Practical #2: Dynamic Programming, Recursivity and Intro to Models

1. Dynamic Programming, Recursivity: Rabbits and Recurrence Relations

In this exercise, we will consider a mathematical exercise regarding the reproduction of a population of rabbits, which considers the following simplifying assumptions about the population dynamics:

1. The population begins in the first month, with a pair of newborn babies.
2. Rabbits reach reproductive age after one month.
3. In any given month, every rabbit of reproductive age mates with another rabbit of reproductive age.
4. Exactly one month after two rabbits mate, they produce one male and one female rabbit.
5. Rabbits never die or stop reproducing.

Let's see how many rabbits there are after one year. We can see that in the second month, the first pair of rabbits reach reproductive age and mate. In the third month, another pair of rabbits is born, and we have two rabbit pairs; our first pair of rabbits mates again. In the fourth month, another pair of rabbits is born to the original pair, while the second pair reach maturity and mate (with three total pairs). One can check that after a year, the rabbit population boasts 144 pairs, *i.e.* 288 rabbits.

Although the rabbits' immortality assumption may seem a bit farfetched, the model above is not unrealistic for reproduction in a predator-free environment. Case-in-point: European rabbits were introduced to Australia in the mid 19th Century, a place with no real indigenous predators for them. Within 50 years, the rabbits had already eradicated many plant species across the continent, leading to irreversible changes in the Australian ecosystem and turning much of its grasslands into eroded, practically uninhabitable parts of the modern Outback.

The rabbit reproduction model can be described via *a recurrence relation* of the *sequence F,* whose n-th element F_n represents the number of rabbit pairs alive after the n-th month.

A *sequence* is an ordered collection of objects (usually *numbers*), in which repetitions are allowed. We use the notation a_n to represent the n-th element of a sequence denoted by *a.*

A *recurrence relation* is a way of defining the terms of a sequence with respect to the values of the previous terms in said sequence. When finding the n-th term of a sequence defined by a recurrence relation, we can simply use the recurrence relation to generate terms for progressively larger values of *n.* This problem introduces us to the computational technique of dynamic programming, which successively builds up solutions by using the answers to smaller cases.

Example: The (inifinite) sequence $a = (0,2,4,6,8,\ldots.)$ of even numbers has $a_1=0$ as the initial element, and is defined by the recurrence relation $a_{n+1} = a_n + 2$

Q1. *What are the initial elements and the recurrence relation of the Fibonacci sequence F_n, under the assumptions that:*

- *the rabbit population dynamic satisfies conditions 1-5 from above*
- *any given month contains the rabbits that were alive the previous month, plus any new offspring*
- *the number of offspring in any month is equal to the number of rabbits that were alive two months prior.*

Have you seen this recurrence relation before? In what context ?

Q2. *Using the recurrence relation above, write an **iterative** Python function that returns the total number of rabbit pairs that will be present after **n** months. Then, write a **recursive** function that does the same thing. Compare the execution time of your two functions for different values of the parameter **n,** ranging from small to big. What can you observe, and what do you think is the cause of this behavior?*

Q3. *Write a Python function that takes as input two positive integers, **n** and **k**, and returns the total number of rabbit pairs that will be present after **n** months, if we begin with one pair and in each generation, every pair of reproductive-age rabbits produces a litter of **k** rabbit pairs (instead of only 1 pair).*

Sample input:

5 3

Sample output:

19

Q4. *Can rabbits become extinct in this model? Motivate your answer, and suggest changes in the model assumptions/description such that rabbits can reach extinction. How would you implement such changes in the mathematical model?*

2. **Mortal Fibonacci Rabbits**

We now want to expand the rabbit population model from above to a more realistic hypothesis, in which rabbits are mortal. For that, we assume that the rabbits die out after a fixed number of months, given by a parameter **m.**

Q1. How many rabbits does the population contain after 6 months, if the rabbit lifespan is 3 months?

Q2. The same as Q2 for Exercice 2, but now for different values of both **n** and **m.**

Q3. "Rabbit extinction, Rabbit apocalypse": we will try to answer the same question as in the first exercise, *i.e.* can rabbits become extinct in this model? What modifications in the model could lead to their extinction and how would you implement these changes? How about the opposite situation: in the

current model, can rabbits reproduce so much that they take over the world? Put your answer to test by trying different values of the model parameters.

3. Tumour cell growth

Preamble: Consider the following growth equation ([Cross&Cotton, 1994]):

$$N_{t+1} = rN_t(1 - \frac{N_t}{K})$$

With K denoting the number of cancer cells per Petri dish, and r denoting the cells' growth rate.
We normalize this equation by taking K=1; this means that N_t is the fraction of the total population of cancer cells that can be sustained in the cell culture container.

Q1. Write a Python program that given an initial fraction value for N_t, a value of r, and a number of iterations n, plots the **history** of the cell population size during n iterations.

Q2. Run the above program for the following parameter values. Observe and comment the different results:

- **a)** N_0 = 0.05, r=1.5, n=50
- **b)** N_0 = 0.05, r=2.5, n=50
- **c)** N_0 = 0.05, r=3, n=50
- **d)** N_0 = 0.05, r=3.5, n=50
- **e)** N_0 = 0.05, r=3.9, n=50

Suppose now multi-clonality is included in the model, with the various cell clones having different initial populations, and assume a growth rate of 3.98. A major pathological interest is in the total size of the tumour, *i.e.* in the total number of cells. We will first consider a model with 5 clones, then a model with 200 clones. Implement these two cloned models, and for each of them plot the total population during 90 iterations. Observe and compare the results, comment and speculate.

Q3.[Bonus question] How do you think the parameters of the initial equation can be estimated?

4. **Dynamic Programming: Point Mutations**

a) Preamble: In the "Counting Point Mutations" exercice from Practical 1, we saw that the Hamming distance gives us a preliminary notion of the evolutionary distance between two DNA strings by counting the minimum number of single nucleotide substitutions that could have occurred on the evolutionary path between the two strands.

However, in practice, homologous strands of DNA or protein are rarely the same length because point mutations also include the insertion or deletion of a single nucleotide. Thus, we need to incorporate these insertions and deletions into the calculation of the minimum number of point mutations between two

strings. One of the simplest models charges a unit "cost" to any single-symbol insertion/deletion, then (in keeping with the parsimony principle), requests the minimum cost over all transformations of one genetic string into another by point substitutions, insertions, and deletions.

Problem: Given two strings **s** and **t**, of possibly different lengths, the edit distance $\mathbf{d_E(s,t)}$ is the minimum number of edit operations needed to transform **s** into **t**. An edit operation is defined as the substitution, insertion, or deletion of a single symbol. The latter two operations incorporate the case in which a contiguous interval is inserted into or deleted from a string; such an interval is called a *gap*. For the purposes of this problem, the insertion or deletion of a gap of length **k** still counts as **k** distinct edit operations.

Write a Python function that takes two DNA strings **s** and **t** of length at most 1000 base-pairs each as input, and returns their edit distance $\mathbf{d_E(s,t)}$.

Sample input:

PLEASANTLY

MEANLY

Sample output:

5