CS4311
Design and Analysis of Algorithms

Tutorial: Assignment 1

Speaker: 劉富翊 (FOGA)

Outline

- What's a good answer
  - A sample question and solution
- Hint for the assignment 1
  - Due date: 3/12

Sample Question

Question:

Given a sequence of numbers, design an algorithm to find the maximum number. Show your algorithm is correct and analyze the time complexity.

How to write the answer?

Bad Answer

- Algorithm:

```
00  public class FindMax {
01      public static void main(String[] args) {
02          int data[] = new int[] {17, 26, 14, 19, 10, 23, 12, 5};
03          int count = 0;
04          int max = Integer.MIN_VALUE;
05          for(int i=0; i<data.length; i++) {
06              if(data[i] > max)
07                  max = data[i];
08              count++;
09          }
10          System.out.println("Max Number: " + max);
11          System.out.println("Count: " + count);
12      }
13  }
```

Bad Answer

```
C:\>java FindMax
Max Number: 26
Count: 8
C:\>
```

- Correctness:

  26 is actually the maximum of the input data, so the algorithm is correct.
- Time complexity:

  Since count = 8 is equal to the number of input data, therefore, the running time is O(n).

Good Answer

- Algorithm:

Given a sequence of number,

Step 1: Pick up the first number and suppose it is maximum.

Step 2: Pick up a number from the remaining numbers.

Step 3: Compare the picked number and current maximum.

Step 4: Throw out the smaller and suppose the larger one as the new maximum.

Step 5: Repeat 2~4 until there's no number remaining.

Step 6: Output the current maximum.

Good Answer

- Correctness:

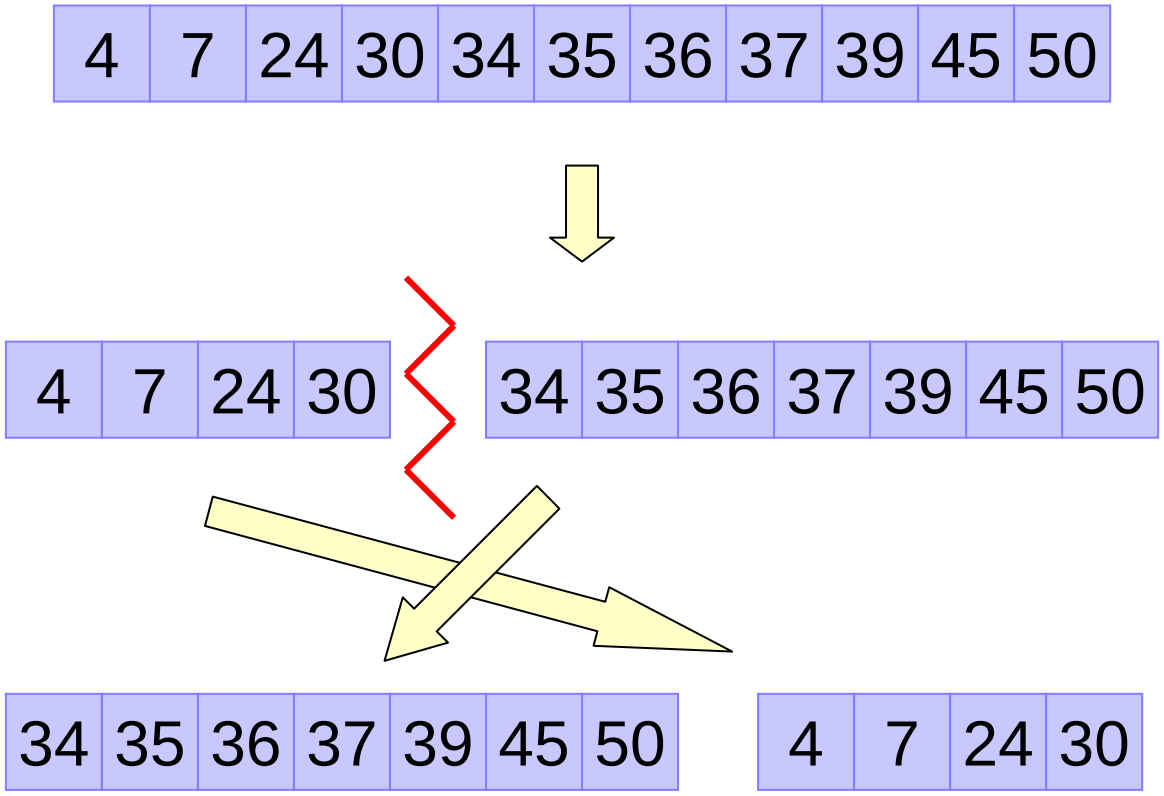
  Suppose our algorithm doesn't output the maximum number m, then m must be thrown out. It means m is smaller than some number. Since some number is greater than m, m is not maximum. We got a contradiction, therefore, our algorithm will output the maximum number.

Good Answer

- Time complexity:

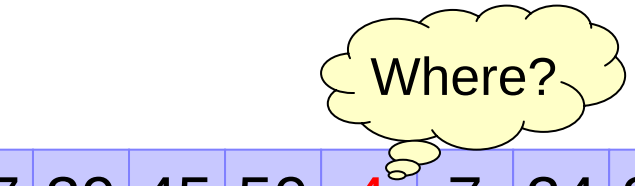
  The operations involved are picking or throwing out numbers. Since each number is only picked once and thrown out once, and there are totally n numbers, the running time is n * $O(1)$ = $O(n)$.

Question 1

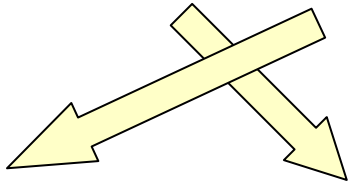

4 7 24 30 34 35 36 37 39 45 50
4 7 24 30
34 35 36 37 39 45 50
34 35 36 37 39 45 50
4 7 24 30

Question 1

34	35	36	37	39	45	50	4	7	24	30



34	35	36	37	39	45	50	4	7	24	30



4	7	24	30	34	35	36	37	39	45	50

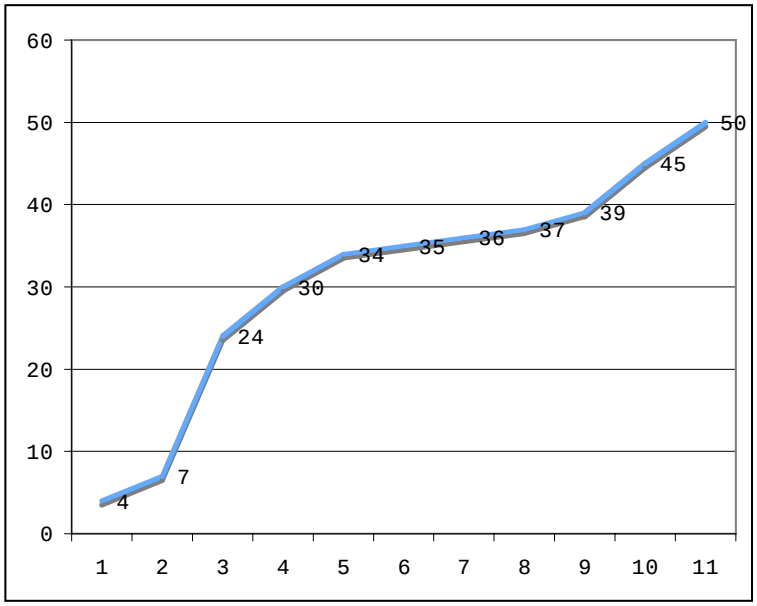
Question 1

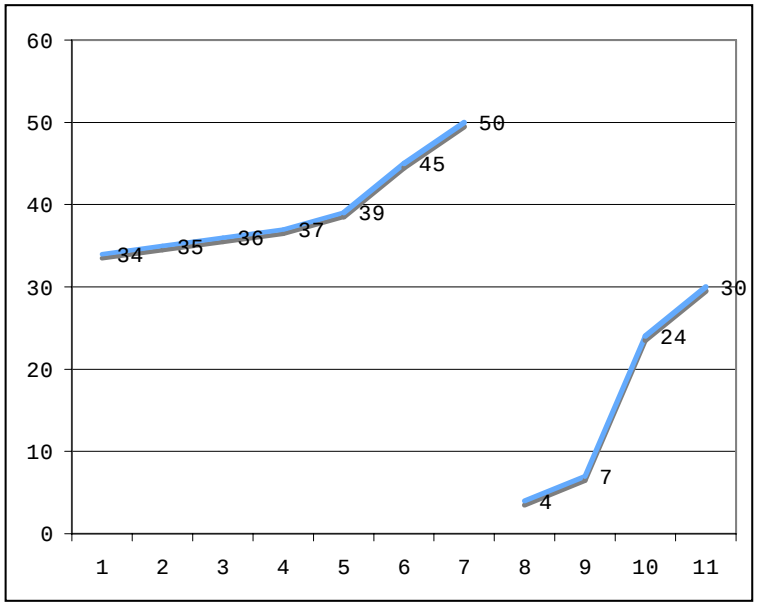
- Design an O(log n)-time algorithm to find the minimum item.

- Show that your algorithm is correct.

Question 1

- Hint:





Question 2

ComputeCount()

1. Input a positive integer *n*
2. Set *count = 0*
3. **for** *j = 1, 2, …, n*
4. **if** *j* is a factor of *n*
5. { Update *count* to become *count + 1* }
6. Output *count*

What's the running time?

$\Theta(n)$

Can it be faster?

Question 2

- Design a faster algorithm that can compute *count*.
  - $O(\sqrt{n})$, O(log n), O(1)…
  - Note: You can only use RAM operations.
  - E.g., you cannot assume 2^x or log x can be computed in O(1) time.
- Explain why your algorithm is correct.

Question 2

- Hint:
  What's the physical meaning of *count*?

Question 3

BubbleSort(*A*)

1. **for** Phase $k = 1, 2, \ldots, n$
2. **for** Position $j = 1, 2, \ldots, n - 1$
3. **if** $A[j] > A[j + 1]$
4. { Swap the entries $A[j]$ and $A[j + 1]$ }

Show the algorithm is correct.

Question 3

Inverted Pair:

Given $(A_0, A_1, A_2, \ldots, A_n)$

if $i < j$ and $A_i > A_j$,

then (A_i, A_j) is inverted.

Example:

Given *(2, 3, 6, 4, 0)*

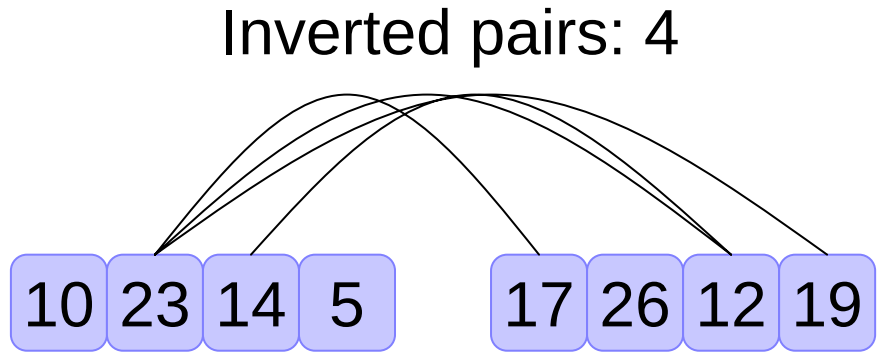
(2, 0), (3, 0), (6, 4), (6, 0), (4, 0) are inverted.

Question 3

- Show that the number of inverted pairs is exactly equal to the number of swaps when we perform BubbleSort.
- Design an algorithm that counts the number of inverted pairs in $O(n \log n)$ time.

Question 3

- Hint:
  (for the last part)



Total inverted pairs = 4 + 4 + 3 = 11

Question 4 (No marks)

Give asymptotic upper bound for each T(n)

a) $T(n) = 9T(n/2) + n^3$

b) $T(n) = 7T(n/2) + n^3$

c) $T(n) = T(\sqrt{n}) + \log n$

d) $T(n) = 0.5T(n/2) + n$

e) $T(n) = 3T(n/3) + n/3$

Good Luck

- Please try to write your answer in large font. Thanks a lot!