University of Toronto - Computer Science - Image

Analyzing Algorithms

Up to now our focus has been on writing correct code. Next, we'll focus on:

- analyzing our algorithms, and
- determining the amount of time algorithms take to run.

Linear Running Time Functions

Function 1:

```
def print_ints(n):
    """ (int) -> NoneType

    Print the integers from 1 to n, inclusive.
    """

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

Output:

```
>>> print_ints(10)
1
2
3
4
5
6
7
8
9
10
```

When the function is called, the value that n refers to is equal to the number of items printed:

- when n refers to 10, it prints 10 integers
- when n refers to 20, it prints 20 integers
- when n refers to 40, it prints 40 integers

This graph shows the number of steps (print function calls) relative to the size of the input:

plot of linear runtime, steps = n

The number of steps is proportional to n.

Function 2:

```
def print_odd_ints(n):
    """ (int) -> NoneType

    Print the odd values from 1 to n, inclusive.
    """
```

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

Output:

```
>>> print_odd_ints(10)
1
3
5
7
9
```

When the function is called, the value that n refers to is twice the number of items printed:

- when n refers to 10, it prints 5 integers
- when n refers to 20, it prints 10 integers
- when n refers to 40, it prints 20 integers

We can plot this as well:
 plot of linear runtime, steps = n/2

The number of steps is still proportional to n.

Both function 1 and 2 are linear functions: the runtime grows linearly with respect to the size of input.

Quadratic Functions

Function 3:

```
def print_pairs(n):
    """ (int) -> NoneType

    Print all combinations of pairs of integers 1 to n,
        inclusive.
    """

    for i in range(1, n + 1):
        for j in range(1, n + 1):
            print(i, j)
```

Output 1:

```
>>> print_pairs(2)
1 1
1 2
2 1
2 2
```

The function call above prints 4 pairs of integers.

Output 2:

```
>>> print_pairs(3)
1 1
1 2
1 3
2 1
```

```
2 2
2 3
3 1
3 2
3 3
```

The function call above prints 9 pairs of integers.

Output 3:

```
>>>print_pairs(4)
1 1
1 2
1 3
1 4
2 1
2 2
2 3
2 4
3 1
3 2
3 3
3 4
4 1
4 2
4 3
4 4
```

The function call above prints 16 pairs of integers.

For argument n, the print function is called n^2 times.

When reading the code, notice that the `print` function call is inside a `for` loop, which loops n times. This `for` loop is also within a `for` loop that executes n times.

Therefore:
n *(iterations of the outside loop)* * **n** *(iterations of the inside loop)* = **n^2** *iterations*

Plotting the graph, we get:
plot of quadratic time, steps = n^2
Function 3 is quadratic: the runtime grows quadratically with respect to the size of input.

Logarithmic Functions

Function 1

```
def print_double_step(n):
    """ (int) -> NoneType

    Print integers from 1 to n inclusive, with an initial
    step size of 1 and each subsequent step size being
    twice as large as it was previously.
    """

    i = 1
    while i < n + 1:
        print(i)
        i = i * 2
```

In this example, the step size varies. It size of the first step is 1, the next step would be 2, then 4, 8, 16, 32...

How many values would be printed for different values of n?

Output 1:

```
>>> print_double_step(4)
1
2
4
```

When n refers to 4, 3 integers are printed.

Output 2:

```
>>> print_double_step(5)
1
2
4
```

When n refers to 5, 3 integers are printed.

It's not until n refers to 8 that more than 3 integers are printed.

Output 3:

```
>>> print_double_step(8)
1
2
4
8
```

When n refers to values 8 to 15, there will be 4 integers printed.

When n refers to 16, 5 integers are printed.
When n refers to 32, 6 integers are printed.
When n refers to 64, 8 integers are printed.

Starting with n referring to 1, each time that n is doubled, it prints one extra line.

This function is logarithmic and as the input size increases, the running time grows more slowly than for linear and quadratic functions.

plot of logarithmic time, steps = log(n)

Jennifer Campbell • Paul Gries
University of Toronto