Tutorial #6. Sequences and Induction

Sequence

We consider sequence as a function over a set if integers from **k** to **n** with a step 1. To express we can either use f(k), f(k+1), …, f(n) OR a_k, a_{k+1}, a_{k+2}, … a_n that is read as «A-sub-K». If domain of the function is finite, this is a **finite sequence**. If domain is infinite 1, 2, 3, …., n, … - **infinite sequence.** First element of the sequence is called **initial term**, last (if exists) – **final term**.

To express the sequence we can use either

- **Explicit (general) formula: a_k = f(k)**, or
- **Recurrent formula**: **a_{k+1} = f*(a_k)**

Converting one formula to another can be helpful to solve some problems:

1) You have bank account with 10% yearly income for $1M sum. When you will be able to buy a flat, if it costs now $2M, and inflation rate is 7%?

   a_0 = 1M
   a_{k+1} = a_k*1.1
   a_k = 1M*1.1^k
   b_n = 2M*1.07^n
   a_k >= b_k;
   1*1.1^k >= 2*1.07^k ;
   $(1.1/1.07)^k$ >= 2 ;
   k >= $\log_{1.028} 2$= $1/\log_2 1.028$
   k >= 25.067

Summation and product notation

There's a *sigma* and *pi* notations to represent a sum and product of a sequence. Upper and lower limits represent **initial term**, **final term**. You can do the following:

1) Move the limits: $\sum_{i=-1}^{80}(i+2)^2 = replace\ i\ with\ k - 2, move\ both\ limits\ "in\ opposite\ direcition" = \sum_{k=1}^{82}((k-2)+2)^2 = \sum_{k=1}^{82}(k)^2 = can\ return\ back\ to\ "i" = \sum_{i=1}^{82} i^2$
   a. Do the exercise with $\prod_{a=4}^{17}(a^2 - 6a + 9)$
2) Detach and attach final and initial terms.

$$\text{a.}\ \sum_{i=1}^{n+1}\frac{1}{i^2} = \sum_{i=1}^{n}\frac{1}{i^2} + \frac{1}{(n+1)^2} \qquad \text{b.}\ \sum_{k=0}^{n} 2^k + 2^{n+1} = \sum_{k=0}^{n+1} 2^k$$

3) Use associative and distributive rules (with the same limits!!!).

$$1.\ \sum_{k=m}^{n} a_k + \sum_{k=m}^{n} b_k = \sum_{k=m}^{n}(a_k + b_k)$$

$$2.\ c \cdot \sum_{k=m}^{n} a_k = \sum_{k=m}^{n} c \cdot a_k \qquad \text{generalized distributive law}$$

$$3.\ \left(\prod_{k=m}^{n} a_k\right) \cdot \left(\prod_{k=m}^{n} b_k\right) = \prod_{k=m}^{n}(a_k \cdot b_k).$$

There are some special sequences:

1) Arithmetic progression, it's sum is $\frac{n(a_1 + a_n)}{2}$
2) Geometric progression, it's sum is $\frac{a(1 - r^m)}{1 - r}$
   a. **Fork bomb** is an attack method, what each process creates **N** subprocesses. Assume, each process finish its task in **one second**, and occupies **64kB of memory**. How much time will it take for a fork bomb (N=2) to bring down a server with 128GB of RAM?

      Solution: this is a sum of geometric progression: 64K, 64K*2, 64K*2*2, …;
      R_n = 64K*2^n. Sum = 64K (1 - 2^x)/(1-2);
      64K (1 - 2^x)/(1-2) > 128GB = 128*2^{20}KB
      $2^x - 1 > 2^{21}$; $2^{x-21} > 1$; x-21 > 0; x > 21 – 21 sec.
3) Natural number sequence (also a kind of arithmetic). Product of this sequence is called **factorial**. It is expressed as

$$n! = \prod_{i=1}^{n} i\ ;\ 0! = 1$$

and can be treated as the number of all possible ways you pull all the balls from the bag blindly. This number is growing fast.

Do the task: calculate 33! **exactly in your program**.

Solution: none of fixed-size types can handle this value. You should use "long arithmetic" for this.

e.g. C#:

```
System.Numerics.BigInteger a = 1;
for (int i = 2; i <= 33; i++)
{
	a *= i;
	Console.WriteLine(a);
}
Console.ReadLine();
```

Proof by induction

Proof by induction is a proof of proposition *P(k)* for a whole sequence *k in* {*a, a+1, a+2, …*}, that is based on:

- well-known fact about sequence start (BASIC STEP, P(a) == true) and
- some information *I*, that makes P(k)∧I ⊢ P(k+1) (Induction step).

Induction step can be, e.g. recursive notation that implies conclusion. We already used induction to prove the complexity of the algorithms.

Coins case: If we remove 1 cent coin from monetary system, will it affect the market? Will we be able to construct any sum greater than 8 cents with 3 & 5-cent coins?

For all integers $n \geq 8$, $P(n)$ is true, where $P(n)$ is the sentence "n cents can be obtained using 3¢ and 5¢ coins."

Let the property $P(n)$ be the sentence

n¢ can be obtained using 3¢ and 5¢ coins. $\leftarrow P(n)$

Show that P(8) is true:

$P(8)$ is true because 8¢ can be obtained using one 3¢ coin and one 5¢ coin.

Show that for all integers $k \geq 8$, if P(k) is true then P(k+1) is also true:

[Suppose that P(k) is true for a particular but arbitrarily chosen integer $k \geq 8$. That is:] Suppose that k is any integer with $k \geq 8$ such that

k¢ can be obtained using 3¢ and 5¢ coins. $\leftarrow P(k)$ inductive hypothesis

[We must show that $P(k+1)$ is true. That is:] We must show that

$(k+1)$¢ can be obtained using 3¢ and 5¢ coins. $\leftarrow P(k+1)$

Case 1 (There is a 5¢ coin among those used to make up the k¢.): In this case replace the 5¢ coin by two 3¢ coins; the result will be $(k+1)$¢.

Case 2 (There is not a 5¢ coin among those used to make up the k¢.): In this case, because $k \geq 8$, at least three 3¢ coins must have been used. So remove three 3¢ coins and replace them by two 5¢ coins; the result will be $(k+1)$¢.

Thus in either case $(k+1)$¢ can be obtained using 3¢ and 5¢ coins *[as was to be shown].*

[Since we have proved the basis step and the inductive step, we conclude that the proposition is true.]

WB: Prove that binary tree has at most $NC_n = 2^n-1$ nodes, and $LC_n = 2^{n-1}$ nodes at the last level, where **n** is number of levels.

Solution: **base case**: 1-level tree has **one node** and one element at the last level. Let tree with **k** levels have a most 2^k-1 nodes, and 2^{k-1} nodes at the last level. Each node at last level can have at most 2 children, so there will be no more then $LC_{k+1} = 2^{k-1}*2 = 2^{(k+1)-1}$. Number of nodes in a tree then will be no more then $NC_{k+1} = NC_k + LC_{k+1} = 2^k-1 + 2^k = 2^{(k+1)}-1$, QED