Tutorial: Algorithms and complexity

Aim

The aim of this tutorial is for students to be able to communicate simple algorithms using pseudocode and flowcharts, perform some basic analyses of complexity, and write some simple recursive functions.

Note: some of these questions might look familiar.

Questions

1. You have been given a file containing two columns of numbers, and you have been asked for the sum of each column. The two sums should be printed out. Use an appropriate loop and draw a flowchart. The columns have the same number of rows.

2. Use pseudocode to write an algorithm that finds the smallest number in a list of n numbers $x_1, x_2, x_3, \ldots, x_n$

   What simple change could be made to find the largest number in the list?

3. Given the following pseudocode, can you be certain that it will halt? Try evaluating it for different values of x. What do you notice?

```
start
    read x
    do while (x does not equal 1)
        if (x is even)
            x = x/2
        else
            x = 3x + 1
        end if
    end do
end
```

4. Determine the likely big-O value for the following pseudocode snippets:

   a)

```
for i = 1 to m by 1
    sum = sum + i
end for
```

   b)

```
for i = 1 to m by 1
    for j = i to m by 1
        sum = sum + i*j
    end for
end for
```

c)

```
for i = 1 to m by 1
    for j = 1 to m by 1
        for k = 1 to m by 1
            if grid[i,j,k] > 0:
                sum = sum + 1
            end if
        end for
    end for
end for
```

d)

```
for i = 1 to m by 1
    sum = sum + i
end for

for j = 1 to m by 1
    product = product * j
end for
```

Consider the dominant part/primary parameter of the algorithm.

5. Determine the steps involved in question 1, and provide an indication of the complexity. Think about how long the code will take with files of 10, 20, and 100 rows; you might like to plot it.
6. Thinking about binary search, give an example of the worst case input/search value and the best case input/search value. How could you characterise these?
7. You have five different algorithms for finding an item in a list: they have complexities of $O(n)$, $O(n^2)$, $O(n^3)$, $O(\log n)$ and $O(2^n)$. Given the running time estimate for the first algorithm for a list of certain length, what is the estimated worst case runtime for the other algorithms for the same list? Round your answers to minutes/hours/days/weeks/months/years as appropriate.

a) 10 items, $O(n)$ = 10^{-5}s

b) 100 items, $O(n)$ = 10^{-4}s
c) 1000 items, $O(n)$ = 0.001s

8. Using pseudocode, write a recursive version for:

a) $f(n) = 2n$ (think about how multiplication is defined)
b) summing even numbers to n
c) binomial coefficients, e.g., $C(n, r)$

Extension questions

9. Write an iterative and a recursive version to calculate a particular Fibonacci number. Recall the Fibonacci sequence is 1, 1, 2, 3, 5, 8, 13 ... where each number is the sum of the preceding two numbers. If you have the inclination, code both up in a language of your choice and time them. Which performs better? Can you explain why? How could you improve your recursive version?

10. If you would like some induction questions, then take a look at DMOI3, section 2.5 (http://discrete.openmathbooks.org/dmoi3/sec_seq-induction.html): question 2 is suitable, as is question 10 but does not have an answer, and question 6 might provide a challenge (look at Example 2.5.3 first).