Lecture 5 - Sep 17

Asymptotic Analysis,
Self-Balancing Binary Search Trees

Amortized RT: Constant Increments
Deriving Sum of Geometric Seq.
Height Balance Property

Announcements/Reminders

- First Class (Syllabus) recording & notes posted
- Today's class: notes template posted
- Exercises:
  - Tutorial Week 1 (2D arrays)
  - Tutorial Week 2 (2D arrays, Proving Big-O)
- Tutorial Week ~~2~~ 3 (this week)
  - No in-person attendance.
  - Exercises will be assigned.

average RT = * total RT / # ops. → e.g. n pushes

* over a seq of push operations

Amortized Analysis: Dynamic Array with Const. Increments

* without loss of generality

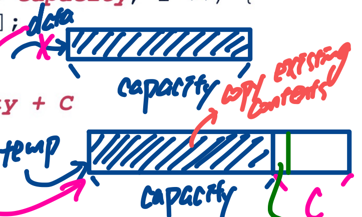
```java
public class ArrayStack<E> implements Stack<E> {
  private int I;   // init. capacity
  private int C;   // → extra space to allocate when full
  private int capacity;   // current limit.
  private E[] data;
  public ArrayStack() {
    I = 1000; /* arbitrary initial size */
    C = 500; /* arbitrary fixed increment */
    capacity = I;
    data = (E[]) new Object[capacity];
    t = -1;
  }
  public void push(E e) {
    if (size() == capacity) {
      /* resizing by a fixed constant */
      E[] temp = (E[]) new Object[capacity + C];
      for(int i = 0; i < capacity; i ++) {
        temp[i] = data[i];
      }
      data = temp;
      capacity = capacity + C
    }
    t++;
    data[t] = e;
  }
}
```

sizes: 1000, 1500, 2000, ...

n

→ when array is full, increase its size by C

worst-case RT: $O(n)$ ↳ # of elements in a full array



$O(1)$ → RT when resizing not needed.

resizing step

k			RT
	initial array:	$[\,I\,]$	
①	1st resizing: (after I pushes)	$[\,I \mid C\,]$	$I + 0 \cdot C$
②	2nd resizing: (after C pushes)	$[\,I \mid C \mid C\,]$	$I + 1 \cdot C$
③	3rd resizing: (after C pushes)	$[\,I \mid C \mid C \mid C\,]$	$I + 2 \cdot C$
⋮	⋮	⋮	
$k = ?$	Last resizing:	$\underbrace{[\,I \mid C \mid C \mid C \mid \cdots \mid C \mid C\,]}_{I + (k-1) \cdot C}$ (size n)	

$$n = I + (k-1) \cdot C \iff k = \boxed{\frac{n-I}{C} + 1}$$

Total RT = $\sum$ resizing steps

$$= I + (I + C) + (I + 2C) + \dots + n$$

$$= \frac{(I+n) \cdot \left(\frac{n-I}{C} + 1\right)}{2} \text{ is } O(n^2)$$

* W.L.O.G, assume: n pushes (consecutive) → first n elements stored.

and the last push triggers the last resizing routine.

Amortized/ Average RT: $\boxed{O\left(\frac{n^2}{n}\right) = O(n)}$

**

highest power

$$\frac{n^2 + C \cdot n + (C \cdot I - I^2)}{2 \cdot C}$$

$$O(n^2)$$

assume:
I = 5

P P P P P P

O(1) O(1) O(1) O(1) O(1) ↳ O(n) ∵ size = 5

more accurate assessment of the strategy by amortd avg case (of push operation)

At runtime, the worst-case RT of a dynamic array occurs when that push op. triggers the $\boxed{\text{resizing}}$.

↳ copying the existing contents to the new, bigger array.

Deriving the Sum of a Geometric Sequence

Initial Term: I

Common Factor: r

Number of Terms: k

$$\boxed{3} \xrightarrow{*2} 6 \xrightarrow{*2} 12 \xrightarrow{*2} 24$$

$$\underbrace{3}_{I = 3\cdot 2^0} + \underbrace{6}_{3\cdot 2^1} + \underbrace{12}_{3\cdot 2^2} + \underbrace{24}_{3\cdot 2^3}$$

$[0, 3] = 4$

terms : 4

$$S_k = \underbrace{I}_{\substack{I\cdot r^0 \\ 1st}} + \underbrace{I\cdot r}_{2nd} + \underbrace{I\cdot r^2}_{3rd} + \underbrace{I\cdot r^3}_{4th} + \cdots + \underbrace{I\cdot r^{k-1}}_{kth}$$

$I\cdot r^{k-2}$

$$r\cdot S_k = I\cdot r + I\cdot r^2 + I\cdot r^3 + I\cdot r^4 + \cdots + I\cdot r^{k-1} + I\cdot r^k$$

$$r\cdot S_k - S_k = (r-1)\cdot S_k = I\cdot r^k - I = I\cdot(r^k - 1) \Rightarrow S_k = \frac{I\cdot(r^k-1)}{r-1}$$

↓ useful for avg. RT of double eng.

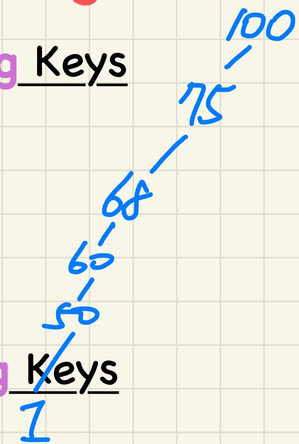
Worst-Case RT: BST with Linear Height

Example 1: Inserted Entries with Decreasing Keys

<100, 75, 68, 60, 50, 1>

key.

$n = 6$



$h = 5$

$(n-1)$

$\boxed{O(n)}$

linear height results in $O(n)$ search, insertion, deletion.

Example 2: Inserted Entries with Increasing Keys

<1, 50, 60, 68, 75, 100>

Exercises.

Example 3: Inserted Entries with In-Between Keys

<1, 100, 50, 75, 60, 68>

$\forall n$. n is internal
difference of heights of n's children ≤ 1

BST + $\boxed{\text{height balance property}}$

$\parallel$

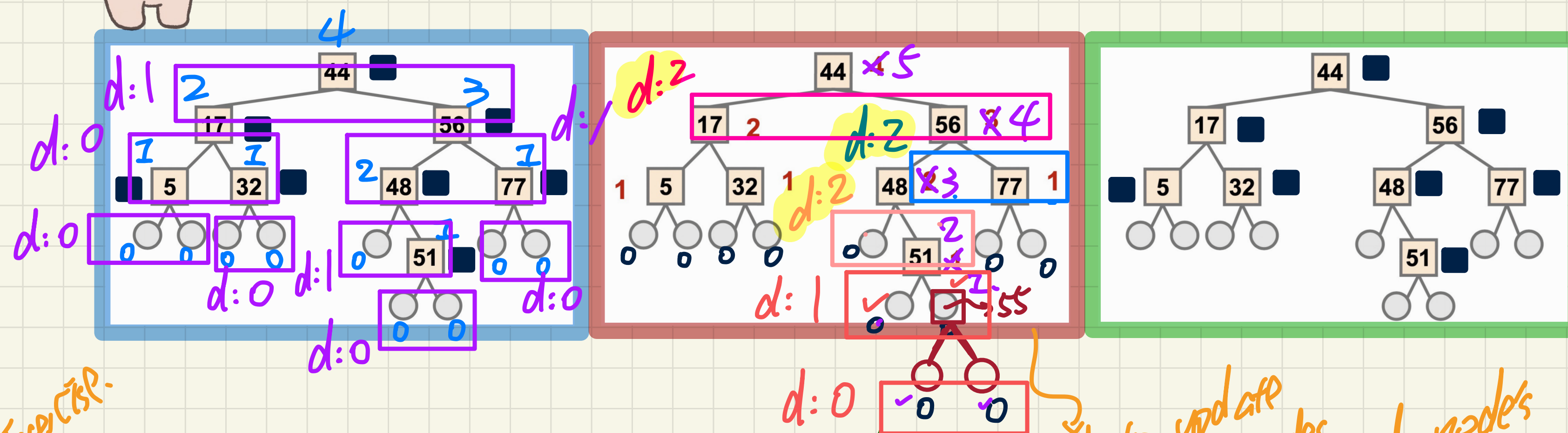
Balanced BST

Balanced BST: Definition

- internal node
- height
- height balance

Given a node p, the ***height*** of the subtree rooted at p is:

$$\textit{height}(p) = \begin{cases} 0 & \text{if } p \text{ is } \textit{external} \\ 1 + \textbf{MAX}\,(\{\ \textit{height}(c) \mid \textit{parent}\ (c) = p\ \}) & \text{if } p \text{ is } \textit{internal} \end{cases}$$



Exercise.

Q. Is the above tree a **balanced BST**? YES.

Q. Still a **balanced BST** after inserting **55**?

Q. Still a **balanced BST** after inserting **63**?

need to update heights of nodes along the inserted nodes' ancestor path