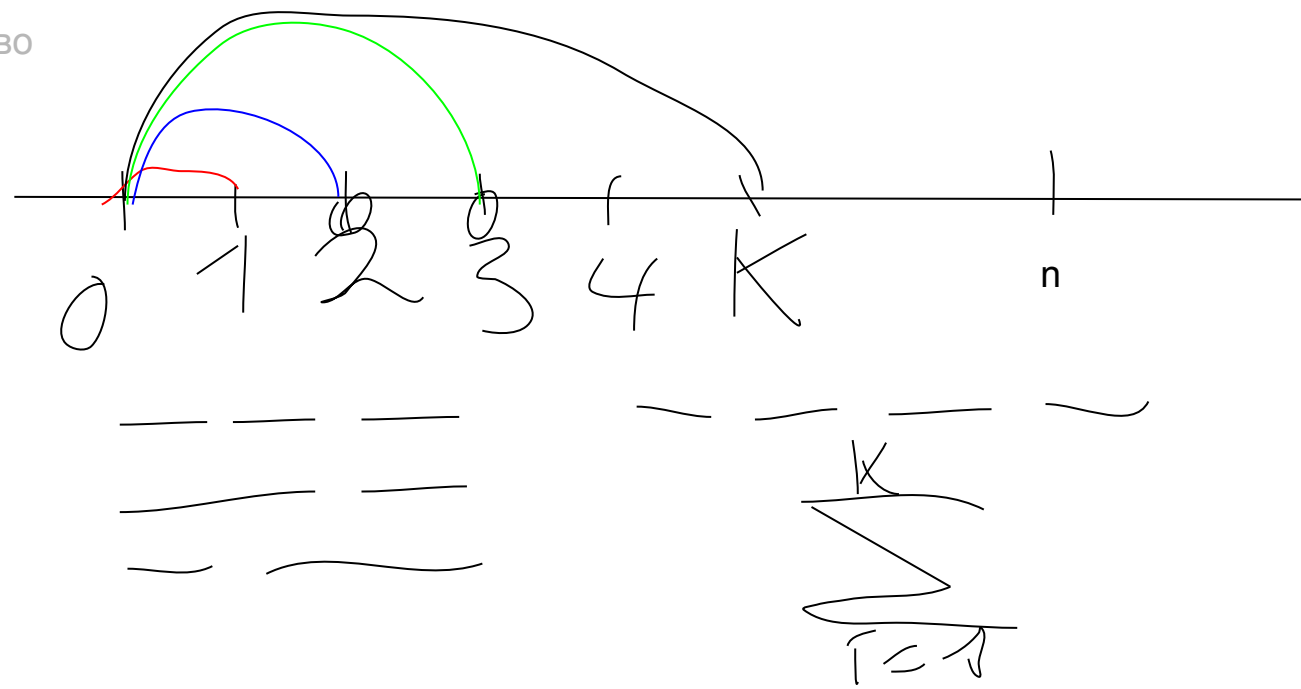


1. Какие значения мы вычисляем  $a[i]$  - количество способов допрыгать до  $i$ -ого столбца
2. Какое рекурсивное соотношение
3. Какие начальные значения  $a[0]=1$
4. В каком порядке вычисляются значения
5. Где искать ответ  $a[n]$



$ku(2)=2$   
 $ku(3)=3$   
 $ku(4)=5$   
 $ku(n)=ku(n-1)+ku(n-2)$

$ku(n)=ku(n-1)+ku(n-2)+ku(n-3)$

$ku(n)=ku(n-1)+ku(n-2)+ku(n-3)+\dots+ku(n-k)$

$ku(n)=\text{СУММА}[j=1;k] ku(n-i)$

Формула для больших  $n \geq k$

$k=5$   
 $ku(3)$

$ku(n)=\text{СУММА}[j=1;\min(k,n)] ku(n-i)$

Формула для всех  $n$

$ku(3)=\text{СУММА}[j=1;\min(5,3)] ku(n-i)=$   
 $=ku(3-1)+ku(3-2)+ku(3-3)$

динамическое программирование - это способ решения переборных задач (когда требуется что-то посчитать), когда основную задачу мы решать не умеем, но мы можем придумать рекурентное соотношение для перехода к меньшим задачам, которые мы можем все решить ровно по 1 разу и запомнить результаты их решения (в массив)

$ku(0)=1$  //ничего не делать - это 1

$ku(1)=1$   
 $ku(2)=2$   
 $ku(3)=4$

...  
 $ku(k-1)=? ku(0)+ku(1)+\dots+ku(k-2)=2^{(k-1)}$

$ku(k)=ku(k-1)+ku(n-2)+ku(n-3)+\dots+ku(k-k)$

$\text{cin} \gg n;$   
 $\text{mass}[1000]=\{\};$   
 $k=5;$   
 $\text{for}(i=0; i \leq n; i++)$

```

{
    mass[i]=0;
    r=min(k,i);
    for(j=1; j<=r; j++)
    {
        mass[i]+=mass[i-j];
    }
}

```

```

int rekku(int n)
{
    if(i>k)
    {
        sum=0;
        for(j=1; j<=r; j++)
        {
            sum+=rekku(n-j);
        }
        return sum;
    }
    else
    {
        r=min(k,n);
        sum=0;
        for(j=1; j<=r; j++)
        {
            sum+=rekku(n-j);
        }
        return sum;
    }
}

```

динамическое программирование сверху

динамическое программирование снизу

```

#include <iostream>

using namespace std;
void grass_hopper()
{
    int n,r;
    cin>>n;
    int mass[1000]={};
    int k=5;
    mass[0]=1;
    for(int i=1; i<=n; i++)
    {
        mass[i]=0;
        r=min(k,i);
        for(int j=1; j<=r; j++)
        {
            mass[i]+=mass[i-j];
        }
    }
    cout<<mass[n]<<endl;
}

```

```

int main()
{
    grass_hopper();
    return 0;
}

```