PSO 6

(capitalization matters for some reason – curse you web devs!!)

→ justin-zhang.com/teaching/CS251

Question 1

If N keys are inserted into an initially empty BST, how many different (unlabelled binary search) tree shapes are possible if

(a) $N = 2$? - 2 different tree shapes exist

(b) $N = 3$? - 5 different tree shapes exist

(c) $N = 4$? - 14 different tree shapes exist

(d) $N = 5$? - 42 different tree shapes exist

Justify your answers.

Question 2

(1) What is the asymptotic performance of inserting n items with keys sorted in a descending order into an initially empty binary search tree?

(2) Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

(b) Can another insertion order give the same state? Explain your answer.

(c) What is the load factor of T? Is there any issue occurring in T?

(d) Illustrate T if the collision management technique used was chaining.

Question 4

You are in the role of a hacker trying to break down a hash table. The information collected so far indicates the hash table uses Quadratic Probing with $h(k, i) = (k + i^2) \bmod m$ for collision management and its current capacity is $m = 9$. The current state of the table is:

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

The system is nearly overloaded and will collapse if the next item inserted causes at least 4 probes. As an attacker you are considering inserting the following keys: 16, 35 and 10. Which (if any) of these values would bring the system down if inserted next? Explain your answer.

Question 5

(a) Show how Mergesort works on the array $[M, E, R, G, E, S, O, R, T]$.

(b) What is the expected runtime complexity for Quicksort running on $[7, 5, 3, 1, 2, 4, 6]$ when always using the rightmost index of each partition as the pivot? Explain your answer.

Question 1

If N keys are inserted into an initially empty BST, how many different (unlabelled binary search) tree shapes are possible if

(a) N = 2? - 2 different tree shapes exist

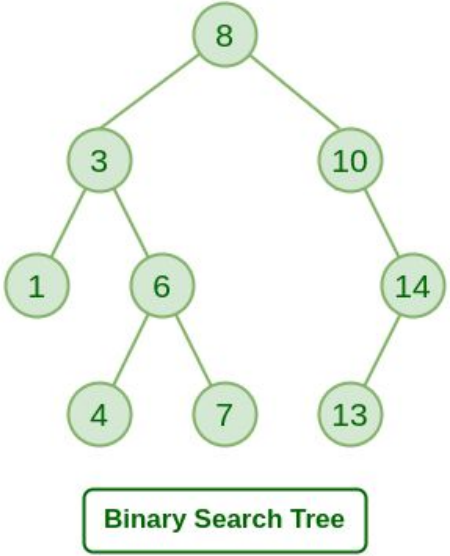
(b) N = 3? - 5 different tree shapes exist

(c) N = 4? - 14 different tree shapes exist

(d) N = 5? - 42 different tree shapes exist

Justify your answers.

What is a BST?



Each node in the tree has

node.left <= node.val <= node.right

Question 1

If N keys are inserted into an initially empty BST, how many different (unlabelled binary search) tree shapes are possible if

(a) $N = 2$? - 2 different tree shapes exist

(b) $N = 3$? - 5 different tree shapes exist

(c) $N = 4$? - 14 different tree shapes exist

(d) $N = 5$? - 42 different tree shapes exist

Justify your answers.

Counting BSTs!

List all the N=2,3 trees

Counting BSTs

We use a recurrence to calculate work.. We can do the same for counting

Let B_i be the number of BSTs on i nodes

Base Cases?

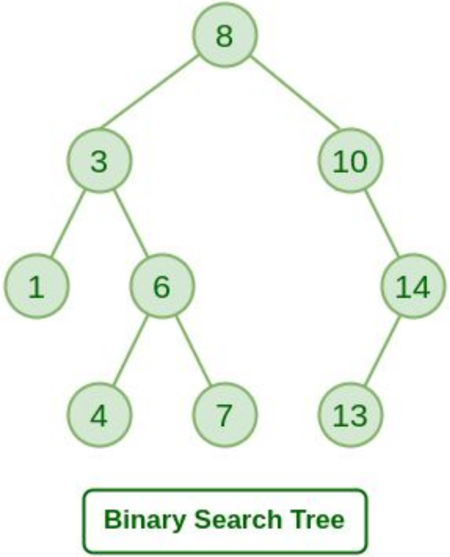
Let Bi be the number of BSTs on i nodes

$B_0 = 1$

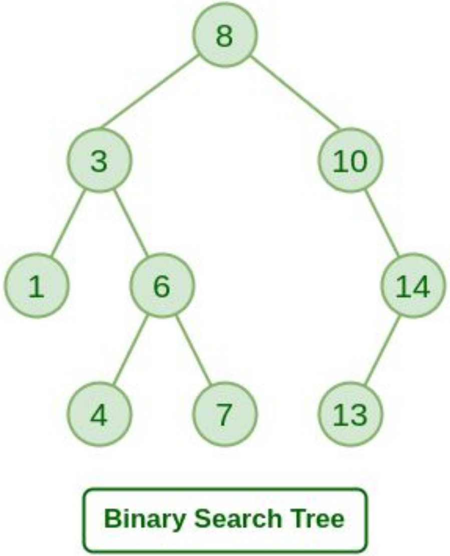
$B_1 = 1$

General Case?

$B_n = \dots$



Let Bi be the number of BSTs on i nodes



$B_0 = 1$

$B_1 = 1$

General Case?

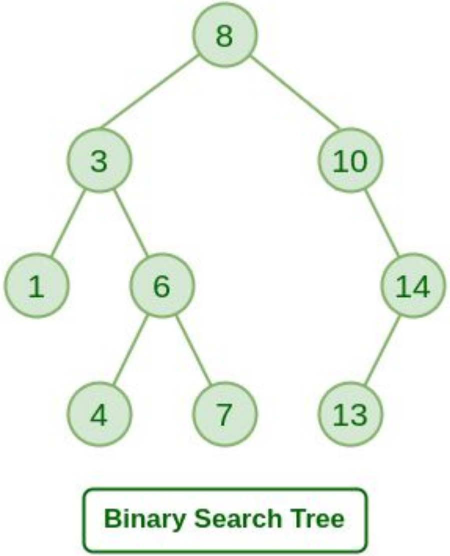
$B_n =$

Insight: subtrees of BSTs are also BSTs

Q: Suppose I know there are i nodes in the left subtree and (n - i -1) nodes in the right subtree

How many BSTs?

Let Bi be the number of BSTs on i nodes



$B_0 = 1$

$B_1 = 1$

General Case?

$B_n =$

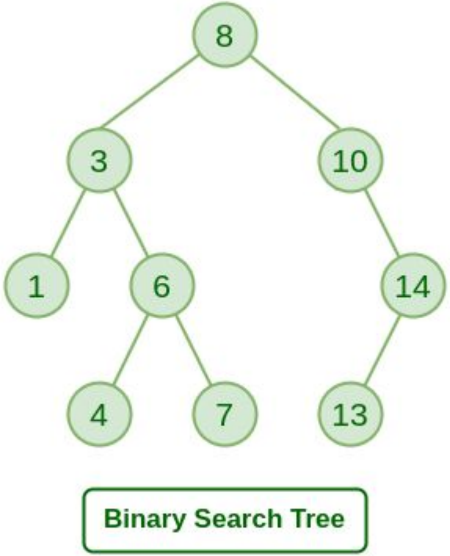
Insight: subtrees of BSTs are also BSTs

Q: Suppose I know there are i nodes in the left subtree and (n - i -1) nodes in the right subtree

How many BSTs?

A: $B_i B_{n-i-1}$

Let Bi be the number of BSTs on i nodes



$B_0 = 1$

$B_1 = 1$

General Case?

$B_n = \sum_{i = 1}^{n-1} B_i B_{n-i-1}$

Insight: subtrees of BSTs are also BSTs

Q: Suppose I know there are i nodes in the left subtree and (n - i -1) nodes in the right subtree

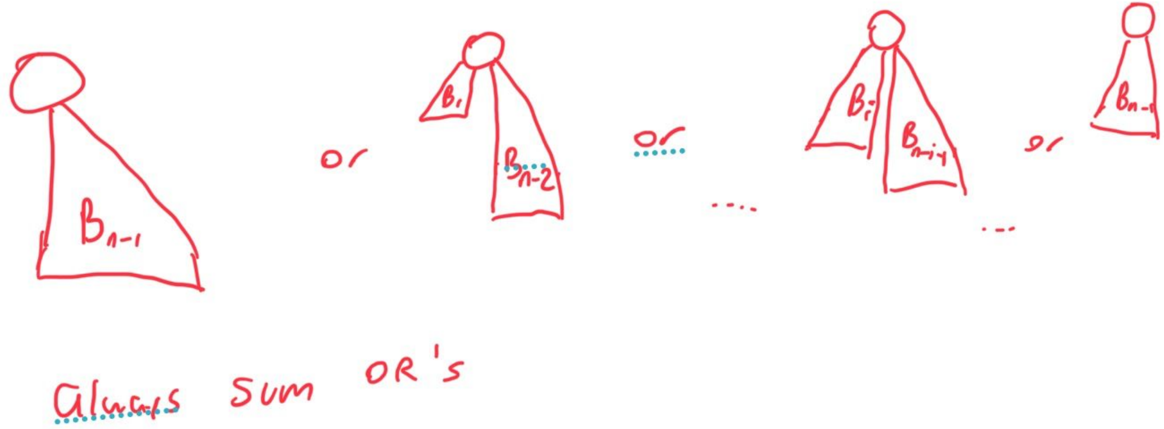
How many BSTs?

A: $B_i B_{n-i-1}$

Sum over all possible values of i

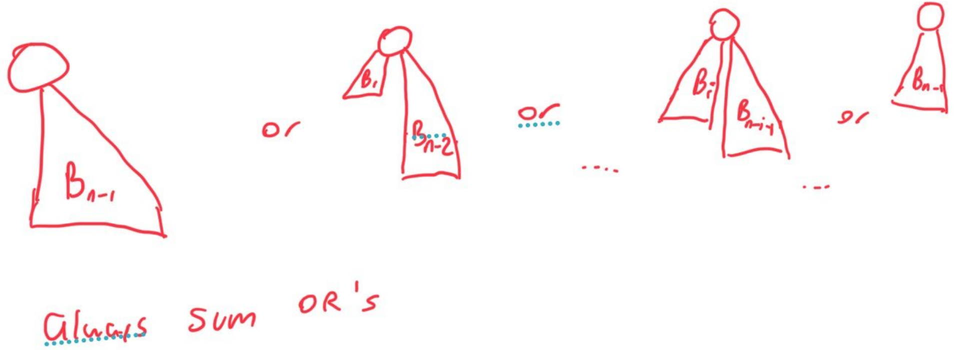
Summary of How we counted

1. **Recurrence:** We set B_n = # bsts with n nodes
2. **Base Case:** B_0,B_1= 1
3. **Recursive Case (B_n):**
   a. A root node can have i left children and (n - i - 1) right children



Summary of How we counted

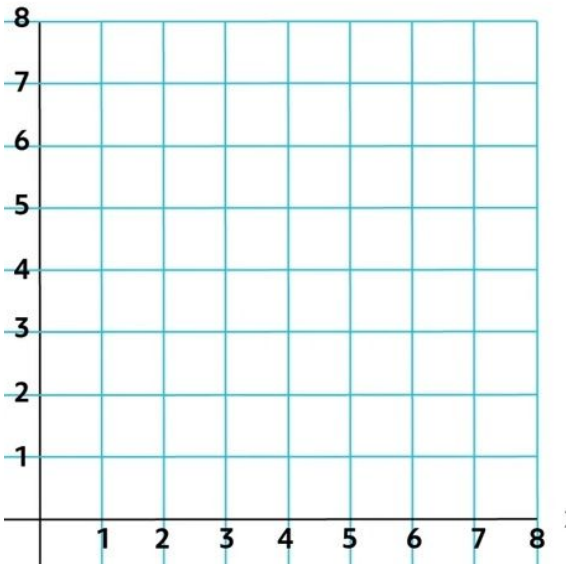
1. **Recurrence:** We set B_n = # bsts with n nodes
2. **Base Case:** $B_0, B_1 = 1$
3. **Recursive Case (B_n):**
    a. A root node can have i left children and (n - i - 1) right children
    b. If i left children, there are $B_i B_{n-i-1}$ possible BSTs



Bonus: A related problem

I am taking a walk on a graph, where I can only go right or up.

How many paths are there from (0,0) to (n,n) where I never go under the diagonal?



Question 2

(1) What is the asymptotic performance of inserting n items with keys sorted in a descending order into an initially empty binary search tree?

(2) Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

```
Insert(root,x):
    If root == null: return x
    If (x <= root.val): insert(root.left,x)
    If (x > root.val): insert(root.right,x)
```

Question 2

(1) What is the asymptotic performance of inserting n items with keys sorted in a descending order into an initially empty binary search tree?

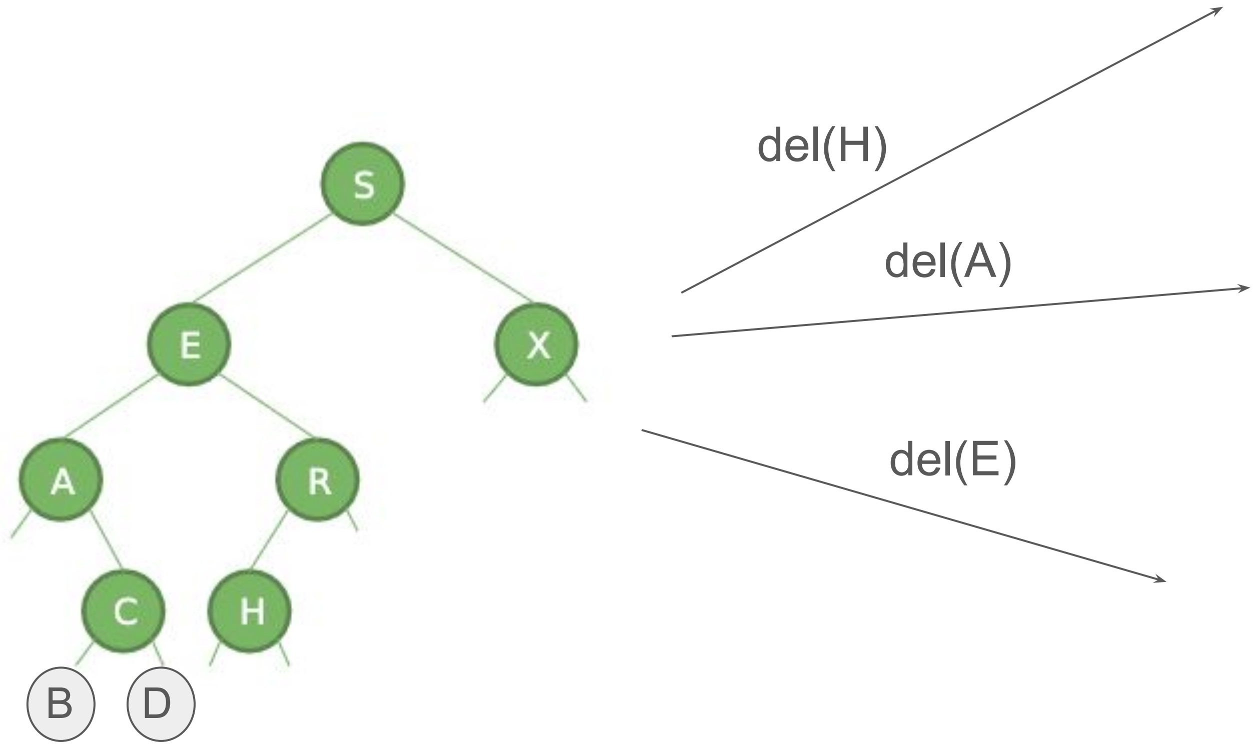
(2) Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

How does deletion work?

Deletion in a BST: Depends on # children

Basically, want to delete while keeping order



Question 2

(1) What is the asymptotic performance of inserting n items with keys sorted in a descending order into an initially empty binary search tree?

(2) Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

Assume 1 child deletion swaps with **successor**

Delete A, then B	A / \\ B D / C
Delete B, then A	A / \\ B D / C

Question 2

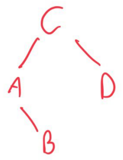
(1) What is the asymptotic performance of inserting n items with keys sorted in a descending order into an initially empty binary search tree?

(2) Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

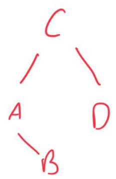
(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

If 1 child deletion swaps with **predecessor**

delete C, then delete D



delete D, then delete C



Question 2

(3) Your friend thinks he has discovered a remarkable property of binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. Your friend claims that any three keys $a \in A$, $b \in B$, and $c \in C$ must satisfy $a \leq b \leq c$. Give a simple counterexample to his claim.

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

(b) Can another insertion order give the same state? Explain your answer.

(c) What is the load factor of T? Is there any issue occurring in T?

(d) Illustrate T if the collision management technique used was chaining.

Linear Probing: If collision, check next box

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	
17	
28	
28	
8	
31	

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17			8			

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28		8			

Next, 28

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28	18	8			

Next, 28, 18

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28	18	8	31		

Next, 28, 18, 31

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

(b) Can another insertion order give the same state? Explain your answer.

(c) What is the load factor of T? Is there any issue occurring in T?

(d) Illustrate T if the collision management technique used was chaining.

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17			8			

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28		8			

Next, 28

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28	18	8			

Next, 28, 18

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

	4	5	6	7	8	9	10	11
	16	17	28	18	8	31		

Next, 28, 18, 31

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

Which ones are in the right place?

Insert 16,17,8 first

4	5	6	7	8	9	10	11
16	17	28	18	8	31		

Next, 28, 18, 31

I can enter 16,17,8 in any order

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

(b) Can another insertion order give the same state? Explain your answer.

(c) What is the load factor of T? Is there any issue occurring in T?

(d) Illustrate T if the collision management technique used was chaining.

Load factor =

Question 3

(Hash table) Let T be an empty hash table of length $m = 12$ with $h(k) = k \bmod 12$, $k \in \mathbb{Z}^+$. T uses linear probing as a collision management technique. The following is the content of T after inserting six values.

0	1	2	3	4	5	6	7	8	9	10	11
				16	17	28	18	8	31		

(a) Write an order of insertion for these six values such that the state of T is the one displayed above.

(b) Can another insertion order give the same state? Explain your answer.

(c) What is the load factor of T? Is there any issue occurring in T?

(d) Illustrate T if the collision management technique used was chaining.

(d) Illustrate T if the collision management technique used was chaining.

Insertion order: 16, 17, 28, 18, 8, 31

k	h(k) = k mod 12
16	4
17	5
28	4
18	6
8	8
31	7

	4	5	6	7	8	9	10	11

Question 4

You are in the role of a hacker trying to break down a hash table. The information collected so far indicates the hash table uses Quadratic Probing with $h(k, i) = (k + i^2) \bmod m$ for collision management and its current capacity is $m = 9$. The current state of the table is:

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

The system is nearly overloaded and will collapse if the next item inserted causes at least 4 probes. As an attacker you are considering inserting the following keys: 16, 35 and 10. Which (if any) of these values would bring the system down if inserted next? Explain your answer.

Quadratic probing:

i = i'th collision

Trying 16

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

h(16,0) = 16 + 0^2 mod 9 = 7

No collision

Trying 35

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(35,0) = 35 + 0^2 \bmod 9 =$

Trying 35

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

h(35,0) = 35 + 0^2 mod 9 = 8 **Collision**

h(35,1) = 35 + 1 mod 9 =

Trying 35

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(35,0) = 35 + 0^2 \bmod 9 = 8$ **Collision**

$h(35,1) = 35 + 1^2 \bmod 9 = 0$ **Collision**

$h(35,2) = 35 + 2^2 \bmod 9 =$

Trying 35

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

h(35,0) = 35 + 0^2 mod 9 = 8 **Collision**

h(35,1) = 35 + 1^2 mod 9 = 0 **Collision**

h(35,2) = 35 + 2^2 mod 9 = 3 **No Collision**

Trying 10

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(10,0) = 10 + 0^2 \bmod 9 =$

Trying 10

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(10,0) = 10 + 0^2 \bmod 9 = 1$ **Collision**

$h(10,1) = 10 + 1^2 \bmod 9 =$

Trying 10

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(10,0) = 10 + 0^2 \bmod 9 = 1$ **Collision**

$h(10,1) = 10 + 1^2 \bmod 9 = 2$ **Collision**

$h(10,2) = 10 + 2^2 \bmod 9 =$

Trying 10

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(10,0) = 10 + 0^2 \bmod 9 = 1$ **Collision**

$h(10,1) = 10 + 1^2 \bmod 9 = 2$ **Collision**

$h(10,2) = 10 + 2^2 \bmod 9 = 5$ **Collision**

$h(10,3) = 10 + 3^2 \bmod 9 =$

Trying 10

0	1	2	3	4	5	6	7	8
17	28	20			5	32		19

$h(10,0) = 10 + 0^2 \bmod 9 = 1$ **Collision**

$h(10,1) = 10 + 1^2 \bmod 9 = 2$ **Collision**

$h(10,2) = 10 + 2^2 \bmod 9 = 5$ **Collision**

$h(10,3) = 10 + 3^2 \bmod 9 = 1$ **Collision**

Question 5

(a) Show how Mergesort works on the array $[M, E, R, G, E, S, O, R, T]$.

(b) What is the expected runtime complexity for Quicksort running on $[7, 5, 3, 1, 2, 4, 6]$ when always using the rightmost index of each partition as the pivot? Explain your answer.

Merge sort: Divide until pairs/singles, then recombine

Question 5

(a) Show how Mergesort works on the array $[M, E, R, G, E, S, O, R, T]$.

(b) What is the expected runtime complexity for Quicksort running on $[7, 5, 3, 1, 2, 4, 6]$ when always using the rightmost index of each partition as the pivot? Explain your answer.

Quick Sort: Sort by pivot partitioning