**Practice\_1\_2.md**

# Найти сложность приведенного ниже соотношения:

𝑇(𝑛) = 3𝑇(𝑛 − 1), если 𝑛 > 0,

1, иначе

T (n) = {3T (n − 1),1,если n > 0,иначе

№1. T(n)=3T(n - 1)=3(3T(n – 2)) = 3(3(3T(n – 3))) = … = 3n T(n – n) = 3n \* 1 = 3n

-> O(3n)

# Найти сложность приведенного ниже соотношения:

𝑇(𝑛) = 2𝑇(𝑛 − 1)–1 если 𝑛 > 0,

1, иначе

T (n) = {2T (n − 1)–11,если n > 0,иначе

№2. T(n) = 2T(n – 1) – 1 = 4(2T(n – 3) – 1) – 2 – 1 = 8T (n - 3) – 4 – 2 – 1 = 2nT(n – n) – 2n – 1 - … - 20 = 2n – (2n – 1) = 1

-> O(1)

# Найти сложность приведенной ниже программы:

def funct(n):

if (n==1):

return

for i in range(1, n+1): #O(n)

#цикл будет выполняться 1 раз -> O(1)

for j in range(1, n + 1):

print("\*", end = "") break

print()

-> O(n)

# Найти сложность приведенной ниже программы:

def funct(n):

count = 0

for i in range(n//2, n+1):

#O(n)

j = 1

#O(log n)

while j <= n:

j = 2 \* j k = 1

#O(log n)

while k <= n:

k = 2 \* k count += 1

-> O(nlog2 n)

# Найти сложность приведенной ниже программы:

def funct(n):

count = 0

#O(n)

for i in range(n//2, n+1): j = 1

#O(n)

while j + n // 2 <= n: j += 1

k = 1

#O(log n)

while k <= n:

k = 2 \* k count += 1

-> O(n2\*log n)

# Найти сложность приведенной ниже программы:

def function(n): i = 1

s = 1

while s <= n: i += 1

s += i print('\*')

-> O(√n)

# Найти сложность приведенной ниже программы:

def function(n): count = 0

#O(n)

for i in range(n):

#O(n2)

for j in range(i, i \* i): if j % i == 0:

#O(n2)

for k in range(j): print('\*')

-> O(n5)

# Найти сложность приведенной ниже программы:

def function(n): i = 1

s = 1

while (s < n): s = s + i i+=1

-> O(√n)

# Найти сложность приведенной ниже программы:

def fun(n):

if (n < 5):

print("Sirius", end ="") else:

#O(n)

for i in range(n): print(i, end= " ")

-> O(n)

# Найти сложность приведенной ниже программы в

**лучшем и в худшем случае:**

def fun(a, b):

while (a != b):

if (a > b):

a = a - b else:

b = b – a

В лучшем случае: (при а = b) -> O(1)

В худшем случае: (при a или b = 1)-> O(max(a, b))

# Найти сложность приведенной ниже программы:

def fun(n):

i = 0

#O(√n)

while i\*i < n: print("Sirius") i += 1

-> O(√n)

# Найти сложность приведенной ниже программы:

def fun(n, x):

for i in range(1, n, i \* x): print("Sirius")

-> ???

# Найти сложность приведенной ниже программы:

import math def fun(n):

#O(n)

for i in range(0,math.floor(n/2)):

#O(n)

for j in range(1,n-math.floor(n/2)+1): k=1;

#O(log n)

for k in range(1,n+1,2\*k): print("Sirius");

-> O(n2\*log n)

# Найти сложность приведенной ниже программы:

def fun(n):

#O(n)

for i in range(1,n+1):

#O(n)

for j in range(1,n+1,i): print("Sirius");

-> O(n2)

# Найти сложность приведенной ниже программы:

def fun(n):

#O(n)

for i in range(n//3 + 1):

#O(n)

for j in range(1, n+1, 4): print("Sirius")

-> O(n2)

# Найти сложность приведенной ниже программы:

def fun(n):

i = 1

#O(log n)

while (i < n): j = n

#O(log n)

while (j > 0): j = j // 2

i = i \* 2

-> O(log2 n)

# Найти какое число сравнений будет сделано при выполнении следующего фрагмента кода?

def fun(n):

j = 1

while (j <= n): j = j \* 2

-> O(log n)

# Найти сложность приведенной ниже программы:

|  |  |  |
| --- | --- | --- |
| a = | 0 |  |
| b = for | 0  i | in range(N): |
|  | a | = a + random() |
| for | i b | in range(M):  = b + random()  -> O(N + M) |

1. **Найти сложность приведенной ниже программы:**

a = 0;

for i in range(N):

for j in reversed(range(i,N)): a = a + i + j;

-> O(n2)

# Найти сложность приведенной ниже программы:

k = 0;

#O(n)

for i in range(n//2,n):

#O(log n)

for j in range(2,n,pow(2,j)): k = k + n / 2;

-> #O(n log n)

# Найти сложность приведенной ниже программы:

a = 0 i = N

#O(log n)

while (i > 0): a += i

i //= 2

-> #O(log n)

# Найти сложность приведенной ниже программы:

for i in range(n): i = i \* k

-> O(n)

# Найти сложность приведенной ниже программы:

value = 0

for i in range(n):

for j in range(i): value = value + 1

-> ???