Tutorial 4: Induction, Recursion

Presented by Rybin Dmitry
dmitryrybin@link.cuhk.edu.cn

The Chinese University of Hong Kong, Shenzhen

September 28, 2022

Tough problem for bored students

Let a_n be the number of all different binary strings that can be obtained from arbitrary concatenation of n strings from the set $\{0, 1, 01\}$. Find a_n. For example,

$$a_1 = 3$$

$$0, 1, 01$$

$$a_2 = 9$$

$$00, 01, 10, 11, 001, 101, 011, 010, 0101$$

$$a_3 = 26$$

In fancy words, consider the free monoid $M\langle 0, 1\rangle$ on letters 0, 1. Let $S = \{0, 1, 01\} \subset M\langle 0, 1\rangle$. compute $|S^n|$.

Recap: Induction

Recall that inductive proofs always consist of two steps:

- Proving base case
- Proving step

During the proof of step we can assume that we already know the truth of all previously obtained statements.

Exercise 1

Prove that each natural number $n > 19$ can be written as a sum of 11's and 3's e.g.

$$21 = 3 + 3 + 3 + 3 + 3 + 3 + 3$$

$$22 = 11 + 11$$

$$25 = 3 + 11 + 11$$

Solution to Exercise 1

Let us show by induction on k that numbers $20 + 3k, 21 + 3k, 22 + 3k$ can be written as a sum of 11's and 3's.

Base case: $k = 0$

$$20 = 3 + 3 + 3 + 11$$

$$21 = 3 + 3 + 3 + 3 + 3 + 3 + 3$$

$$22 = 11 + 11$$

Step: knowing representation as as sum for k, we get representation for $k + 1$ since

$$A + 3(k + 1) = (A + 3k) + 3.$$

Recap: Recursion and Recurrence

Algorithms and certain mathematical objects are often constructed step-by-step in some recursive maneer.

Sometimes sequence of objects $X_1, X_2, X_3, \ldots$ is generated by repeating the same operation $X_n = f(X_{n-1})$. Induction is a useful tool to prove properties of objects constructed in such fashion.

For example, how to construct all binary strings of length n? Use all binary strings of length $n-1$ and append 0 or 1 to the end. Hence the number x_n of binary strings of length n satisfies the recursion

$$x_n = 2x_{n-1}.$$

This is a very easy recursion and induction shows that $x_n = x_0 2^n = 2^n$.

Exercise 2

In how many ways can you write a number $n \in \mathbb{N}$ as a sum of positive natural numbers (different order of terms count as different ways) ?

For example, $3 = 3 = 2 + 1 = 1 + 2 = 1 + 1 + 1$.

Solution to Exercise 2

Let $F(n)$ denote the number of ways to write n as a sum of natural numbers. Let's compute first few values.

$$0 = 0,\ \ F(0) = 1,$$

$$1 = 1,\ \ F(1) = 1,$$

$$2 = 2 = 1 + 1,\ \ F(2) = 2,$$

$$3 = 3 = 2 + 1 = 1 + 2 = 1 + 1 + 1,\ \ F(3) = 4,$$

$$4 = 4 = 3 + 1 = 1 + 3 = 2 + 2 = 2 + 1 + 1 =$$

$$= 1 + 2 + 1 = 1 + 1 + 2 = 1 + 1 + 1 + 1,\ \ F(4) = 8.$$

Solution to Exercise 2

So our conjecture is $F(n) = 2^{n-1}$ for $n > 0$.
Recursive definition of partitioning the number n into sum. Take any $0 < k \leq n$ and write $n = k + (...)$. We can put any partition of $n - k$ instead of $(...)$. Hence there is a recursion

$$F(n) = F(n-1) + F(n-2) + ... + F(0).$$

By induction from this recursion we get $F(n) = 2^{n-1}$ - done.
Here is an alternative solution without recursive construction. This approach is called “balls and bars”. Consider the following string consisting of n 1's and $n-1$ 0's.

$$1|1|1|1|...|1.$$

Any subset of $n-1$ bars can be removed, giving a partition of n. E.g.

$$1\ 1\ 1|1\ 1 \leftrightarrow 3+2$$

Hence there are 2^{n-1} partitions.

Merge Sort

We are given an array $[a_1, ..., a_n]$ and we have to sort this array. We will do this by *Merge Sort*. It runs as follows:

1. if $n = 1$, then array is sorted.
2. Otherwise divide an array of length n into 2 sub-arrays (of sizes roughly $n/2$), sort them recursively by Merge Sort, and perform *merge* of two sorted arrays (the last step is known to take time $\approx cn$)

Let $T(n)$ denote the running time of the algorithm. Then by design we have

$$T(n) = 2T(n/2) + cn.$$

First term stands for recursive call of the procedure, second term stands for *merge*.

Exercise 3

Prove that $T(n) \leq An\log(n)$ for some A.

Solution to Exercise 3

Let's make a reasonable assumption that $T(n) \leq T(n+1)$. Let's prove by induction that $T(2^k) \leq c2^k \cdot k$.
Base: $T(1) = 0$, $T(2) = 2c \leq c \cdot 2 \cdot 1$.
Step:

$$T(2^{k+1}) = 2T(2^k) + c2^{k+1} = (ck + c)2^{k+1} \leq c(k+1)2^{k+1}.$$

Now, for arbitrary $n = 2^k + r$ we can derive the following bound:

$$T(n) = T(2^k + r) \leq T(2^{k+1}) \leq c \cdot 2^{k+1} \cdot (k+1) \leq 4cn \cdot \log(n).$$

Exercise 4

We have initial capital of \$3000 at year $t = 0$. We found 2 investment products. One changes its valuation each year by recursion

$$x_{t+1} = 2x_t - \$1000.$$

Another changes according to recursion

$$y_{t+1} = 1.5y_{t+1} + \$2000.$$

Which one should we choose if our planning horizon is 10 years?

Solution to Exercise 4

We should compare x_{10} and y_{10}, given that $x_0 = y_0 = \$3000$. By induction we can show that

$$x_n = 1000 \cdot (2 \cdot 2^n + 1),$$

$$y_n = 1000 \cdot (7 \cdot 1.5^n - 4),$$

hence

$$x_{10} = 1000 \cdot 2049 > 1000 \cdot (2.25^5 - 4) = y_{10}.$$

Extra question: what is the optimal strategy if each year you can use capital to form any non-negative combination of two products (no debt allowed)?

Thank You

Thank you for your attention!