INTERNATIONAL BACCALAUREATE · IB DIPLOMA ·
COMPUTER SCIENCE · HL • EXERCISE SHEET

6.3 Searching, sorting and Big O

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

KEY WORDS linear search • binary search • bubble sort • selection sort • Big O O

Objective

Build the skills to answer exam questions on **searching, sorting and efficiency** – linear and binary search, bubble and selection sort, and Big O notation.

You must be able to:

- trace a **linear search** and a **binary search**, giving each value compared
- state why binary search needs a sorted list
- show a list after each pass of a **bubble sort** and a **selection sort**
- state the **Big O** O of each algorithm and compare their efficiency

1 Key knowledge and worked examples

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

■ Key knowledge: Big O

Big O	Example	Steps for $n = 1000$
$O(1)$	reading `a[5]`	1
$O(\log n)$	binary search	about 10
$O(n)$	linear search	1000
$O(n^2)$	bubble sort, selection sort	about 1 000 000

- Big O gives the worst case as n grows. One loop over the data: $O(n)$; a loop inside a loop: $O(n^2)$.

■ Worked example: binary search

```
low, high = 0, len(a) - 1
while low <= high:
    mid = (low + high) // 2
    if a[mid] == target:
        return mid
    elif a[mid] < target:
        low = mid + 1
    else:
        high = mid - 1
return -1
```

`low`	`high`	`mid`	`a[mid]`	Action
0	8	4	16	$16 < 23$: `low` $= 5$
5	8	6	38	$38 > 23$: `high` $= 5$
5	5	5	23	found: return 5

Three comparisons; a linear search would need six. On an unsorted list the discarded half might hold the target.

■ Worked example: two sorts

Sort `[7, 3, 9, 1]` into ascending order.

After pass	Bubble sort	Selection sort
1	`[3, 7, 1, 9]`	`[1, 3, 9, 7]`
2	`[3, 1, 7, 9]`	`[1, 3, 9, 7]`
3	`[1, 3, 7, 9]`	`[1, 3, 7, 9]`

1. **Bubble**: compare neighbours, swap if out of order. Pass 1 moves 9 to the end.
2. **Selection**: find the smallest of the unsorted part and swap it into place. Pass 1 swaps 1 with 7.
3. Both are $O(n^2)$. A bubble sort with a `swapped` flag stops after a pass with no swaps.

2 Practice

2.1 List the values compared when a linear search looks for 40 in `[12, 7, 40, 3, 25]`, and the number of comparisons if it looks for 99. [2]

2.2 Show `[5, 2, 8, 1, 4]` after each of the first **two** passes of a bubble sort. [2]

2.3 Show `[5, 2, 8, 1, 4]` after each of the first **two** passes of a selection sort. [2]

2.4 A sorted list has 1000 items. State the greatest number of items compared by (i) a linear search; (ii) a binary search. [2]

3 Exam-style questions

3.1 Which statement about binary search is true? [1]

- **A** it works on any list
- **B** it checks every item in turn
- **C** it needs the list to be sorted
- **D** it is $O(n^2)$

3.2 Which is the Big O of a selection sort? [1]

- **A** $O(1)$
- **B** $O(\log n)$
- **C** $O(n)$
- **D** $O(n^2)$

3.3 A bubble sort makes a pass with no swaps. What can be concluded? [1]

- **A** the list is sorted
- **B** the list has duplicates
- **C** one more pass is needed
- **D** the list is in reverse order

3.4 A library keeps a sorted list of book codes:

`[104, 117, 131, 148, 152, 166, 179, 183, 190, 205, 218]`

(a) Trace a binary search for 190, giving `low`, `high`, `mid` and `a[mid]` at each probe. [4]

(b) State how many items a linear search would compare to find 190. [1]

(c) A new code, 140, is added to the end of the list without sorting it again. Explain why a binary search for 140 now fails. [2]

(d) The library has 50 000 books and searches the list many times a day but adds books rarely. Recommend linear or binary search and justify your choice. [3]

Solutions

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

2.1

- 12, 7, **40**
- **5** comparisons: every item

2.2

- pass 1: `[2, 5, 1, 4, 8]`
- pass 2: `[2, 1, 4, 5, 8]`

2.3

- pass 1: 1 swaps with 5: `[1, 2, 8, 5, 4]`
- pass 2: 2 is already in place: `[1, 2, 8, 5, 4]`

2.4

- (i) **1000**
- (ii) **10**, since $2^{10} = 1024 \geq 1000$

3.1 C

- it discards the half that cannot hold the target

3.2 D

- a loop inside a loop

3.3 A

- every neighbouring pair is already in order

3.4

(a)

`low`	`high`	`mid`	`a[mid]`
0	10	5	166
6	10	8	190

- first probe correct: `mid` 5, 166
- $166 < 190$, so `low` $= 6$
- second probe: `mid` $(6 + 10)//2 = 8$
- 190 found at index **8**

(b)

- **9**

(c)

- the list is no longer sorted
- the first probe finds $166 > 140$ and discards the right half, which holds 140, so it returns not found

(d)

- **binary** search
- it is $O(\log n)$: about 16 comparisons for 50 000 items, against up to 50 000 for linear
- the list must be kept sorted, which is cheap as books are added rarely