Abracadabra

Students: Arion Dan Vasile 12th grade, Toader Rareș 11th grade
Jidavu Daria 10th grade, Harambaș Diana 10th grade
Claudiu Gabriel Pop 10th grade, Marcus Cotrau 10th grade
Laurențiu Grad 10th grade, Lăuran David 10th grade
Ștefan Boca 10th grade
Coordinator Teacher: Ariana Văcărețu
Researcher: Lorand Parajdi, “Babeș-Bolyai” University

Colegiul Național ”Emil Racoviță”
Cluj-Napoca, Romania,
November 2019

Contents

1 Introduction

Let a transformation be A becomes AB and B becomes A.
For example, if we start from A, we obtain AB, then ABA, ABAAB and so on.
What can we say about the element at the n^{th} stage?

2 Solution

2.1 Rewriting the problem

Let $(T_n)_{n\geq 1}$ be an array, $T_1 = A$ and $\Upsilon = \{strings|letters \in \{A, B\}\}$
Let f: $\Upsilon \to \Upsilon$ so that $f(x) = \begin{cases} A \to AB \\ B \to A \end{cases}$ f(T_{k-1})=T_k
Let $\oplus$ be a composition over Υ defined as concatenation of 2 strings.
We observe that $\oplus$ is not commutative, but is associative.
Because f is applied on unitary elements, if we have t_1 and t_2, $f(t_1 \oplus t_2) = f(t_1) \oplus f(t_2)$.
As we previously said, $\oplus$ is associative, and it is a law of composition over Υ.
Then f is an endomorphism over the semi-group $(\Upsilon, \oplus)$.

$T_1 = A$	$T_3 = ABA$	$T_5 = ABAABABA$
$T_2 = AB$	$T_4 = ABAAB$	$T_6 = ABAABABAABAAB$

2.2 Demonstration regarding the number of letters

Let $(a_n)_{n\geq 1}$ and $(b_n)_{n\geq 1}$ be two arrays, a_k and b_k defined by the number of A and respectively B letters in T_k.

a_1=1	a_2=1	a_3=2	a_4=3	a_5=5	a_6=8	a_7=13	a_8=21
b_1=0	b_2=1	b_3=1	b_4=2	b_5=3	b_6=5	b_7=8	b_8=13

We observe that $a_3 = a_2 + a_1$, $a_4 = a_3 + a_2$ and so on.
Because of the form of f, we have $a_k = a_{k-1} + b_{k-1}$ and $b_k = a_{k-1} \Longrightarrow$

$$\begin{cases} a_k = a_{k-1} + a_{k-2} \\ b_k = a_{k-1} \end{cases}$$

2.3 Demonstration regarding the formation of elements

We also observe that $T_3 = T_2 \oplus T_1$, $T_4 = T_3 \oplus T_2$ and so on.
We are going to demonstrate this by induction.
P_n : "$T_n = T_{n-1} \oplus T_{n-2}$", $\forall n \in N$, n ≥ 3
I)Exemplification:
P_3 : "$T_3 = T_2 \oplus T_1$"

$$\begin{cases} T_3 = ABA \\ T_2 \oplus T_1 = AB \oplus A = ABA \end{cases} \implies P_3 \text{ True}$$

II)Demonstration :
We suppose $P_k =$ "$T_k = T_{k-1} \oplus T_{k-2}$" True
Having in mind f and it's properties, we have:
$f(T_k) = f(T_{k-1} \oplus T_{k-2}) \iff f(T_k) = f(T_{k-1}) \oplus f(T_{k-2}) \iff$
$T_{k+1} = T_k \oplus T_{k-1} \iff P_{k+1}$ True
According to I) and II) $\implies P(n)$ is True

2.4 Finding a general form of the n^{th} element

Let g: $\Upsilon \rightarrow N_2$, g(x)=$\begin{cases} A\rightarrow 1 \\ B\rightarrow 0 \end{cases}$
Let $\oplus$ be a composition over N_2 defined as concatenation of 2 numbers.
Let $(N_{T_n})_{n\geq 1}$ be an array, $N_{T_k} = (g(T_k))_{(10)}$
$g^{-1}(N_{T_n(2)}) = T_n$

$N_{T_1} = 1$	$N_{T_3} = 5$	$N_{T_5} = 181$
$N_{T_2} = 2$	$N_{T_4} = 22$	$N_{T_6} = 5814$

Let nrDig(x) be the number of digits x has in base 2.
Because $T_n = T_{n-1} \oplus T_{n-2} \implies g(T_n) = g(T_{n-1}) \oplus g(T_{n-2})$
Switching from $\oplus$ to $+ \implies$
$g(T_n) = g(T_{n-1}) \oplus g(T_{n-2}) \iff g(T_n) = (g(T_{n-1}) \oplus \overline{0000..00}) + g(T_{n-2})$
Moving the last equation to base 10 $\implies$
$N_{T_n} = N_{T_{n-1}} * 2^{nrDig(T_{n-2})} + N_{T_{n-2}}$
nrDig(T_{n-2})=$a_{n-2} + b_{n-2}$=$a_{n-1} \implies N_{T_n} = N_{T_{n-1}} * 2^{a_{n-1}} + N_{T_{n-2}}$,
$\forall n \in N, n > 2$

$a_1 = 1, a_2 = 1, a_n = a_{n-1} + a_{n-2}, \forall n \in N, n > 2, N_{T_1} = 1, N_{T_2} = 2$
We are going to try explicitating N_{T_n}
Writing the characteristic equation we obtain $t^2 - 2^{a_{n-1}} - 1 = 0$
Because $\Delta = 2^{2a_{n-1}} + 4$
The possible solutions of the equation are
$t_{1,2} = \frac{2^{a_{n-1}} \pm \sqrt{2^{2a_{n-1}}+4}}{2} = 2^{a_{n-1}-1} \pm \sqrt{2^{2a_{n-1}-2}+1}$
Because neither solution is independent from n, then the recurrent series cannot be explicitated

3 Other Solution

3.1 Description of the algorithm

Because the other solution was inconcludent, we started an algorithm that can form the number

Let be the rule Let f_2: $g(\Upsilon) \to g(\Upsilon)$ so that $f_2(x) = \begin{cases} 1 \to 101 \\ 0 \to 10 \end{cases}$

Proceeding with the following rule, we observe that $f_2(x)$=$f \circ f$(x)
Knowing that f is applicable on singular elements and with this rule,
we can recursively generate the n^{th} stage

3.2 Code

```
#include <iostream>
#define msize 100000
using namespace std;
int v[5];
int v2[5];
unsigned long long fib1=1,fib2=2;
unsigned long long fibbo(int n){
    if(n==1) return 1;
    if(n==2) return 2;
    for(int i=1;i<=n-2;i++){
        unsigned long long a= fib1+fib2;
        fib1=fib2;
        fib2=a;
    }
    return fib2;
}
void afisare(int v[], int v2[], int k, int p)
{
 if(p==1)
 {
     if(k==0)
          for(int i=1;i<=v[0];i++)
              if(v[i]==1)cout<<"A";
                   else cout<<"B";
     else if(k==1)
         for(int i=1;i<=v2[0];i++)
              if(v2[i]==1)cout<<"A";
                   else cout<<"B";
 }
 else {
    if(k==0)
          for(int i=1;i<=v[0];i++)
```

```
                afisare(v,v2,v[i],p-1);
        else if(k==1)
            for(int i=1;i<=v2[0];i++)
                    afisare(v,v2,v2[i],p-1);
 }
}
int main()
{

    v2[0]=3,v2[1]=1,v2[2]=0,v2[3]=1,v[0]=2,v[1]=1,v[2]=0;
    int n;
cout<<"Enter the n-th element of the string you want to know:"<<endl;
    cin>>n;
  if(92>=n) cout<<"The number of letters is: "<<fibbo(n)<<endl;
  else
    cout<<"The number of letters is way greater than: "<<fibbo(92)<<endl;
        if(n==1)
            cout<<"A";
        else if(n==2)
            cout<<"AB";
        else if(n==3)
            cout<<"ABA";
        else
           afisare(v,v2,n%2,n/2);
    return 0;
}
```

4 Alternative code

4.1 Description of the algorithm

In order to reduce the time of execution, we adapted the last algorithm.
At this stage, we modified the code to accept any number of steps
In the previous section, we used 2 as a number of steps. Now, user can input a number from 2 to 25.

4.2 Code

```
#include <iostream>
#define msize 100000
using namespace std;

int w[30][125000];
unsigned long long fib1=1,fib2=2;
unsigned long long fibbo(int n){
    if(n==1) return 1;
    if(n==2) return 2;
    for(int i=1;i<=n-2;i++){
        unsigned long long a= fib1+fib2;
        fib1=fib2;
        fib2=a;
    }
    return fib2;
}
void afisare(int v[], int v2[], int k, int p)
{
  //  cout<<v[0]<<endl;
 if(p==1)
 {
     if(k==0)
         for(int i=1;i<=v[0];i++)
             if(v[i]==1)cout<<"A";
                 else cout<<"B";
     else if(k==1)
        for(int i=1;i<=v2[0];i++)
            if(v2[i]==1)cout<<"A";
                else cout<<"B";
 }
 else {
    if(k==0)
         for(int i=1;i<=v[0];i++)
              afisare(v,v2,v[i],p-1);
     else if(k==1)
        for(int i=1;i<=v2[0];i++)
                afisare(v,v2,v2[i],p-1);
 }
}
void formatare(){
    for(int i=3;i<=25;i++){
        w[i][0]=w[i-1][0]+w[i-2][0];
        for(int j=1;j<=w[i-1][0];j++)
                w[i][j]=w[i-1][j];
        for(int j=1;j<=w[i-2][0];j++)
            w[i][j+w[i-1][0]]=w[i-2][j];
    }
}
int main()
{

    w[1][0]=1,w[1][1]=1;
    w[2][0]=2,w[2][1]=1,w[2][2]=0;
```

```
     formatare();
     int n,k;
cout<<"Enter the n-th element of the string you want to know:"<<endl;
     cin>>n>>k;
      if(92>=n) cout<<"The number of letters is: "<<fibbo(n)<<endl;
   else
     cout<<"The number of letters is way greater than: "<<fibbo(92)<<endl;
          if(n<=k)
               for(int i=1;i<=w[n][0]; i++)
                    if(w[n][i]==1)cout<<'A';
                              else cout<<"B";
          else
               for(int i=1;i<=w[n%k][0]; i++)
                    afisare(w[k],w[k+1],w[n%k][i],n/k);
     return 0;
}
```

5 First Generalisation

5.1 Introduction

At this point of the problem we will not longer consider that we start from A

5.2 Demonstration regarding the number of letters

Using notations from the begging of the solve, we have that
Due to the form of f, the reccurence for a_n and b_n is
$$\begin{cases} a_k = a_{k-1} + a_{k-2} \\ b_k = a_{k-1} \end{cases} \quad \forall k > 2$$
But now, a_1 and b_1 are dependent on the form of T_1
Let $T_1 = \underbrace{AAA...A}_{c_1 \text{ times}} \underbrace{BBB...B}_{d_1 \text{ times}} .... \underbrace{AAA...A}_{c_x \text{ times}} \underbrace{BBB...B}_{d_x \text{ times}}$
Where $c_k, d_k \in N, \forall k \in \{1, 2, 3..., x\}$
So, here, $a_1 = c_1 + c_2 + ... + c_x$ and $b_1 = d_1 + d_2 + ... + d_x$

5.3 Demonstration regarding formation of the elements

Keeping in mind that $f(B) = A$, so $f^n(B) = f^{n-1}(A) \forall n \in N$
Because f is an endomorphism over $(\Upsilon, \oplus)$ we can descompose T_1 in letters and write the reccurent series for T_n
Let $\xi(n,p) = f^n(A) \oplus f^n(A) \oplus ... \oplus f^n(A)$, where there are p $f^n(A)$
So, we have $f^n(T_1) = \xi(n, c_1) \oplus \xi(n-1, d_1) \oplus ..... \oplus \xi(n, c_x) \oplus \xi(n-1, d_x)$

6 Second Generalisation

6.1 Introduction

At this point we are going to consider that f is more complex that before.
Let $\gamma(A,p) = AA....A$, where A appears p times
Let F: $\Upsilon \to \Upsilon$ so that $F(x) = \begin{cases} A \to \gamma(A,c_1) \oplus \gamma(B,d_1) \oplus ..... \oplus \gamma(A,c_x) \oplus \gamma(B,d_x) \\ B \to \gamma(A,h_1) \oplus \gamma(B,i_1) \oplus ..... \oplus \gamma(A,h_y) \oplus \gamma(B,i_y) \end{cases}$ F(T_{k-1})=T_k
Where x,y $\in$ N and $c_k, d_k \in N, \forall k \in \{1,2,...,x\}$ and $h_k, i_k \in N, \forall k \in \{1,2,...,y\}$

6.2 Demonstration regarding the number of letters

To caracterize the number of letters, $a_k = a_{k-1} * (c_1 + c_2 + ...c_x) + b_{k-1} * (h_1 + h_2 + ... + h_y)$
and $b_k = a_{k-1} * (d_1 + d_2 + ...d_x) + b_{k-1} * (i_1 + i_2 + ... + i_y)$
Splitting T_k into 2 strings and applying F over them, we observe that F is a morphism over $(\Upsilon, \oplus)$.

6.3 Demonstration regarding formation of the elements

Now we only need to find a point where $T_k = T_{o_1} \oplus T_{o_2} \oplus .... \oplus T_{o_j}$,
with j $\in N$ and $o_1, o_2....o_j \in \{1,2,3...k\}$ where k is minimum.
Let T_{k_A} be the k^{th} term of the array, starting from A
Let T_{k_B} be the k^{th} term of the array, starting from B
Thus, we have:
$T_{k_A} = \gamma(T_{k-1_A}, c_1) \oplus \gamma(T_{k-1_B}, d_1) \oplus ..... \oplus \gamma(T_{k-1_A}, c_x) \oplus \gamma(T_{k-1_B}, d_x)$
$T_{k_B} = \gamma(T_{k-1_A}, h_1) \oplus \gamma(T_{k-1_B}, i_1) \oplus ..... \oplus \gamma(T_{k-1_A}, h_y) \oplus \gamma(T_{k-1_B}, i_y)$
We are going to create a table, to show what happens with variants of F

6.4 Studying some particular cases

We'll not study the cases where at least three of $c_1, d_1, h_1, i_1, c_2, h_2$ are not null

$$1)F_1 = \begin{cases} A \to \gamma(A,c_1) \oplus \gamma(B,d_1) \oplus ..... \oplus \gamma(A,c_x) \oplus \gamma(B,d_x) \\ B \to A \end{cases}$$

Because $F_1(B) = A$ then $(F_1(B))^n = (F_1(A))^{n-1}$
We can rewrite F_1 as

$$F_1 = \begin{cases} A \to \gamma(F_1(B),c_1) \oplus \gamma(B,d_1) \oplus ..... \oplus \gamma(F_1(B),d_x) \oplus \gamma(B,d_x) \\ B \to A \end{cases}$$

Then $(F_1)^n(A) = \gamma((F_1)^n(B), c_1) \oplus \gamma((F_1)^{n-1}(B), d_1) \oplus ..... \oplus \gamma((F_1)^n(B), c_x) \oplus \gamma((F_1)^{n-1}(B), d_x)$
$\iff T_k = \gamma(T_{k-1}, c_1) \oplus \gamma(T_{k-2}, d_1) \oplus ..... \oplus \gamma(T_{k-1}, c_x) \oplus \gamma(T_{k-2}, d_x)$

7 Third Generalisation

7.1 Introduction

Let $\Gamma : \tau \to \tau$ so that $\Gamma(x) = \begin{cases} \mathrm{a}_1 \to a_{b_{(1,1)}} a_{b_{(1,2)}} ... a_{b_{(1,c_1)}} \\ \mathrm{a}_2 \to a_{b_{(2,1)}} a_{b_{(2,2)}} ... a_{b_{(2,c_2)}} \\ . \\ . \\ . \\ \mathrm{a}_v \to a_{b_{(v,1)}} a_{b_{(v,2)}} ... a_{b_{(v,c_v)}} \end{cases}$ $\Gamma(T_{k-1}) = T_k$

$\tau = \{strings\}$
$v \in N^*, v \geq 2$
$c_1, c_2 ... c_k \in N^*$
$\{b_{k,1}, b_{k,2} ... b_{k,c_k}\} \subseteq \{1, 2, 3..v\}, \forall k = \overline{1, \mathrm{v}}$

7.2 Demonstration regarding the number of letters

Let $(\mathrm{Dig}_{a_{k_n}})_{(n \geq 1)}$ be the array, where $\mathrm{Dig}_{a_{k_n}}$ number of a_k letter in T_n, $\forall$ n $\in N^*$ and $\forall k = \overline{1, \mathrm{v}}$
Let E_{a_k, a_p} the number of a_p letters after the transformation of a_k, $\forall k, p = \overline{1, \mathrm{v}}$
So E_{a_k, a_p}= number of times a_p appears in the array $b_{(a_k,1)}, b_{(a_k,2)}, ..., b_{(a_k,c_{a_k})}$
$Dig_{a_{k_n}} = \sum\limits_{i=1}^{v} E_{a_k,a_i} \mathrm{Dig}_{a_{i_{(n-1)}}}$
Let T_{p_u} be the p^{th} element of the array starting from $u \implies$
$T_{p_{a_u}} = T_{(p-1)_{a_{b_{(u,1)}}}} \oplus T_{(p-1)_{a_{b_{(u,2)}}}} \oplus .... \oplus T_{(p-1)_{a_{b_{(u,c_u)}}}}$