**Домашна работа 1 - СДА**

Фн 62072 Николай Иванов

Зад.1 Рекурсивно търсене на n-тото число на Фибоначи чрез алгоритъм със сложност О(2n):

int FibOn2(int n)

{

if (n <= 1)

return n;

else

return FibOn2(n - 1) + FibOn2(n - 2);

}

int main()

{

int n;

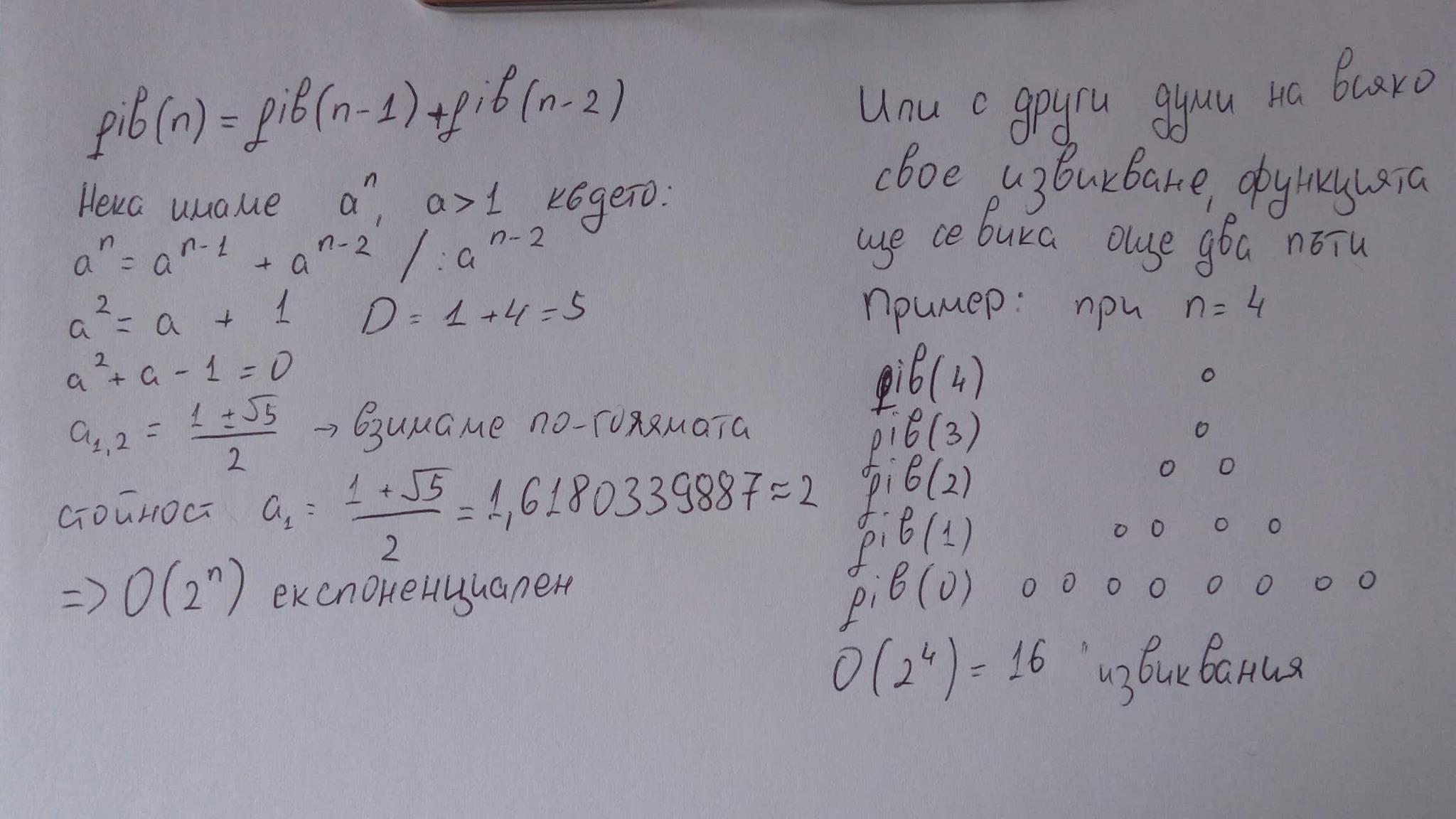
cin >> n;

cout << FibOn2(n) << endl;

return 0;

}

Изчисляване на времевата сложност на алгоритъма:



Зад.2 Търсене на n-тото число на Фибоначи със сложност О(n)

int fibOn(int term, int val, int prev)

{

if (term == 0) return prev;

if (term == 1) return val;

return fibOn(term - 1, val + prev, val);

}

int main()

{

int n;

cin >> n;

cout << fibOn(n, 1, 0) << endl;

return 0;

}

Сега броят на извикванията на рекурсивната функция е равен на броя на n, което води до намаляване на сложността до О(n).

Зад.3\* Със сложност O(logn) (чрез матрици)

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;

}

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);

}

int fibOlogn(int n)

{

int F[2][2] = { { 1,1 },{ 1,0 } };

if (n == 0)

return 0;

power(F, n - 1);

return F[0][0];

}

int main()

{

int n;

cin >> n;

cout << fibOlogn(n) << endl;

return 0;

}