Complexity Worksheet results for Ishan Mukherjee

Correct answers are hidden.

Score for this attempt: 5 out of 5

Submitted 21 Jan at 23:15

This attempt took 5 minutes.

Question 1

1 / 1 pts

What is the complexity of the `pearson` function, as a function of the length of its `input` vector?

```
def pythagoras(input):
    let x = 0
    let y = 0
    for elt in input:
        x = x + 1
        y = y + elt
    return y / x

def pow(n, k):
    let out = 0
    for i in range(k):
        out = out + n
    return n

def pearson(input):
    let w = 0
    for elt in input:
        let q = elt - pythagoras(input)
        w = w + pow(q, 4)
    return w
```

O(n^2)

Question 2

1 / 1 pts

What is the complexity of the `descartes` function, as a function of the length of either of its input vectors (`in1` and `in2` are the same length).

```
def descartes(in1, in2):
    assert in1.len() == in2.len()
    let out = [None; in1.len()]
    for i in range(in1.len()):
        out[i] = [None; in2.len()]
    for i in range(in1.len()):
        for j in range(in2.len()):
            out[i][j] = [in1[i], in2[j]]
    return out
```

O(n^2)

Question 3

1 / 1 pts

What is the complexity of the `fibonacci` function, as a function of its input `n`?

```
def different_pow(n, k):
    def helper(y, x, k):
        if k == 0: return y
        if k % 2 == 0: return helper(y, x*x, k//2)
        else: return helper(x*y, x*x, (k-1)//2)
    return helper(1, n, k)

def fibonacci(n):
    let phi = (1+(5).sqrt()) / 2
    let psi = (1-(5).sqrt()) / 2
    return (different_pow(phi, n) - different_pow(psi, n)) / (5).sqrt()
```

O(log n)

Question 4

1 / 1 pts

What is the complexity of the `sidewalk` function, as a function of the length of its input linked list?

```
def list_length(list):
    let length = 0
    while list is not None:
        length = length + 1
        list = list.next
    return length

def celery(n):
    if n == 1:
        return 0
    else:
        return celery(n // 2) + 1

def sidewalk(input):
    assert input is not None
    let bound = celery(list_length(input))
    while input is not None:
        if bound == 0:
            return input.data
        else:
            input = input.next
            bound = bound - 1
```

O(n)

Question 5

1 / 1 pts

What is the complexity of the `falafel` function, as a function of the length of its input linked list?

```
def filter(f, l):
    if l is None: return None
    if f(l.data): return cons(l.data, filter(f, l.next))
    else: return filter(f, l.next)

def map(f, l):
    if l is None: return None
    else: return cons(f(l.data), map(f, l.next))

def foldr(f, b, l):
    if l is None: return b
    else: return f(l.data, foldr(f, b, l.next))

def falafel(input):
    foldr(lambda x, y: x + y,
          0,
          map(lambda x: x * x,
              filter(lambda x: x % 2 == 0,
                     input)))
```

O(n)

Quiz score: 5 out of 5