Lecture 13 - Oct 22

Graphs

Test 1 Review

Announcements/Reminders

- Today's class: notes template posted
- Assignment 1 due on Wednesday, October 22
- Test 1 next Monday, October 27:
  + Guide released not yet (Tuesday).
  + Review Session (slides, notes): Wednesday
  + Review Session (A1, more Q&A): Friday
- Tutorial Exercises so far:
  + Tutorial Week 1 (2D arrays)
  + Tutorial Week 2 (2D arrays, Proving Big-O)
  + Tutorial Week 3 (avg case analysis on doubling strategy)
  + Tutorial Week 4 (Trinode restructuring after deletions)

- A1 (Wed).

- 50 minutes Monday, Oct 27.
  4:30pm ~ 5:20pm (50 minutes)
  WSC

- 04 - Graph.pdf
  ↳ up to and including slide 42.

Properties: Structure vs. |V| and |E|

Given $G = (V, G)$ an **undirected** graph with $|V| = n$, $|E| = m$:

$$\begin{cases} m = n-1 & \text{if G is a } \textit{spanning tree} \\ m \le n-1 & \text{if G is a } \textit{forest} \\ m \ge n-1 & \text{if G is } \textit{connected} \\ m \ge n & \text{if G contains a } \textit{cycle} \end{cases}$$

→ G is connected ⇒ $m \geq n-1$ ($|E|$, $|V|$)

connected: every pair of vertices has some path connecting between them

↳ if there is no pair of vertices with no path between → it's connected.

In general, given a graph property,
to prove it: mathematical induction
to disprove it: give a witness graph G' that violates the property

Exercise: show that this is true (via I.H.).

Prove by induction on value of $|V| = n \geq 0$

1. Base Cases

$|V| = 0$ (n), $|E| = 0$ (m): empty graph.
↳ no violation pair of vertices can be found.

$m \geq n-1$: 0, 0

$|V| = 1$, $|E| = 1$ (v, v): $1 \geq (1-1)$

2. I.H.

For a undirected graph with k vertices ($|V| = k$, $k \geq 1$) if graph is connected, then $|E| \geq k-1$

3. Inductive Case.

I.H.: k vertices, $|E| \geq k-1$, $k \geq 1$ & connected

Create a larger graph with k+1 vertices s.t. *

* new graph is connected and $|E'| \geq |V'| - 1$

then k+1

x

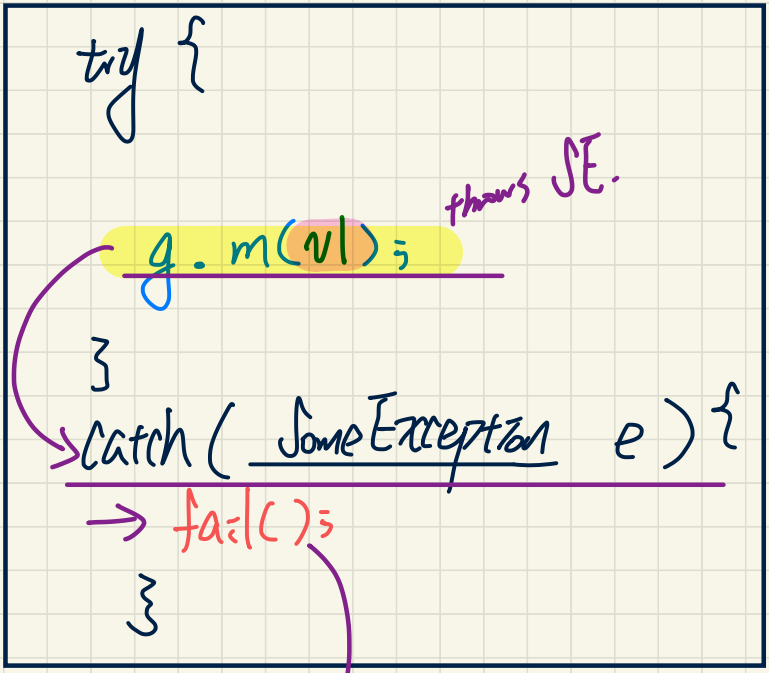
one-vertex graph connected.

say it's connected

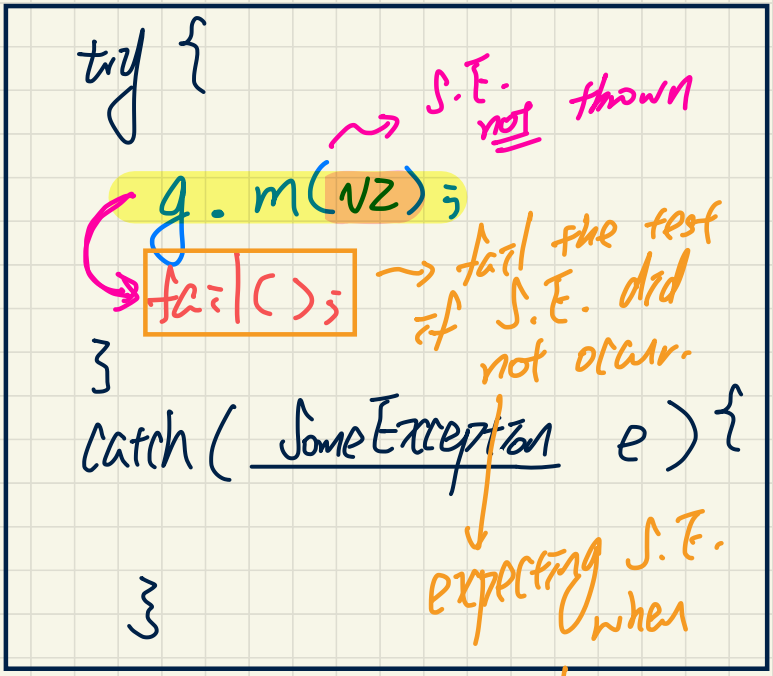
↳ $|V| = 1$

$|E| = 0$

$0 \geq 1 - 1$ ✓.

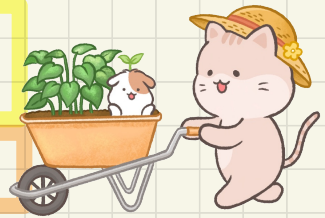


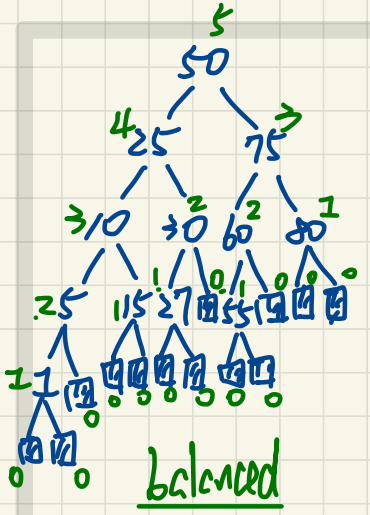
fail the test if SomeException occurred
↳ not expecting S.E. when calling m on v1.

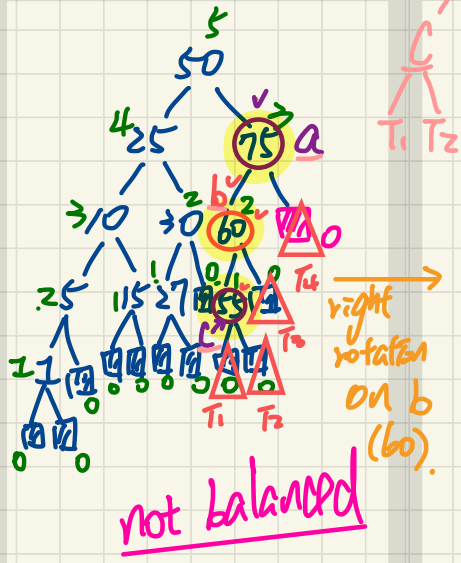


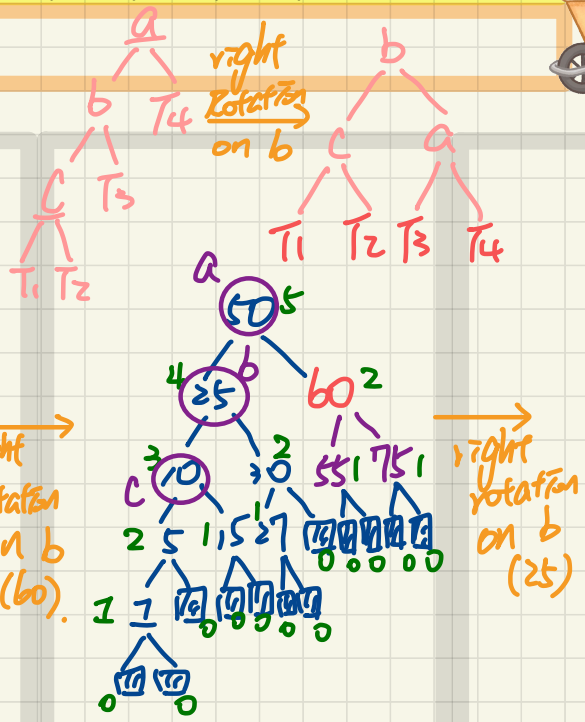
Trinode Restructuring after Deletion: Multiple Rotations

- Insert the following sequence of keys into an empty BST:
  <50, 25, 10, 30, 5, 15, 27, 1, 75, 60, 80, 55>
- Delete 80 from the BST.









After Deletions: Continuous Trinode Restructuring

- *Recall*: ***Deletion*** from a BST results in
  removing a node with zero or one ***internal*** child node.
- After ***deleting*** an existing node, say its child is *n*:
  Case 1: Nodes on *n*'s ***ancestor path*** remain ***balanced***. ⇒ No rotations
  Case 2: At least one of *n*'s ***ancestors*** becomes ***unbalanced***.
  1. Get the **first**/**lowest** ***unbalanced*** node *a* on *n*'s ***ancestor path***.
  2. Get *a*'s ***taller*** child node *b* [$b \notin$ *n*'s ***ancestor path***]
  3. Choose *b*'s child node *c* as follows:
     - *b*'s two child nodes have ***different*** heights ⇒ *c* is the ***taller*** child
     - *b*'s two child nodes have ***same*** height ⇒ *a*, *b*, *c* slant the ***same*** way
  4. Perform rotation(s) based on the ***alignment*** of *a*, *b*, and *c*:
     - Slanted the ***same*** way ⇒ ***single*** *rotation* on the **middle** node *b*
     - Slanted ***different*** ways ⇒ ***double*** *rotations* on the **lower** node *c*
- As *n*'s ***unbalanced ancestors*** are found, keep applying **Case 2**,
  until **Case 1** is satisfied. [$O(h) = O(\log n)$ ***rotations***]