Linear and binary search

Algorithms and programming (Paper 2) · revision notes

A linear search checks each item in turn; a binary search repeatedly halves a sorted list. You need to know the steps of each, that binary search needs sorted data, and be able to apply each search to a list and recognise it from code. Expect 'describe the steps' and 'compare the two searches' questions.

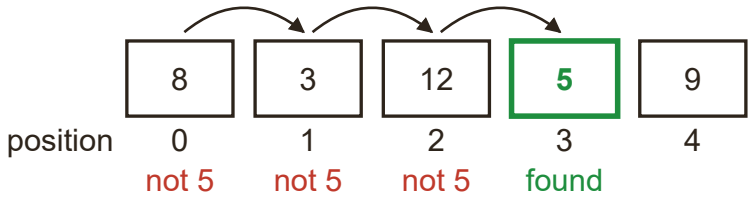
Key facts

- Linear search: check each item in turn from the start until found or the end is reached
- Binary search: the data MUST be sorted
- Target larger than middle: first = mid + 1; smaller: last = mid − 1
- Binary search halves the list with every comparison
- Linear search works on unsorted data
- Middle position = (first + last) DIV 2
- Not found when first > last (nothing left to search)
- 1000 items: binary search needs at most 10 comparisons; linear search up to 1000

Notes

Linear search

- Start at the first item and compare it with the item you are looking for (the target).
- If it is not a match, move on to the next item. Repeat until the target is found or the end of the list is reached.

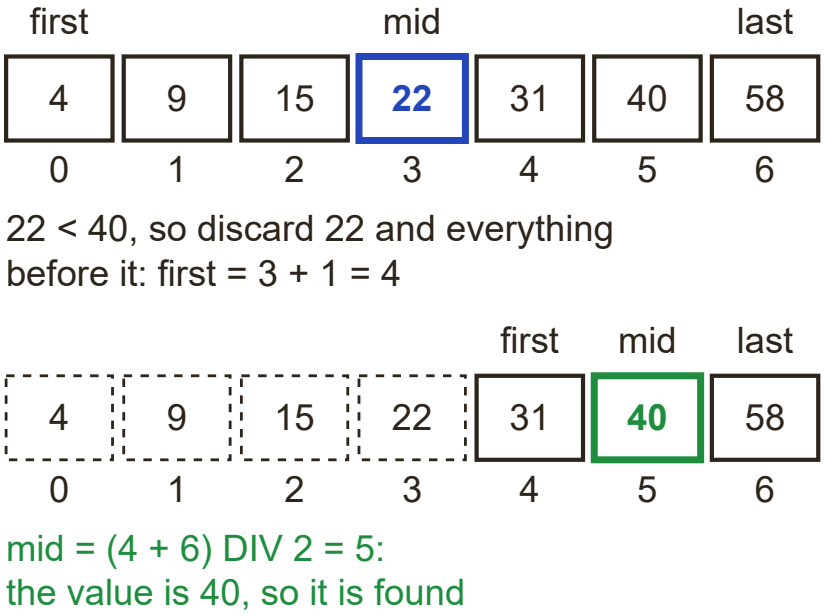


Linear search for 5: each item is checked in turn until it is found

- It works on **unsorted** or sorted data and is simple to code.
- It is slow for large lists: in the worst case every item is checked, so a list of 1000 items can need 1000 comparisons.

Binary search

- The data **must be sorted** first.
- Find the middle item. In code the middle position is usually `mid = (first + last) DIV 2`, which rounds down.
- If the middle item is the target, stop: it has been found.
- If the target is smaller, discard the middle item and everything after it (`last = mid - 1`). If it is larger, discard the middle item and everything before it (`first = mid + 1`).
- Repeat with the half that is left, until the target is found or there is nothing left to search (first is greater than last), which means it is not in the list.
- Example: find 40 in 4, 9, 15, 22, 31, 40, 58 (positions 0 to 6). Middle is position 3, value 22. 40 is larger, so search positions 4 to 6. Middle is position 5, value 40: found after 2 comparisons.



Binary search for 40: each comparison discards half of what is left

Which is better?

- Each comparison in a binary search halves the part of the list still to search, so it is much faster on large lists: at most 10 comparisons for 1000 items.
- A linear search is better when the list is unsorted (sorting first takes time), very small, or when the target is likely to be near the start.

How to answer each type of question

Describe the steps of a search on given data

3 to 4 marks | Grade 5

1. For binary search, write down the positions and find the middle item using the rule given.
2. Say what it is compared with and which half is discarded.
3. Repeat with the new middle item until the target is found (or the list is empty).
4. Name the actual values you compare: that is where the marks are.

Example. The sorted list 2, 5, 8, 11, 16, 20, 27, 30, 35 is stored in positions 0 to 8. Describe how a binary search would find 30. Use (first + last) DIV 2 for the middle position. [4]

Model answer. Compare 30 with the middle item, 16 at position 4 (1). 30 is larger, so discard 16 and every item before it (1). Compare 30 with the new middle item, 27 at position 6; 30 is larger, so discard 27 and the items before it (1). Compare 30 with the middle of positions 7 to 8, which is 30 at position 7: found, so stop (1).

Compare linear and binary search

2 to 4 marks | Grade 6

1. Make a point about efficiency: binary search halves the list each time.
2. Link it to the size of the data in the question.
3. For an advantage of linear search, mention unsorted data or very small lists.

Example. A phone stores 500 contacts in alphabetical order.
Explain why a binary search is more efficient than a linear search for finding a contact. [2]

Model answer. A binary search halves the number of contacts left to search with each comparison (1), so it needs at most 9 comparisons, whereas a linear search could need 500 (1).

List the items checked in a linear search

1 to 2 marks | Grade 4

1. Start at the first item.
2. List every item in order up to and including the target.
3. If the target is not there, every item is checked.

Example. A linear search is used to look for 9 in the list 7, 3, 12, 9, 5.
(a) State the items that are compared with 9, in order. [1]
(b) Explain why a binary search could not be used on this list. [1]

Model answer.

(a) 7, 3, 12, 9 (1)
(b) The list is not sorted (1).

Complete or identify search code

3 to 4 marks | Grade 7

1. Look for clues: a mid variable and DIV 2 means binary search; checking every item with a loop from 0 means linear search.
2. For binary search, the target is larger: first moves to mid + 1; smaller: last moves to mid − 1.
3. A value such as −1 is often returned to mean 'not found'.

Example. The function returns the position of `target` in the sorted array `items`, or −1 if it is not there.

```
function search(items, target)
   first = 0
   last = items.length - 1
   while first <= last
      mid = (first + last) DIV 2
      if items[mid] == target then
         return mid
      elseif items[mid] < target then
         first = ....(1)....
      else
         last = ....(2)....
      endif
   endwhile
   return ....(3)....
endfunction
```

(a) Complete the gaps (1) to (3). [3]
(b) Name the search algorithm used. [1]

Model answer.

(a) (1) `mid + 1` (1); (2) `mid - 1` (1); (3) `-1` (1)
(b) Binary search (1)

Don’t lose marks

- Forgetting to say that binary search needs the data to be sorted.
- Saying binary search 'splits the list in half' without saying it compares the middle item and discards the half that cannot contain the target.
- Keeping the middle item in the half you search next: it has already been checked.
- Stopping a linear search too early, or listing items after the target has been found.
- Choosing a different middle item from the rule given in the question.

More tips

Memory tricks

- Binary search needs a sorted list, just like looking up a word in a dictionary: open it in the middle, then choose a half.
- Maximum binary search comparisons: keep halving the number of items until you reach 1, counting the halvings, then add 1. For example 1000, 500, 250, 125, 62, 31, 15, 7, 3, 1 is 9 halvings, so at most 10 comparisons.
- Powers of 2 check: $2^{10} = 1024$, so up to 1023 items need at most 10 comparisons.

Exam technique

- Write the position numbers above the list before you start a binary search.
- Quote actual values in 'describe' answers: 'compare 30 with 16' earns more than 'compare with the middle'.
- If a question asks for an advantage of linear search, 'the data does not need to be sorted' is the safest answer.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3 — **Describe how a linear search works**
Check each item in turn from the start until the item is found or the end is reached.

Grade 4 — **State the condition needed for binary search**
The data must already be sorted, in ascending or alphabetical order.

Grade 5 — **Carry out a binary search on a list**
Find the middle item, compare, discard half the list, and repeat until found or empty.

Grade 6 — **Compare linear and binary search**
Binary search is much quicker on large sorted lists; linear search works on unsorted lists.

Grade 7 — **Complete or identify search algorithm code**
Recognise each search from its code and fill in missing lines, such as first = mid + 1.

Grade 8 — **Work out the maximum number of comparisons**
Binary search halves the list each time, so 1000 items need at most 10 comparisons.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.