

Exercise walk-through

8.2

Binary search trees

International Baccalaureate · IB Diploma · Computer Science · HL

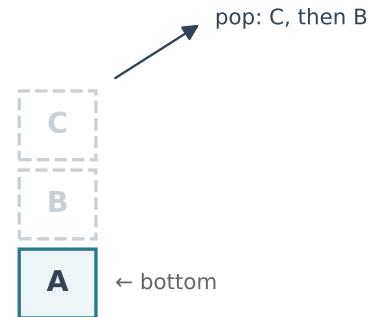
Adt · read the diagram

Queue (FIFO): enqueue A, B, C; dequeue twice; enqueue D



A and B leave from the front, in arrival order.
C, then D, are still waiting.

Stack (LIFO): push A, B, C; pop twice



The last item pushed is the first removed.

- Choose the structure for the required access pattern.
- A queue removes the oldest item first; a stack removes the newest item first

Adt · executable trace

- Trace the state changes and boundary guard.
- Run the boundary case and compare the printed output.

```
from collections import deque
queue = deque()
queue.extend(["A", "B"])
print(queue.popleft())
queue.append("C")
print(queue.popleft())
print(list(queue))
queue.clear()
print(queue.popleft() if queue else "EMPTY")
```

Exercises 2.1-2.3

2.1 Draw the BST made by inserting 8, 3, 10, 1, 6, 14, 4, 7. [3]

2.2 For your tree in 2.1, state the root and the leaves. [2]

2.3 For your tree in 2.1, give the in-order and pre-order traversals. [2]

Exercises 2.1-2.3

2.1 Draw the BST made by inserting 8, 3, 10, 1, 6, 14, 4, 7. [3]

2.1

- root 8, with 3 left and 10 right
- 1 and 6 below 3; 14 right of 10
- 4 left and 7 right of 6

2.2 For your tree in 2.1, state the root and the leaves. [2]

2.2

- root **8**
- leaves **1, 4, 7, 14**

2.3 For your tree in 2.1, give the in-order and pre-order traversals. [2]

2.3

- in-order **1, 3, 4, 6, 7, 8, 10, 14**
- pre-order **8, 3, 1, 6, 4, 7, 10, 14**

Exercises 2.4·3.1

2.4 State the keys compared when searching your tree for 7.

[1]

3.1 Which traversal of a BST outputs its keys in ascending order?

[1]

☐

A pre-order

B in-order

C post-order

D none of them

Exercises 2.4-3.1

2.4 State the keys compared when searching your tree for 7. [1]

2.4

- 8, 3, 6, 7

3.1 Which traversal of a BST outputs its keys in ascending order? [1]

☐

A pre-order

B in-order

C post-order

D none of them

3.1 B

- left, node, right gives ascending order

Exercises 3.2-3.3

3.2 A BST holds 1023 keys and is perfectly balanced. What is the greatest number of comparisons needed to find a key? [1]

☐

A 10

B 11

C 512

D 1023

3.3 A post-order traversal of a BST is 5, 15, 10, 30, 20. Which is the root? [1]

☐

A 5

B 10

C 15

D 20

Exercises 3.2-3.3

3.2 A BST holds 1023 keys and is perfectly balanced. What is the greatest number of comparisons needed to find a key? [1]

☐

- A 10
- B 11
- C 512
- D 1023

3.2 A

- $1023 = 2^{10} - 1$: 10 levels

3.3 A post-order traversal of a BST is 5, 15, 10, 30, 20. Which is the root? [1]

☐

- A 5
- B 10
- C 15
- D 20

3.3 D

- post-order visits the root last

Exercise 3.4 (a)

3.4 A dictionary app stores words in a BST in alphabetical order.

(a) Draw the BST made by inserting: lion, deer, tiger, ant, fox, zebra, owl. [3]

Exercise 3.4 (a)

3.4 A dictionary app stores words in a BST in alphabetical order.

(a) Draw the BST made by inserting: lion, deer, tiger, ant, fox, zebra, owl. [3]

(a)

- root lion, with deer left and tiger right
- ant and fox below deer
- owl left and zebra right of tiger

Exercise 3.4 (b)

3.4 A dictionary app stores words in a BST in alphabetical order.

(b) Give the post-order traversal of your tree.

[2]

Exercise 3.4 (b)

3.4 A dictionary app stores words in a BST in alphabetical order.

(b) Give the post-order traversal of your tree.

[2]

(b)

- ant, fox, deer, owl, zebra, tiger, lion

Exercise 3.4 (c)

3.4 A dictionary app stores words in a BST in alphabetical order.

(c) State the words compared when searching for `goat`, and the result.

[2]

Exercise 3.4 (c)

3.4 A dictionary app stores words in a BST in alphabetical order.

(c) State the words compared when searching for **goat**, and the result.

[2]

(c)

- lion, deer, fox
- the right of fox is empty: **not found**

Exercise 3.4 (d)

3.4 A dictionary app stores words in a BST in alphabetical order.

(d) The words are later loaded from a file that is already in alphabetical order. Explain the effect on search time, and suggest a fix. [3]

Exercise 3.4 (d)

3.4 A dictionary app stores words in a BST in alphabetical order.

(d) The words are later loaded from a file that is already in alphabetical order. Explain the effect on search time, and suggest a fix. [3]

(d)

- each word is larger than the last, so each goes right: the tree is one chain
- search becomes $O(n)$ instead of $O(\log n)$
- fix: shuffle the words before inserting, or build the tree from the middle word outwards (or use a self-balancing tree)