# **Answer of Tutorial for Section 1.3**

# Sorting algorithms

**Exercise 1**

1. Find the number of comparisons performed by selection sort algorithm to sort the array A = 5, 4, 1, 7, 5, 2, 3.
2. Sort the list A = {E, X, A, M, P, L, E} in alphabetical order by selection sort (refer to the slide page 6). How many swaps are there in sorting list A?

**Answer**

1. The number of comparisons made by selection sort algorithm to sort an array of size *n* is . As A has 7 elements, the number of comparisons will be 7×6/2 = 21.
2. 5 swaps are needed to sort A

E X **A** M P L E

A X **E** M P L E

A E X M P L **E**

A E E M P **L** X

A E E L P **M** X

A E E L M **P** X

A E E L M P X

**Exercise 2**

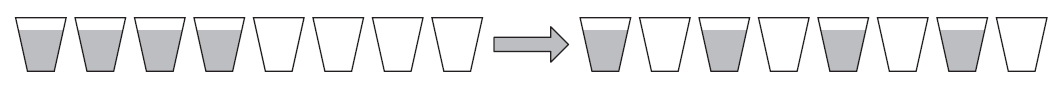
You have a row of 2n disks of two colors, *n* dark and *n* light. They alternate: dark, light, dark, light, and so on. You want to get all the dark disks to the right-hand end, and the light disks to the left-hand end. The only moves you are allowed to make are those that interchange the positions of two neighboring disks. Design an algorithm to solve this puzzle and determine the number of moves it takes.

**Answer**

1. Here is a simple and efficient (in fact, optimal) algorithm for this problem: Start with the leftmost light disk, swap it with each of the dark disks to the left of it. Repeat the same thing with the next most left light disk until all the light disks are swapped to the left-hand end.
2. The problem can also be solved by mimicking the swaps made by bubble sort in sorting the array of 1’s and 0’s representing the dark and light disks, respectively.

**Exercise 3**

There are 2*n* glasses standing next to each other in a row, the first *n* of them filled with orange juice while the remaining *n* glasses are empty. Make the glasses alternate in a filled-empty-filled-empty pattern in the minimum number of glasses moves.



**Answer**

Assuming that the glasses are numbered left to right from 1 to 2n, pour soda from glass 2*i* into glass 2(*n*-*i*)+1 for each *i* between 1 and floor(*n*/2). Thus, floor(*n*/2) is the least number of moves needed to solve the problem.

**Exercise 4**

Trace the Counting Sort algorithm on array A = {7, 4, 2, 1, 1, 3}.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 4 | 2 | 1 | 1 | 3 |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 2 | 1 | 1 | 1 | 0 | 0 | 1 |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 2 | 3 | 4 | 5 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 3 |  |  |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 2 | 3 | 3 | 5 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 1 |  | 3 |  |  |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 1 | 3 | 3 | 5 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 |  | 3 |  |  |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 3 | 3 | 5 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 2 | 3 |  |  |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 2 | 3 | 5 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 2 | 3 | 4 |  |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 2 | 3 | 4 | 5 | 5 | 6 |
| B | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 2 | 3 | 4 | 7 |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 2 | 3 | 4 | 5 | 5 | 5 |

**Exercise 5**

Design a linear algorithm to check if an array A[0..n-1] is sorted in increasing order. Your algorithm should return true if A is sorted and false otherwise.

**Answer**

**Algorithm** isSorted(A[0..n-1])

**Input**: An array A

**Output**: True if A is sorted, false otherwise.

**for** i 🡨 0 to n-2 **do**

**if** A[i] > A[i+1]

**return** false

**return** true