



Quiz 07 Review Session

COMP 210 / 2024 Summer Session I

Ajay Gandecha

LAST Review Session! ✨ 😊 🎉

Other Announcements:

Th 6/14 · Ex11-AVL Trees Due

F 6/14 · Quiz 07 (Final Quiz)

Su 6/16 · Ex12-HashMaps Due ↙ ↘ Last day of Ajay's OH is Mon.

Tu 6/18 · LDOL:

- In Class Presentations ★ ← Start now!
- Ex13: Dijkstra's Algorithm Due ← Last day of Kaki's OH.
- Written Report Due

Sa 6/22


@

8:00AM

COMP 210 Final Exam



Quiz 07 Format

- 30 minutes at the start of class.
 - *On paper* - bring a pencil!
 - Question Types:
 - Multiple choice, T/F, select all that apply, fill in the blank, ~~drawing trees~~
- 



Exercise Check-In Question

- Similar format to the exercise question on the last quiz.
- Review **Ex11** (AVL trees)
- Questions?

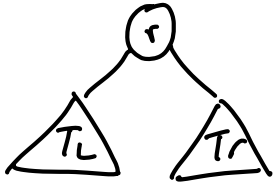
AVL \rightarrow insert: $O(\log N)$
 \rightarrow Rotation: $O(1)$
 \rightarrow Contains: $O(\log N)$



On Quiz 07

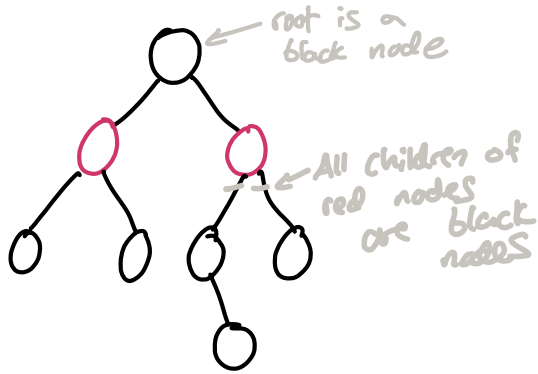
- Red-Black Trees
- Hashing
- ~~Graphs?~~

Red-Black Trees



$$\max(T_L) < r < \min(T_r)$$

↑ BST invariant





Red-Black Trees Time Complexity

Insert: $O(\log N)$

Delete: $O(\log N)$



Red-Black Trees vs AVL Trees

- Red-Black Trees can have a height difference factor of 2.
 - *RBTs require less rotations than AVLTs on average*
 - *Therefore marginally more efficient*

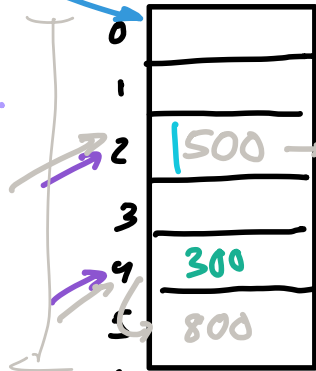
T/F

Hashing ★

COMP 110: Dictionary dict

{
 Key: value,
 Key: value ...
 }

Hash Table



Load factor: $\frac{3}{6} = 50\%$

$\frac{4}{6}$ if we include the chain.

flight-capacities.values()

① Chaining
 "collision strategies"

② Probing

foo → h() → 4

add 500 to "v21"
 add 800 to "foo"

flight-capacities = {
 "S18": 200,
 "A371": 300
 }

flight-capacities["A371"] = 300
 [2] → O(1)



Hash function: $h()$

"A371" → h() → 4
 ↑ key ↑ hash function

Collision

"v21" → h() → 2
 "S18" → h() → 2

Hashing

- Can have collisions.
 - To lower the chance of collisions, we can resize our hash tables.
 - Depends on the load factor (# items in table / # spots in table) 
i.e.
stored values! 

Collision Resolution

- Chaining

- Each item in the hash table is a *linked list*.
- If we try to put a value at an index and it is taken, we add to the front of the list.
- Get: $O(1)$
- Put: $O(1)$ average, $O(N)$ w.c. for resize.
- Remove: $O(1)$

Given:
val : "Kris", "Kati", "KMP", "Syed"
hash: 0 1 2 1

Q: Insert given values using the chaining collision strategy.

• 0 Kris
• 1 Syed → Kati *
• 2 KMP
• 3 _____
• 4 _____

Collision Resolution

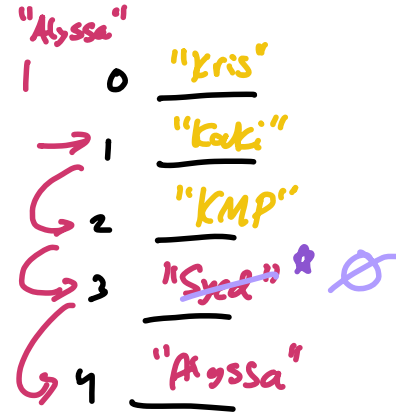
- Probing

- Each item in the hash table is max 1 value.
- If we try to put a value at an index and it is taken, we try to place it in the next available spot.
- Get: $O(1)$
- Put: $O(1)$ average, $O(N)$ w.c. for resize.
- Remove: $O(1)$

Given:

val : "Kris", "Kati", "KMP", "Syed"
hash: 0 1 2 1

Q: Insert given values using the probing collision strategy.



If you delete "Syed", the value would be replaced with ∅ ← contributes to the load factor