Exercise Set Part 3
Algorithms

Hans Georg Schaathun

10th August 2015

1 Thursday 18 September

Exercise 1.1 (Rosen p. 204, ex. 2) *Recall the characteristics Input, Output, Definiteness, Correctness, Finiteness, Effectiveness, and Generality, as defined in the video, or in Rosen's book p. 195.*

Consider the following algorithms, and determine for each one, which characteristics they posess and which they lack.

a)

```
1    procedure double(n : positive integer)
2    while n > 0
3      n := 2n
```

b)

```
1    procedure divide(n : positive integer)
2    while n > 0
3      m := 1/n
4      n := n − 1
```

c)

```
1    procedure sum(n : positive integer)
2    sum := 0
3    while i < 10
4      sum := sum + i
```

d)

```
1    procedure choose(a,b : positive integer)
2    x := either a or b
```

Exercise 1.2 *How many steps are required to move a Tower of Hanoi of ...*

1. *1 disk?*
2. *2 disks?*
3. *3 disks?*
4. *5 disks?*
5. *n disks? (this is difficult; we return to it later)*

Exercise 1.3 (Rosen p. 204, ex. 3) *Devise an algorithm which finds the sum of all the integers in a list.*

Exercise 1.4 (Video Solution «Exercise Example») *Devise an algorithm which takes a sorted array as input, and outputs an array of all repeated elements in the input.*

Exercise 1.5 *Devise an algorithm which finds the common elements in two sorted arrays. The output should be an array.*

2 Friday 19 September

Exercise 2.1 *Consider the following linear search algorithm:*

```
1       procedure find(k, A_1, A_2, ..., A_n)
2       for i := 1, 2, ..., n
3           if k = A_i, return i
```

It finds the index of an element k in an array $A_1, A_2, \ldots, A_n$. Rewrite the algorithm using recursion instead of a loop.

The trick to solve this problem recursively follows many known problems, such as *how to eat an elephant?* First you take one bite, then you eat the rest of the elephant. Or you could think of searching for your keys. First you check one pocket, then (if you don't find them) you check the rest of your pockets.

Specifically, searching for an element in a list, first you check the first element A_1 and see if it matches. If it does, you are done. Otherwise, you check the rest of the elements $(A_2, A_3, \ldots, A_n)$.

Solution:

```
1       function find(k, A_1, A_2, ..., A_n)
2       if n = 0, error
3       if A_1 = k, return 1
4       return 1 + find(k, A_2, A_3, ..., A_n)
```

Exercise 2.2 *Consider the following list of numbers:* $[6, 2, 5, 4, 7, 1]$*, and recall the algorithms for selection sort and insertion sort in the lectures.*

1. *Demonstrated how you sort the numbers step by step using selection sort.*

2. *Do the same for insertion sort.*

Discuss: Is the algorithm recursive as you perform it?

Exercise 2.3 *Consider the selection sort algorithm as described in the videos:*

```
1        For i = 1, 2, ..., n − 1 ,
2            for j = j + 1, j + 2, ..., n ,
3                if A_i > A_j , then swap A_i with A_j
```

Rewrite the selection sort algorithm in recursive form.

Hint: Consider the array at the start of iteration i In the outer loop. Can you identify a subarray which has to be sorted?

Exercise 2.4 *Give recurrence equations to give the number of comparisons required to sort an n-element array using ...*

1. *insertion sort.*

2. *merge sort.*

Exercise 2.5 *Tabulate the following recurrence for $n = 0, \ldots, 7$:*

$$T(n) = 0.5T(n-1) + 2, \tag{1}$$
$$T(0) = 0 \tag{2}$$

Can you spot a pattern? Try to guess a closed form expression.

Exercise 2.6 *Consider a sorted array A as input. Devise an algorithm which finds a given element k in A.*

1. *Give an iterative formulation of your algorithm,*

2. *Give a recursive formulation of your algorithm,*

3. *How many comparisons does your algorithm require to find k?*

3 Tuesday 23 September

Exercise 3.1 *Describe the main steps of a proof by induction.*

Exercise 3.2 *Consider the following recurrence.*

$$T(n) = 0.5T(n-1) + 2, \tag{3}$$
$$T(0) = 0 \tag{4}$$

Prove that $T(n) = 4 - 2^{2-n}$

Exercise 3.3 (Video 3-4-1) *Consider a ladder. If you stand on the ground, it is possible to step onto the first rung. If you stand at the nth rung, it is possible to climb onto the $(n+1)$st rung.*

Prove that you can climb to the top of the ladder, starting on the ground, using a proof by contradiction.

Exercise 3.4 *Prove that the Tower of Hanoi algorithm is correct.*

Exercise 3.5 *Prove that the output array of insertion sort (as given in previous videos) is sorted in increasing order.*

4 Thursday 25 September

Exercise 4.1 (Video Solution «Recurrence») *Iterate the recurrence to solve the following equation*

$$T(n) = 1.5T(n-1) + 1 \quad T(0) = 0.$$

Use mathematical induction to prove that the solution is correct.

Recall the recurrence for the Tower of Hanoi:

$$M(n) = 2M(n-1) + 1 \quad M(1) = 1$$

Exercise 4.2 *Solve the following recurrences, and explain how they differ from the recurrence of the Tower of Hanoi (above):*

$$M(n) = 2M(n-1) + 1 \quad M(0) = 0, \tag{5}$$
$$M(n) = 3M(n-1) + 1 \quad M(1) = 1, \tag{6}$$
$$M(n) = M(n-1) + 2 \quad M(1) = 1, \tag{7}$$

Exercise 4.3 *Solve the following recurrences, and explain how they differ from the recurrence of the Tower of Hanoi (above):*

$$T(n) = 0.5T(n-1) + 2, \quad T(0) = 0 \tag{8}$$
$$T(n) = r \cdot T(n-1) + 1, \quad T(0) = 0 \tag{9}$$
$$T(n) = r \cdot T(n-1) + s^n, \quad T(0) = 0 \tag{10}$$

Exercise 4.4 *Prove the following formula for geometric series directly using recursion:*

$$\sum_{i=0}^{n-1} r^i = \frac{1-r^n}{1-r}$$

Exercise 4.5 *Consider the function $f(n) = n^2 + 2n + 14$. Show that $f(n) = O(n^2)$. You have to review the definition of Big-O and find suitable constants c and k.*

5 Friday 26 September

Exercise 5.1 *Prove that every natural number greater than 7 is the sum of a non-negative integer multiple of 3 and a non-negative integer multiple of 5.*

Exercise 5.2 *Prove that the Strong Principle of Mathematical induction is valid. (You can use contradiction, following the pattern used in the Video introducing induction.)*

Exercise 5.3 (Video on «Structural Induction») *A set S can be defined as follows.*

Either S is the empty set $\emptyset$, or S is the union $S = S' \cup \{x\}$ of a set S' and a singleton set containing some element x.

Use structural induction to show that the number of possible subsets of S is 2^n where $|S| = n$ is the number of elements of S.

Exercise 5.4 *Consider*

$$f(x) = \sum_{i=0}^{n} a_i x^i$$

Prove that $f(x) \in O(x^n)$.

In other words, find c and k so that $|f(x)| \leq c\,|g(x)|$ for $x \geq k$.

Solution: Take $k = 1$ so that $x^i > x^{i-1}$ for all $x \geq k$. Then it is easy to see that

$$f(x) = \sum_{i=0}^{n} a_i x^i \geq x^n \sum_{i=0}^{n} a_i,$$

and we can take

$$c = \sum_{i=0}^{n} a_i,$$

to prove that $f(x) \in O(x^n)$.

Exercise 5.5 *What is the Big-O bound on the number of comparisons required by*

1. *insertion sort*
2. *selection sort*

Solution: 1. (insertion sort) Recall the previous counting exercises and not that sorting n elements take up to $c(n) = n(n-1)/2$ iterations in the worst case. We write

$$c(n) = 0.5n^2 - 0.5n,$$

and observe that n^2 is the dominant term. Thus $c(n) \in O(n^2)$.

2. Selection sort does the same nested loop as insertion sort, and has the same complexity.