Bubble sort

Algorithms and programming (Paper 2) · revision notes

A bubble sort goes through a list comparing each pair of neighbouring items and swapping them if they are in the wrong order, repeating passes until no swaps are made. You need to apply it by hand, describe it, recognise its code and compare it with the other sorts.

Key facts

- Compare adjacent pairs; swap if they are in the wrong order
- After pass 1 (ascending), the largest item is at the end
- Swap: temp = a, then a = b, then b = temp
- Con: slow and inefficient for large lists
- One pass = one run through the list, comparing every adjacent pair
- Stop when a pass makes no swaps
- Pro: simple; in place (little memory); quick to confirm a list is sorted

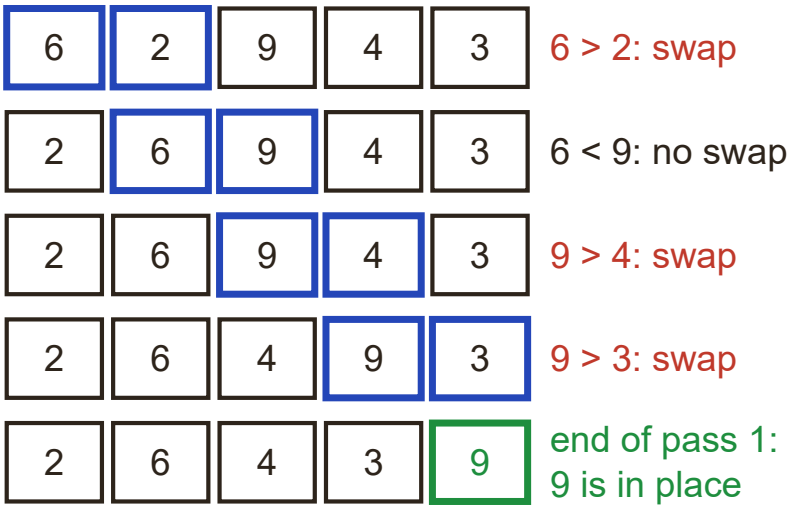
Notes

How a bubble sort works (ascending order)

- Compare the first two items. If they are in the wrong order, swap them.
- Move along one place and compare the next pair (items 2 and 3), and so on to the end of the list. This is one **pass**.
- After the first pass, the largest item has 'bubbled' to the end, into its final position.
- Repeat passes until a pass makes **no swaps**. The list is then sorted.
- Each pass can stop one item earlier than the last, because the end of the list is already sorted.

Worked example: 6, 2, 9, 4, 3

- Pass 1: swap 6 and 2; 6 and 9 stay; swap 9 and 4; swap 9 and 3. Result: 2, 6, 4, 3, 9.



Pass 1: each adjacent pair is compared, and swapped if it is in the wrong order

- Pass 2: 2 and 6 stay; swap 6 and 4; swap 6 and 3; 6 and 9 stay. Result: 2, 4, 3, 6, 9.
- Pass 3: swap 4 and 3. Result: 2, 3, 4, 6, 9.
- Pass 4: no swaps, so the sort stops.

The code

- A **temporary variable** is needed to swap two items, otherwise one value is overwritten and lost. A flag records whether any swap happened in the pass.

```
swapped = true
while swapped == true
   swapped = false
   for i = 0 to items.length - 2
      if items[i] > items[i + 1] then
         temp = items[i]
         items[i] = items[i + 1]
         items[i + 1] = temp
         swapped = true
      endif
   next i
endwhile
```

Advantages and disadvantages

- Advantages: simple to understand and code; sorts the list in place, so it needs very little extra memory; on a list that is already sorted it finishes after one pass.
- Disadvantage: very slow on large lists, because it makes many comparisons and swaps.

How to answer each type of question

Show the list after each pass

2 to 4 marks **Grade 5**

1. Work left to right, one adjacent pair at a time.
2. Write the list out again after every swap if it helps.
3. Only write the list as it stands at the end of each pass as your answer.
4. Count the final pass that makes no swaps if the question asks how many passes.

Example. A bubble sort is used to put 8, 3, 7, 1, 5 into ascending order.
(a) Show the list after the first pass and after the second pass. [2]
(b) State how many passes are made in total, including the final pass with no swaps. [1]

Model answer.

(a) After pass 1: 3, 7, 1, 5, 8 (1). After pass 2: 3, 1, 5, 7, 8 (1).
(b) 4 (1): pass 3 gives 1, 3, 5, 7, 8 and pass 4 makes no swaps.

Describe how a bubble sort works

3 to 4 marks **Grade 5**

1. Compare adjacent items (the first two to start with).
2. Swap them if they are in the wrong order.
3. Move along the list to the end: one pass.
4. Repeat passes until there are no swaps.

Example. Describe how a bubble sort puts a list of numbers into ascending order. [4]

Model answer. Compare the first two (adjacent) items (1). If the first is larger than the second, swap them (1). Move along one place and repeat for each pair to the end of the list, which completes a pass (1). Repeat the passes until a pass is made with no swaps (1).

Complete the swap in bubble sort code

2 to 3 marks | Grade 7

1. Store one value in a temporary variable first.
2. Copy the second value into the first position.
3. Copy the temporary value into the second position.
4. If asked why, say the first value would otherwise be overwritten and lost.

Example. Complete the code that swaps `items[i]` and `items[i + 1]`, and explain why a temporary variable is needed.

```
temp = ..........
items[i] = ..........
items[i + 1] = ..........
```

[3]

Model answer.

`temp = items[i]`, `items[i] = items[i + 1]`, `items[i + 1] = temp` (1 for the first two lines, 1 for the last line).

Without temp, the value in items[i] would be overwritten and lost before it could be copied (1).

Compare bubble sort with another sort

2 marks | Grade 8

1. Give an advantage of bubble sort: simple, or little extra memory.
2. Give a disadvantage: slow for large lists.

Example. Give **one** advantage and **one** disadvantage of a bubble sort compared with a merge sort. [2]

Model answer.

Advantage: it sorts in place, so it uses less memory / it is simpler to code (1).

Disadvantage: it is much slower (less efficient) for large lists (1).

Don’t lose marks

- Comparing items that are not next to each other.
- Stopping after one pass because the list 'looks' sorted: the sort only stops after a pass with no swaps.
- Writing the list after every single swap when the question asks for the list after each pass.
- Swapping two items without a temporary variable, so one value is lost.
- Mixing up bubble sort and insertion sort descriptions.

More tips

Memory tricks

- Picture bubbles rising: after each pass, the biggest unsorted item has risen to the top (the end of the list).
- Quick check after pass k: the last k items should be the k largest, in order.
- A list of n items needs at most n − 1 passes that make swaps, plus one final pass with no swaps.

Exam technique

- Show your working lightly beside the answer box; write each pass result clearly on its own line.
- When describing the sort, use the words 'adjacent', 'swap', 'pass' and 'no swaps'.
- If the question says descending order, swap when the first item is smaller.

What each grade needs

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

Grade 3 — **State what a bubble sort does**
It sorts a list by comparing adjacent items and swapping them if they are out of order.

Grade 4 — **Show the list after one pass**
Compare each adjacent pair from left to right, swapping where needed.

Grade 5 — **Describe the steps of a bubble sort**
Compare pairs, swap, finish the pass, then repeat passes until a pass makes no swaps.

Grade 6 — **Explain how the sort knows it has finished**
A pass with no swaps means every pair is in order, so the list is sorted.

Grade 7 — **Complete bubble sort code, including the swap**
Use a temporary variable to swap two items, and a flag to record whether a swap happened.

Grade 8 — **Evaluate bubble sort against other sorts**
Simple and memory-efficient, but slow for large lists compared with merge sort.

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.