

8.2 Binary search trees

Name: _____ Class: _____ Date: _____ Total: 21 marks

KEY WORDS root • leaf • binary search tree • pre-order
• in-order • post-order

Objective

Build the skills to answer HL exam questions on **binary search trees** – inserting keys, searching, the three traversals and the effect of balance.

You must be able to:

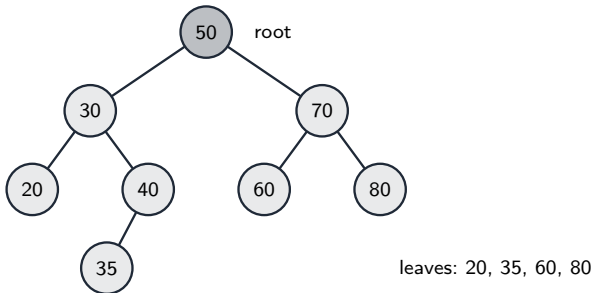
- use the terms **root** , **leaf** , child and subtree
- build a **binary search tree** by inserting keys in a given order
- search a BST, giving each node compared
- give the **pre-order** , **in-order** and **post-order** traversals
- explain why an unbalanced tree is slow to search

1 Key knowledge and worked examples

Study these first. Each one shows the method for a question type used later.

■ Worked example: building a BST

Insert 50, 30, 70, 20, 40, 60, 80, 35 in that order.



1. 50 is the root.

2. Each new key starts at the root: smaller goes left, larger goes right, until an empty place.
3. 35: 35 < 50 left; 35 > 30 right; 35 < 40 left; empty: the left child of 40.

Searching for 35 compares 50, 30, 40, 35: 4 comparisons. Searching for 65 compares 50, 70, 60, then finds the right of 60 empty: not found.

■ Key knowledge: traversals

Traversal	Visit order	For the tree above
pre-order	node, left, right	50, 30, 20, 40, 35, 70, 60, 80
in-order	left, node, right	20, 30, 35, 40, 50, 60, 70, 80
post-order	left, right, node	20, 35, 40, 30, 60, 80, 70, 50

- In-order gives a BST’s keys in **sorted** order.
- Pre-order rebuilds the tree; post-order deletes it, children first.

```
def in_order(node):
    if node is not None:
        in_order(node.left)
        print(node.key)
        in_order(node.right)
```

■ Key knowledge: balance

- A **balanced** tree of n keys has about $\log_2 n$ levels: search is $O(\log n)$.
- Keys inserted in sorted order make one long chain: search is $O(n)$.

2 Practice

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]

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

3 Exam-style questions

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

3.2 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

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]

(b) Give the post-order traversal of your tree. [2]

(c) State the words compared when searching for **goat**, and the result. [2]

(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]

Solutions

Each answer shows the steps, then the **result**.

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

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

2.3

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

2.4

- 8, 3, 6, 7

3.1 B

- left, node, right gives ascending order

3.2 A

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

3.3 D

- post-order visits the root last

3.4

(a)

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

(b)

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

(c)

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

(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)