Audit Access Expires 5 de ago de 2020
You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise 6

Finger Exercises due Aug 5, 2020 20:30 -03

Exercise 6

3/4 points (graded)

ESTIMATED TIME TO COMPLETE: 6 minutes

Consider the following Python procedures. For each one, specify its order of growth.

1.
```
# Assume n has been previously bound to some value
i = 0
while i < 5:
    n *= 2
    i += 1

print(n)
```

O(1) ✔ **Answer:** O(1)

2.
```
def iterPower(a, b):
    result = 1
    while b > 0:
        result *= a
        b -= 1
    return result
```

O(b) ✔ **Answer:** O(b)

3.
```
def recurPower(a, b):
    print(a, b)
    if b == 0:
        return 1
    else:
        return a * recurPower(a, b-1)
```

O(b) ✔ **Answer:** O(b)

4.
```
def recurPowerNew(a, b):
    print(a, b)
    if b == 0:
        return 1
    elif b%2 == 0:
        return recurPowerNew(a*a, b/2)
    else:
        return a * recurPowerNew(a, b-1)
```

O(log(b)) **Answer:** O(log(b))

Enviar

Exercise 6

Topic: Lecture 11 / Exercise 6

Ocultar discussão

Add a Post

All Posts

Question 4

question posted about 11 hours ago by **mmesomaokafor22**

Anyone care to explain question 4?

This post is visible to everyone.

Add a Response

2 responses

paralogyx
about 10 hours ago

Just started to write, that I also didn't understand, and during that got it :)

If `b` is even, then `b%2 == 0` is `true` and we divide it by 2, so reduce amount of recursive calls in two times. That is for sure logarithmic. When `b` is odd, we call recursion with reducing in only by one - looks like linear, but not really: this call will be only once, as in next call `b` for sure will even again: (odd - 1 is always even). So, this part of algorithm is not dependent linearly of `b`

Add a comment

coleman85
about 8 hours ago

Thought maybe it was nlogn but that wasnt an option :D but thinking about it now, there are no two odd numbers in a row, so at most lets assume you get odd, even, odd, even, then it is still a log function, halving every third call.

(odd even odd even not sure that can even happen, write some numbers out, its more like odd, even, even, odd, even, even, even...)

Exactly correct, even calls will be of O(log(b)) -- you cannot say better e.g. put b=2**n. The worst case can be odd, even, odd, even, ... i.e. alternating parity of second argument in the function call so the calls of type recurPowerNew(a, b-1) will be at most equal to the number of calls of type recurPowerNew(a*a, b/2) and as a result are of O(log(b)). So, the combination too is O(log(b))

posted about 7 hours ago by **abhinav_vikram**

Add a comment

Exibindo todas as respostas

Add a response:

Pré-visualizar

Enviar