Assignment_2

February 16, 2024

1 Assignment 2

Peder Ørmen Bukaasen, Bård Tollef Pedersen and Eivind Lid Trøen

1.1 Exercise 1

1.1.1 A

For all positive integers n, The sum of the first n square numbers is eaual to n(n+1)(2n+1) / 6

We assume that the formula holds for some arbitrary positive integer 12 + 22 + 32 + … + k2 = k(k+1)(2k+1) / 6

We need to show that the formula holds for n = k + 1,

12 + 22 + … + (k+1)2 = (k+1)((k+1)+1)(2(k+1)+1) / 6
LHS = RHS

Starting with the left-hand side (LHS), which is the sum of the first k+1 square numbers:

LHS => 12 + 22 + … + k2 + (k+1)2

We know that 12 + 22 + … + k2 can be expressed as k(k+1)(2k+1) / 6 (our inductive hypothesis).
So, the LHS becomes:
LHS = k(k+1)(2k+1) / 6 + k2
LHS = (k(k+1)(2k+1) + 6(k+1)2) / 6
LHS = ((k+1)(k(2k+1) + 6(k+1))) / 6
LHS = ((k+1)(2k^2 + k + 6k + 6)) / 6
LHS = ((k+1)(2k^2 + 7k + 6)) / 6
LHS = ((k+1)(k+2)(2k+3)) / 6

RHS = (k+1)((k+1)+1)(2(k+1)+1) / 6
RHS = ((k+1)(k+2)(2k+3)) / 6

The LHS is the same as the right-hand side (RHS). Thus, we've shown that if the formula holds for k, it also holds for k + 1.

By the principle of mathematical induction, the formula holds for all positive integers n. Therefore, we've proven that the sum of the first n square numbers is indeed (n(n+1)(2n+1)) / 6.

1.1.2 B

For all integers n, if n3 + 5 is odd then n is even.

Some ground rules:
When two even numbers are multiplied it becomes even.
When two odd numbers are multiplied it becomes odd.
When even and odd are multiplied it becomes even.

Lets say that n is even, and that measn n+1 is odd.
We can then test the statement for even numbers, n.
n3 + 5 = n * n * n + 5 => n + 5 => n + n + 1 => n + 1 => odd

No test if n is an odd number as n+1.
(n+1)*(n+1)*(n+1) + 5 => n3 + 3*n2 + 3*n + 6

n3 => n * n * n => n
3 * n2 => 3 * n * n => n
3 * n => n
6 => n

n3 + 3*n2 + 3*n + 6 => n + n + n + n => 4 * n => n, wich is even.

The result of n3 + 5 is the oppisit of n, so if the result is odd then n is even, and vice versa.

1.2 Exercise 2

First function takes four arguments instead of 3, but loops each element max once.

[ ]:
```python
def recursively_search(search_list, value, index=0, start_index=0): # Maybe reomve the start_index from the function call

    # If the current index is the value, return the index
    if search_list[index] == value:
        return index

    # If the list is at the end, begin at the start
    if index == len(search_list) -1:
        index = -1

    # If the element is not found, return -1
    index += 1
    if index == start_index:
        return -1

    return recursively_search(search_list, value, index, start_index)
```

[ ]:
```python
# Test in python to see that it works:

list_number = [1, 2, 3, 4, "value", "test", 2.5]
start = 1
start_index = start
value = "value"
```

```
recursively_search(list_number, value, start, start_index) # Maybe reomve the␣
↪start_index from the function call
```

[ ]: 4

This function has only 3 inputs, but in worst case it loops entire list twice.

[ ]:
```
def recursively_search1(search_list, value, index=0):
    print(search_list, index)

    temp_index = index

    if temp_index >= len(search_list):
        temp_index = index - len(search_list)

    if temp_index == len(search_list):
        return -1

    if search_list[temp_index] == value:
        return temp_index

    index += 1
    return recursively_search1(search_list, value, index)
```

[ ]:
```
# Test in python to see that it works:
#              0  1  2  3  4     5        6     7
list_number = [0, 1, 2, 3, 4, "value", "test", 2.5]
start = 0
value = -1
recursively_search1(list_number, value, start)
```

```
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 0
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 1
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 2
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 3
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 4
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 5
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 6
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 7
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 8
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 9
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 10
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 11
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 12
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 13
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 14
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 15
[0, 1, 2, 3, 4, 'value', 'test', 2.5] 16
```

[ ]: -1

1.3 Exercise 3

From the lecture,
"Sn = 1/2 * n(2a + (n − 1)d), where a is the starting value, d is the constant, n is the number of terms"

With the list [-4, -1, 2, 5, … up to 20 term],
we then get:
a = -4
d = 3
n = 20

Sn = 1/2 * n * (2 * a + (n − 1) * d)
Sn = 1/2 * 20 * (2*-4 + (20-1)*3)
Sn = 1/2 * 20 * (-8 + 57)
Sn = 1/2 * 980
Sn = 490

[ ]:
```
# Test in python to see that the answers are the same:

Sn = 0
value = -4
for _ in range(20):
    Sn += value
    value += 3
Sn
```

[ ]: 490