edX (https://www.edx.org)
MITx: 6.00.1x Introduction to Computer Science and Programming Using Python
sandipan_dey (/dashboard)
Courseware (/courses/MITx/6.00.1_4x/3T2014/courseware) Updates & News (/courses/MITx/6.00.1_4x/3T2014/info)
Calendar (/courses/MITx/6.00.1_4x/3T2014/89309559b0414f6d8cbef9e48ca19f4b/) Wiki (/courses/MITx/6.00.1_4x/3T2014/course_wiki)
iscussion (/courses/MITx/6.00.1_4x/3T2014/discussion/forum) Progress (/courses/MITx/6.00.1_4x/3T2014/progress)
Help

L9 PROBLEM 2 (5/5 points)

For the following programs, fill in the best-case and the worst-case number of steps it will take to run each program.

For these questions, you'll be asked to write a mathematical expression. Use +, -, / signs to indicate addition, subtraction, and division. Explicitly indicate multiplication with a * (ie say "6*n" rather than "6n"). Indicate exponentiation with a caret (^) (ie "n^4" for n^4). Indicate base-2 logarithms with the word log2 followed by parenthesis (ie "log2(n)").

1. Program 1:

```
def program1(x):
    total = 0
    for i in range(1000):
        total += i

    while x > 0:
        x -= 1
        total += x

    return total
```

What is the number of steps it will take to run Program 1 in the best case? Express your answer in terms of *n*, the size of the input `x`.

3003

3003

Answer: 3003

What is the number of steps it will take to run Program 1 in the worst case? Express your answer in terms of *n*, the size of the input `x`.

5*n+3003

$5 \cdot n + 3003$

Answer: 5*n + 3003

EXPLANATION:

In the best case scenario, `x` is less than or equal to 0. We first execute the assignment `total = 0` for one step. Next we execute the `for i in range(1000)` loop. This loop is executed 1000 times and has three steps (one for the assignment of `i` each time through the loop, as well as two for the `+=` operation) on each iteration. We next check if `x > 0` - it is not so we do not enter the loop. Adding one more step for the return statement, in the best case we execute 1 + 3*1000 + 1 + 1 = 3003 steps.

In the worst case scenario, `x` is a large positive number. In this case, we first execute the assignment `total = 0` for one step. Next we execute the first loop 1000 times (3000 total steps), then we execute the second loop (`while x > 0`) *n* times. This loop has five steps (one for the conditional check, `x > 0`, and two each for the `-=`

and `+=` operations). When we finally get to the point where `x = 0`, we execute the conditional check `x > 0` one last time - since it is not, we do not enter the loop. Adding one more step for the return statement, in the worst case we execute 1 + 3*1000 + 5*n + 1 + 1 = 5*n + 3003 steps.

2. Program 2:

```
def program2(x):
    total = 0
    for i in range(1000):
        total = i

    while x > 0:
        x /= 2
        total += x

    return total
```

What is the number of steps it will take to run Program 2 in the best case? Express your answer in terms of *n*, the size of the input `x`.

2003

2003

Answer: 2003

What is the number of steps it will take to run Program 2 in the worst case? Express your answer in terms of *n*, the size of the input `x`.

5*log2(n)+2003

$5 \cdot \log_2(n) + 2003$

Answer: 5*log2(n) + 2008

EXPLANATION:

In the best case scenario, `x` is less than or equal to 0. We first execute the assignment `total = 0` for one step. Next we execute the `for i in range(1000)` loop. This loop is executed 1000 times and has two steps (one for the assignment of `i` each time through the loop, as well as one for the `=` operation) on each iteration. We next check if `x > 0` - it is not so we do not enter the loop. Adding in one step for the return statement, in the best case we execute 1 + 2*1000 + 1 + 1 = 2003 steps.

In the worst case scenario, `x` is a large positive number. In this case we first execute the assignment `total = 0` for one step, then we execute the first loop 1000 times (2000 total steps). Finally execute the second loop (`while x > 0`) *log2(n)* + *1* times. **This is tricky!** Because we divide x by 2 every time through the loop, we only execute this loop a logarithmic number of times. `log2(n)` divisions of `x` by 2 will get us to `x = 1`; we'll need one more division to get `x <= 0`. Don't worry if you couldn't get this fact; we'll go through it a few more times in this unit.

This while loop has five steps (one for the conditional check, `x > 0`, and two each for the `-=` and `+=` operations). When we finally get to the point where `x = 0`, we execute the conditional check `x > 0` one last time - since it is not, we do not enter the loop. Adding in one step for the return statement, in the worst case we execute 1 + 2*1000 + 5*(log2(n) + 1) + 1 + 1 = 5*log2(n) + 2008 steps.

3. Program 3:

```
def program3(L):
    totalSum = 0
    highestFound = None
    for x in L:
        totalSum += x

    for x in L:
        if highestFound == None:
            highestFound = x
        elif x > highestFound:
            highestFound = x

    return (totalSum, highestFound)
```

What is the number of steps it will take to run Program 3 in the best case? Express your answer in terms of *n*, the number of elements in the list `L`.

3

3

Answer: 3

What is the number of steps it will take to run Program 3 in the worst case? Express your answer in terms of *n*, the number of elements in the list `L`.

7*n+2

$7 \cdot n + 2$

Answer: 7*n + 2

EXPLANATION:

In the best case scenario, `L` is an empty list. Thus we execute only the first two assignment statements, then the return statement. Therefore in the best case we execute 3 steps. Note that since the list is empty, no assignments are performed in the `for x in L` lines.

In the worst case scenario, `L` is a list with its elements sorted in increasing order (eg, `[1, 3, 5, 7, ...]`). In this case we execute the first two assignment statements (2 steps). Next we execute the first loop *n* times. This first loop has three steps (one for the assignment of `x` each time through the loop, as well as two for the `+=` operation), adding 3*n steps.

Finally we execute the second loop *n* times. The first time we execute this loop, we perform 3 steps - one for the assignment of `x`; then we run the check `if highestFound == None`, and finding it to be True, we execute the assignment `highestFound = x`.

The next (n-1) times we execute this loop, we perform 4 steps: one for the assignment of `x`, then we run the check `if highestFound == None`, and finding it to be False, we run the check `elif x > highestFound`. Since this is always True (the list is sorted in **increasing** order), we execute the assignment `highestFound = x`. Therefore in the second loop we execute 3 + (n-1)*4 = 3 + 4*n - 4 = 4*n - 1 steps.

Finally we execute the return statement, which is one more step.

Pulling this all together, we can see that in the worst case we execute 2 + 3*n + 4*n - 1 + 1= 7*n + 2 steps.

Reminder: You do not lose points for trying a problem multiple times, nor do you lose points if you hit "Show Answer". If this problem has you stumped after you've tried it a few times, feel free to reveal the solution.

Click the "Reset" button to clear your answers.

Check Reset Hide Answer

Show Discussion

New Post

EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

About & Company Info

About (https://www.edx.org/about-us)

News (https://www.edx.org/news)

Contact (https://www.edx.org/contact)

FAQ (https://www.edx.org/student-faq)

edX Blog (https://www.edx.org/edx-blog)

Donate to edX (https://www.edx.org/donate)

Jobs at edX (https://www.edx.org/jobs)

Follow Us

Twitter (https://twitter.com/edXOnline)

Facebook (http://www.facebook.com/EdxOnl

Meetup (http://www.meetup.com/edX-Global-Community)

LinkedIn (http://www.linkedin.com/compan

Google+ (https://plus.google.com/+edXOnli