Weekplan: Warmup

Philip Bille Inge Li Gørtz

Reading

Survival Guide and Programming Prerequisites. Do the test in the Programming Prerequisites.

Exercises

1 Loops This exercise aims to check if you understand some basic programming concepts. Do *not* implement the functions. What do the functions `loop1`, `loop2`, `loop3` and `loop4` in Figure 1 return when

1.1 `n` $= 4$?

1.2 `n` $= 10$?

1.3 `n` $= 1000$?

1.4 as a function of `n`?

2 Recursion and Iteration A function is *recursive* if it calls itself. For instance the function `f(A,n)` in Figure **??**/ 1 is recursive. Solve the following exercises.

2.1 What does `f(A,n)` compute if `A` is an array of integers of length `n`? You should *not* implement it.

2.2 Rewrite `f(A,n)` to be *iterative*, ie. make a function that computes the same as `f(A,n)` but without calling itself.

3 [*w*†] **Standard Input and Output, and Redirection** In our practical implementation exercises, we will use the standard input and output for input and output, i.e., using `input()` and `print()`, respectively. We will use redirection for large inputs and outputs. Make sure you understand how this works by solving these exercises.

3.1 Implement a program `sum.py` that reads (from the standard input) two integers on a single line with a space between. Write the sum to the standard output.

3.2 Make a new text file `input.txt` containing two integers on a single line with a space between them. Redirect the file's contents to the standard input for your program, i.e., execute `python sum.py < input.txt` in your terminal. Do not modify `sum.py`.

3.3 Now redirect the standard output to a file, i.e., execute `python sum.py > output.txt`. Do not modify `sum.py`. Check that the newly created file `output.txt` contains the correct answer. Finally, also try the combination, i.e., execute `python sum.py < input.txt > output.txt`.

4 Linearthritis You have recently hired 128 programmers for your new high-tech startup company. Unfortunately, one suffers from the feared Linearthritis disease that makes everybody near the person write slow programs. To identify the diseased programmer, you have rented a special room that you can use to determine if the diseased programmer is within a group of your programmers. It is extremely expensive to rent this room, and the process needed to test a group is complicated (long and exhausting programming tests are necessary). Therefore, you would like to minimize the number of times you have to use the room to find the diseased programmer. Solve the following exercises.

4.1 Show you can find the diseased programmer using at most 7 tests.

4.2 How many tests do you need if you have n programmers instead of 128?

4.3 [$*$] Assume that you rent $k > 1$ rooms you can use to test k groups of programmers simultaneously. How many *rounds* of tests are enough to identify the diseased programmer? In each round, you can test k groups in parallel.

5 Zombie Duels You have an army of n brainless zombies. You want to find the strongest and the weakest zombie in the army. By pairing up two zombies in a cage with a big chunk of brain matter, you can quickly determine which of the two are the strongest. Unfortunately, zombies wear out in this process, so you want to minimize the number of duels needed. Solve the following exercises.

5.1 Explain how to find the strongest zombie using at most $n-1$ duels.

5.2 [$*$] Explain how to find the strongest and the weakest zombie using at most $3n/2$ duels.

5.3 [$**$] Explain how to find the strongest and second strongest zombie using at most $n + \log_2 n$ duels.

6 [$**$] **Ants on a Stick** Suppose that you have 100 ants on a stick of length 100 cm. At the start, each ant is placed at some position on the stick, pointing either toward the left or right end of the stick. Then, all ants begin to move simultaneously. The ants all move at a speed of 1 cm per second. If an ant bumps into another ant, they both immediately reverse directions and continue at the same speed, and if an ant reaches the end of the stick, it falls off the stick. What is the maximum duration of time before all ants have fallen off the stick over all possible initial placements of the ants?

```
def loop1(n):
   x = 0
   for i in range(n):
       for j in range(n):
           x += 1
   return x

def loop2(n):
   x = 0
   for i in range(n):
      x += 1
   for j in range(n):
      x += 1
   return x

def loop3(n):
   x = 0
   for i in range(n):
      if (i == n-1):
         for j in range(n):
            x += 1
   return x

def loop4(n):
   x = 0
   for i in range(n):
      for j in range(i,n):
         x += 1
   return x

def f(A, n):
   if (n == 0):
      return 0
   else:
      return f(A, n - 1) + A[n-1]
```

Figure 1: Loops and recursion in Python.