = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{i,j})$$

- For indicator random variable, $E(X_{i,j}) = \Pr(X_{i,j} = 1)$

Analysis of Randomized Quicksort

- Need to calculate expected number of comparisons $E(X)$.

Linearity of Expectations

$$E(X) = E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E(X_{i,j})$$

- For indicator random variable, $E(X_{i,j}) = \Pr(X_{i,j} = 1)$
- What is the probability that z_i was compared to z_j?

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$
- z_i is compared to z_j if and only if one of them is chosen as pivot.

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$
- z_i is compared to z_j if and only if one of them is chosen as pivot.

Claim

$X_{i,j} = 1$ (z_i is compared to z_j) if and only if the first pivot chosen from $Z_{i,j}$ is z_i or z_j.

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$
- z_i is compared to z_j if and only if one of them is chosen as pivot.

Claim

$X_{i,j} = 1$ (z_i is compared to z_j) if and only if the first pivot chosen from $Z_{i,j}$ is z_i or z_j.

- As long as pivots in $Z_{i,j}$ are not chosen, z_i and z_j are never separated by the algorithm.

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$
- z_i is compared to z_j if and only if one of them is chosen as pivot.

Claim

$X_{i,j} = 1$ (z_i is compared to z_j) if and only if the first pivot chosen from $Z_{i,j}$ is z_i or z_j.

- As long as pivots in $Z_{i,j}$ are not chosen, z_i and z_j are never separated by the algorithm.
- If z_i or z_j is the first pivot chosen from $Z_{i,j}$, then z_i is compared to z_j.

Analysis of Randomized Quicksort

- Let $Z_{i,j} = \{z_i, z_{i+1}, \ldots, z_j\}$
- z_i is compared to z_j if and only if one of them is chosen as pivot.

Claim

$X_{i,j} = 1$ (z_i is compared to z_j) if and only if the first pivot chosen from $Z_{i,j}$ is z_i or z_j.

- As long as pivots in $Z_{i,j}$ are not chosen, z_i and z_j are never separated by the algorithm.
- If z_i or z_j is the first pivot chosen from $Z_{i,j}$, then z_i is compared to z_j.
- If the first pivot is from $Z_{i,j} \backslash \{z_i, z_j\}$, then z_i and z_j are never compared.

Analysis of Randomized Quicksort

- What is the probability that z_i was compared to z_j?

Analysis of Randomized Quicksort

- What is the probability that z_i was compared to z_j?
- What is the probability that z_i or z_j is the first chosen pivot from $Z_{i,j}$?

Analysis of Randomized Quicksort

- What is the probability that z_i was compared to z_j?
- What is the probability that z_i or z_j is the first chosen pivot from $Z_{i,j}$?
- Since $|Z_{i,j}| = j - i + 1$,

$$E(X_{i,j}) = \Pr(X_{i,j} = 1) = 2/(j - i + 1).$$

Analysis of Randomized Quicksort

$$
\begin{aligned}
E(X) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{i,j}) \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2/(j-i+1) \\
&= 2 \cdot \sum_{i=1}^{n-1} \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+1} \right) \\
&\leq 2 \cdot \sum_{i=1}^{n-1} \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \right) = 2(n-1)H_n.
\end{aligned}
$$

Analysis of Randomized Quicksort

$$
\begin{aligned}
H_n &= \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \\
&\leq \int_1^n \frac{1}{y} dy \\
&= \ln n - \ln 1 = \ln n
\end{aligned}
$$

Analysis of Randomized Quicksort

$$\begin{aligned} H_n &= \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \\ &\leq \int_1^n \frac{1}{y} dy \\ &= \ln n - \ln 1 = \ln n \end{aligned}$$

- $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is the harmonic series.
- H_n is $\Theta(\log n)$.

Analysis of Randomized Quicksort

$$\begin{aligned} H_n &= \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \\ &\leq \int_1^n \frac{1}{y} dy \\ &= \ln n - \ln 1 = \ln n \end{aligned}$$

- $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is the harmonic series.
- H_n is $\Theta(\log n)$.

$$E(X) = 2(n-1)H_n = \Theta(n \log n).$$

Randomized Quicksort

Theorem

Randomized Quicksort correctly sorts the input array in-place and requires $\Theta(n \log n)$ comparisons in expectation.

Randomized Quicksort

Theorem

Randomized Quicksort correctly sorts the input array in-place and requires $\Theta(n \log n)$ comparisons in expectation.

- Can still take $\Theta(n^2)$ time in worst case.
- But with low probability.

Randomized Quicksort

Theorem

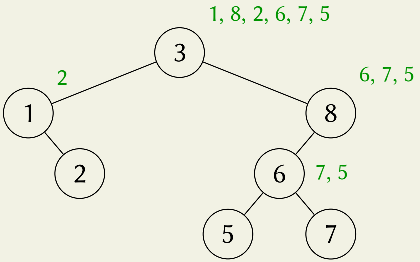
Randomized Quicksort correctly sorts the input array in-place and requires $\Theta(n \log n)$ comparisons in expectation.

- Can still take $\Theta(n^2)$ time in worst case.
- But with low probability.

- Instead of random pivot choice each time, we could also permute the input in a random order at the beginning and then choose first element as pivot.
- These two processes are equivalent.

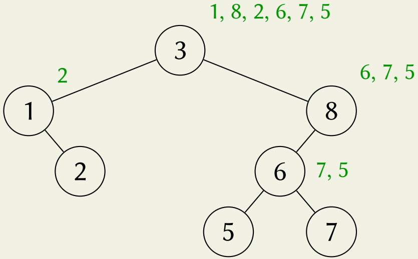
Back to BST Sort: What is the connection?

Consider the sequence 3, 1, 8, 2, 6, 7, 5



Back to BST Sort: What is the connection?

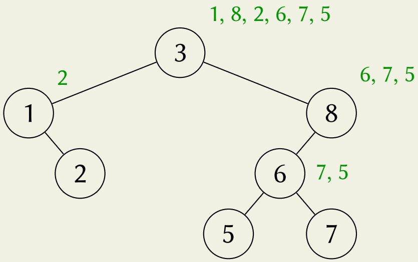
Consider the sequence 3, 1, 8, 2, 6, 7, 5



What are the comparisons if we ran det. quicksort with the first element as pivot?

Back to BST Sort: What is the connection?

Consider the sequence 3, 1, 8, 2, 6, 7, 5



What are the comparisons if we ran det. quicksort with the first element as pivot?
Exactly the same!

Randomize!

- The comparisons for BST sort are the same comparisons that happen in quicksort!
- In worst case, we can have $O(n^2)$ comparisons
- What if we randomize?

Randomize!

- The comparisons for BST sort are the same comparisons that happen in quicksort!
- In worst case, we can have $O(n^2)$ comparisons
- What if we randomize?

- We randomly permute the input sequence before insertion
- The total no. of comparisons are the same as that of randomized quicksort
- The running time of random BST Sort is thus $O(n \log n)$

What does this tell us about the randomly built BST?

- Depth of a node = No. of comparisons while INSERT.

$$
\begin{aligned}
\text{Average node depth} &= \frac{1}{n}\sum_{\text{node} i} \text{Depth of node } i \\
&= \frac{1}{n}\sum_{\text{node} i} \text{No. of comp. while inserting node} i \\
E\,(\text{Avg. node depth}) &= \frac{1}{n}E\left(\sum_{\text{node} i} \text{No. of comp. while inserting node } i\right) \\
&= \frac{1}{n}O(n\log n) \qquad \text{By quicksort} \\
&= O(\log n)
\end{aligned}
$$

Randomly built BST

- Expected average node depth $= O(\log n)$
- What about about expected height?
- Does $O(\log n)$ average depth imply $O(\log n)$ height?

Randomly built BST

- Expected average node depth $= O(\log n)$
- What about about expected height?
- Does $O(\log n)$ average depth imply $O(\log n)$ height?

- No!

Relation between average depth and height

- Consider a BST in which $n - \sqrt{n}$ nodes form a complete binary tree and the remaining $\sqrt{n}$ nodes form a chain
- Height is $\sqrt{n}$
- Average node depth is

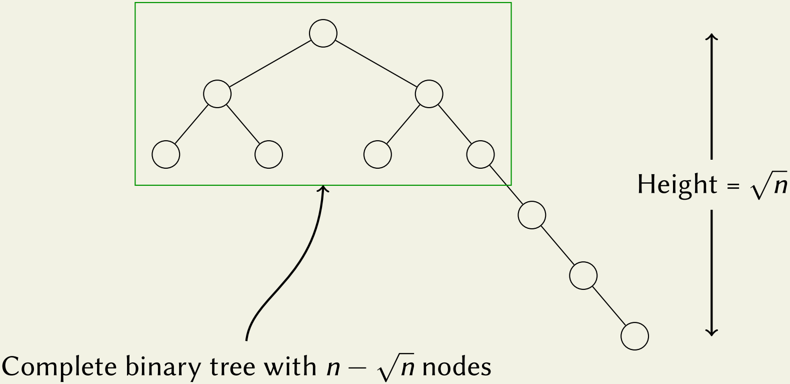
$$\leq \frac{1}{n}\left(n \log n + \frac{\sqrt{n} \cdot \sqrt{n}}{2}\right) = \frac{n \log n}{n} \approx O(\log n)$$

Relation between average depth and height

- Consider a BST in which $n - \sqrt{n}$ nodes form a complete binary tree and the remaining $\sqrt{n}$ nodes form a chain
- Height is $\sqrt{n}$
- Average node depth is

$$\leq \frac{1}{n}\left(n \log n + \frac{\sqrt{n} \cdot \sqrt{n}}{2}\right) = \frac{n \log n}{n} \approx O(\log n)$$

- Low average depth does not imply low height.


Height = $\sqrt{n}$
Complete binary tree with $n - \sqrt{n}$ nodes

Expected height of randomly built BST

- We saw that in general, low average depth does not imply low height.
- However, in the case of randomly built BSTs, we can show that the expected height is also $O(\log n)$.
- This proof is more involved, and can be found in CLRS.
- If anyone is interested, you can meet me.