ЧИСЛА НА ФИБОНАЧИ – НИЯ МИТЕВА, ФН:62043

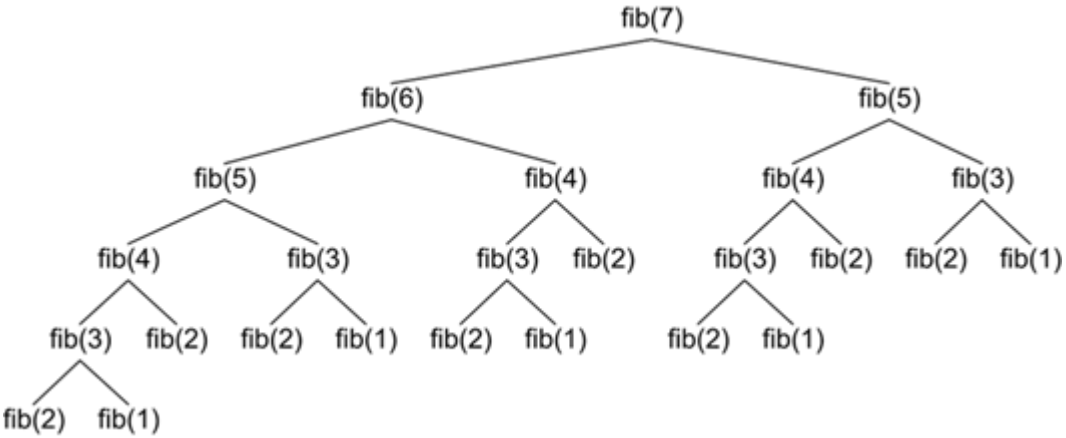
Задача 1.

```
unsigned fib(unsigned n)
{
    if (n < 2)
    {
        return 1;
    }
    return fib(n - 1) + fib(n - 2);
}
```

T(0) = T(1) = O(1)

T(n) = T(n-1) + T(n-2) + O(1), n>=2

Ако начертаем дърво на извикванията на рекурсивната функция:



Ще видим, дървото е с дълбочина n. От това следва, че функцията расте експоненциално, т.е. T(n) = O(2^n). Това се доказва с индукция.

При този алгоритъм се вижда, че се правят излишни изчисления, т.е. някои от членовете на редицата се пресмятат по няколко пъти.

Задача 2.

Нерекурсивна функция:

```
void fib2(unsigned n)
{
    unsigned fib = 1, f0 = 1, f1 = 1;
    for (int i = 0; i < n - 1; i++)
    {
        if (i == 0 || i == 1)
        {
            cout << fib << endl;
        }
        else
        {
            fib = f0 + f1;
            f0 = f1;
            f1 = fib;
            cout << f1 << endl;
        }
    }
}
```

Тогава ясно се вижда, че сложността е O(n)

Задача 3.

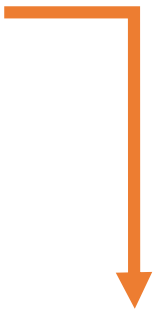
Решение на задачата със сложност O(log(n)) :

Квадратна матрица

https://en.wikipedia.org/wiki/Fibonacci_number#Matrix_form

http://www.ics.uci.edu/~eppstein/161/960109.html -> Алгоритъм 5

Решение в stackoverflow.com : https://stackoverflow.com/questions/16388982/algorithm-function-for-fibonacci-series



```
 void multiply(int F[2][2], int M[2][2]);

 void power(int F[2][2], int n);

 /* function that returns nth Fibonacci number */
int fib(int n)
 {
     int F[2][2] = { { 1,1 },{ 1,0 } };
     if (n == 0)
         return 0;
     power(F, n - 1);
     return F[0][0];
 }

 /* Optimized version of power() in method 4 */
void power(int F[2][2], int n)
 {
     if (n == 0 || n == 1)
         return;
     int M[2][2] = { { 1,1 },{ 1,0 } };

     power(F, n / 2);
     multiply(F, F);

     if (n % 2 != 0)
         multiply(F, M);
 }

void multiply(int F[2][2], int M[2][2])
 {
     int x = F[0][0] * M[0][0] + F[0][1] * M[1][0];
     int y = F[0][0] * M[0][1] + F[0][1] * M[1][1];
     int z = F[1][0] * M[0][0] + F[1][1] * M[1][0];
     int w = F[1][0] * M[0][1] + F[1][1] * M[1][1];

     F[0][0] = x;
     F[0][1] = y;
     F[1][0] = z;
     F[1][1] = w;
 }
```