Homework #2

Analysis of Algorithms

Illya Starikov

Due Date: September 11^{th}, 2017

Question #1

We define $Q = \{q | q + n = n + k\}$, and $n \in Z^+$. Given that $n + 0 = n = 0 + n$

$$\begin{aligned} q + (k+1) &= (q+k) + 1 \\ &= (k+q) + 1 \\ &= k + (1+k) \\ &= (k+1) + m \\ &\therefore q \in Q \\ &\therefore n + q = q \end{aligned}$$

Question #2

Theorem 1. $\forall i \in \mathbb{Z}^+$,

$$\sum_{i=1}^{n} \frac{i^2}{(2i-1)(2i+1)} = \frac{n(n+1)}{2(2n+1)} \tag{1}$$

Proof. We will use the principle of recursion to solve this problem.

Step #1 Our problem states for an arbitrary integer i, summed to an arbitrary integer n, we have the summation (we denote by $f(x)$) and explicit forms ($g(x)$) described by Equation 1. We take the stopping value to be 0.

Step #2 Checking two values is trivial; take 1 and 2.

$$\sum_{i=1}^{1} \frac{i^2}{(2i-1)(2i+1)} = \frac{1(1+1)}{2(2*1+1)} = \frac{1}{3}$$
$$\sum_{i=1}^{n} \frac{i^2}{(2i-1)(2i+1)} = \frac{2*(2+1)}{2(2*2+1)} = \frac{3}{5}$$

To check the stopping value, we check the 0th value. We clearly see that $f(x) = g(x) = 0$.

Step #3 If n in $\mathbb{Z}^+$ triggers a recursive call, then $n > 0$. The only value used in the call is $n-1$, which is in $\mathbb{Z}$ and greater than or equal to 0, because it is an integer and $n1 \geq 0$ since $n > 0$.

Step #4 We use the integer n as the counter. When recursion is called the function is called with the value $n-1$. The counter strictly decreases and the recursion halts.

Step #5 To prove recursion stops, we take the following:

$$\begin{aligned}
f(n) &= \sum_{i=1}^{n} \frac{i^2}{(2i-1)(2i+1)} && \text{definition} \\
&= f(n-1) + \frac{n^2}{(2n-1)(2n+1)} && \text{recursive structure} \\
&= g(n-1) + \frac{n^2}{(2n-1)(2n+1)} && \text{by our assumption} \\
&= \frac{(n-1)((n-1)+1)}{2(2(n-1)+1)} + \frac{n^2}{(2n-1)(2n+1)} && \text{replacing } g(n) \text{ with definition} \\
&= \frac{n(n+1)}{2(2(2n+1))} && \text{algebraic simplification} \\
&= g(n)
\end{aligned}$$

Because $f(n) = g(n)$, the property is inherited recursively.

Step #6 Since Steps 1–5 have been verified, it follows from the Principle of Recursion that P holds for all values in $\mathbb{Z}^+$, i.e., $f(n) = g(n), \forall n \in \mathbb{Z}, n \geq 0$

□

Table 1: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	**Explicit Values**	**Difference**
0.333 333 333 333	0.333 333 333 333	0.000 000 000 000 000
0.600 000 000 000	0.600 000 000 000	0.000 000 000 000 000
0.857 142 857 143	0.857 142 857 143	0.000 000 000 000 000
1.111 111 111 111	1.111 111 111 111	0.000 000 000 000 000
1.363 636 363 636	1.363 636 363 636	0.000 000 000 000 000
1.615 384 615 385	1.615 384 615 385	0.000 000 000 000 000
1.866 666 666 667	1.866 666 666 667	0.000 000 000 000 000
2.117 647 058 824	2.117 647 058 824	0.000 000 000 000 000
2.368 421 052 632	2.368 421 052 632	0.000 000 000 000 000
2.619 047 619 048	2.619 047 619 048	0.000 000 000 000 000
2.869 565 217 391	2.869 565 217 391	0.000 000 000 000 000
3.120 000 000 000	3.120 000 000 000	0.000 000 000 000 001
3.370 370 370 370	3.370 370 370 370	0.000 000 000 000 001
3.620 689 655 172	3.620 689 655 172	0.000 000 000 000 001
3.870 967 741 935	3.870 967 741 935	0.000 000 000 000 001
4.121 212 121 212	4.121 212 121 212	0.000 000 000 000 001
4.371 428 571 429	4.371 428 571 429	0.000 000 000 000 001
4.621 621 621 622	4.621 621 621 622	0.000 000 000 000 002
4.871 794 871 795	4.871 794 871 795	0.000 000 000 000 001
5.121 951 219 512	5.121 951 219 512	0.000 000 000 000 002
5.372 093 023 256	5.372 093 023 256	0.000 000 000 000 002
5.622 222 222 222	5.622 222 222 222	0.000 000 000 000 002
5.872 340 425 532	5.872 340 425 532	0.000 000 000 000 003
6.122 448 979 592	6.122 448 979 592	0.000 000 000 000 003
6.372 549 019 608	6.372 549 019 608	0.000 000 000 000 003
6.622 641 509 434	6.622 641 509 434	0.000 000 000 000 004
6.872 727 272 727	6.872 727 272 727	0.000 000 000 000 004
7.122 807 017 544	7.122 807 017 544	0.000 000 000 000 004
7.372 881 355 932	7.372 881 355 932	0.000 000 000 000 004
7.622 950 819 672	7.622 950 819 672	0.000 000 000 000 004
7.873 015 873 016	7.873 015 873 016	0.000 000 000 000 003
8.123 076 923 077	8.123 076 923 077	0.000 000 000 000 004
8.373 134 328 358	8.373 134 328 358	0.000 000 000 000 004
8.623 188 405 797	8.623 188 405 797	0.000 000 000 000 004
8.873 239 436 620	8.873 239 436 620	0.000 000 000 000 004
9.123 287 671 233	9.123 287 671 233	0.000 000 000 000 004
9.373 333 333 333	9.373 333 333 333	0.000 000 000 000 004
9.623 376 623 377	9.623 376 623 377	0.000 000 000 000 004
9.873 417 721 519	9.873 417 721 519	0.000 000 000 000 004

Question #3

Theorem 2. *For an arbitrary $n \in Z$, such that $n \geq 0$,*

$$2^n \leq fib(n) \leq 2^{\frac{n}{2}}$$

Proof. *Step #1* Prove by induction that for $n > 0$

$$2^n \leq fib(n) \leq 2^{\frac{n}{2}}$$

Step #2 Check the base case and two other values,

$$2^0 \leq fib(0) \leq \leq 2^0 2^1 \leq fib(1) \leq \leq 2^{\frac{1}{2}} 2^2 \leq fib(2) \leq \leq 2^1$$

Step #3 To prove the recursive case, we take the following.

$$\begin{aligned} 2^n \leq fib(n) &\leq 2^{\frac{n}{2}} \\ 2^{n-1} \leq fib(n) + fib(n-1) &\leq 2^{\frac{n-1}{2}} \\ \frac{2^n}{2} \leq fib(n) + fib(n-1) &\leq \frac{2^{\frac{n}{2}}}{\sqrt{2}} \end{aligned}$$

Because we know $\frac{2^n}{2} \leq 2^n$ and $\frac{2^{\frac{n}{2}}}{\sqrt{2}} \leq 2^{\frac{n}{2}}$, we know that $f(n)$ must be bounded by those function.

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows, namely, for all natural numbers, $f(n) \mod 133 = 0$.

□

Question #4

Upon inspection, it's quite apparent that the series is the sum of all natural numbers, with every other number being negative. More formally,

$$\sum_{q=1}^{n} (-1)^{q-1} q^2 = (-1)^{n-1} \frac{n * (n+1)}{2}$$

The results can be summarized below.

Table 2: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	Explicit Values	Sum of Numbers
1.0000000000	1.0000000000	1.000000000000000
-3.0000000000	-3.0000000000	3.000000000000000
6.0000000000	6.0000000000	6.000000000000000
-10.0000000000	-10.0000000000	10.000000000000000
15.0000000000	15.0000000000	15.000000000000000
-21.0000000000	-21.0000000000	21.000000000000000
28.0000000000	28.0000000000	28.000000000000000
-36.0000000000	-36.0000000000	36.000000000000000
45.0000000000	45.0000000000	45.000000000000000
-55.0000000000	-55.0000000000	55.000000000000000
66.0000000000	66.0000000000	66.000000000000000
-78.0000000000	-78.0000000000	78.000000000000000
91.0000000000	91.0000000000	91.000000000000000
-105.0000000000	-105.0000000000	105.000000000000000
120.0000000000	120.0000000000	120.000000000000000
-136.0000000000	-136.0000000000	136.000000000000000
153.0000000000	153.0000000000	153.000000000000000
-171.0000000000	-171.0000000000	171.000000000000000
190.0000000000	190.0000000000	190.000000000000000
-210.0000000000	-210.0000000000	210.000000000000000
231.0000000000	231.0000000000	231.000000000000000
-253.0000000000	-253.0000000000	253.000000000000000
276.0000000000	276.0000000000	276.000000000000000
-300.0000000000	-300.0000000000	300.000000000000000
325.0000000000	325.0000000000	325.000000000000000
-351.0000000000	-351.0000000000	351.000000000000000
378.0000000000	378.0000000000	378.000000000000000
-406.0000000000	-406.0000000000	406.000000000000000
435.0000000000	435.0000000000	435.000000000000000
-465.0000000000	-465.0000000000	465.000000000000000
496.0000000000	496.0000000000	496.000000000000000
-528.0000000000	-528.0000000000	528.000000000000000
561.0000000000	561.0000000000	561.000000000000000
-595.0000000000	-595.0000000000	595.000000000000000
630.0000000000	630.0000000000	630.000000000000000
-666.0000000000	-666.0000000000	666.000000000000000
703.0000000000	703.0000000000	703.000000000000000
-741.0000000000	-741.0000000000	741.000000000000000
780.0000000000	780.0000000000	780.000000000000000

Question #5

Question #6

```python
def depth(n):
    if n < 2:
        return 1
    if n % 2 == 1:
        return 1 + depth(3 * n + 1)
    else:
        return 1 + depth(n // 2)


def main():
    for i in range(101):
        print("%3.0f & %5.0f\\\\" % (i, depth(i)))


if __name__ == "__main__":
    main()
```

Iteration	Depth
0	1
1	1
2	2
3	8
4	3
5	6
6	9
7	17
8	4
9	20
10	7
11	15
12	10
13	10
14	18
15	18
16	5
17	13
18	21
19	21
20	8
21	8
22	16
23	16
24	11
25	24
26	11
27	112
28	19
29	19
30	19
31	107
32	6
33	27
34	14
35	14
36	22
37	22
38	22
39	35
40	9
41	110
42	9
43	30
44	17
45	17
46	17
47	105
48	12
49	25
50	25
51	25
52	12
53	12
54	113
55	113
56	20
57	33
58	20
59	33
60	20
61	20
62	108
63	108
64	7
65	28
66	28
67	28
68	15
69	15
70	15
71	103
72	23
73	116
74	23
75	15
76	23
77	23
78	36
79	36
80	10
81	23
82	111
83	111
84	10
85	10
86	31
87	31
88	18
89	31
90	18
91	93
92	18
93	18
94	106
95	106
96	13
97	119
98	26
99	26
100	26

Question #7

Theorem 3. *For an arbitrary $n \in Z$, such that $n \geq 0$,*

$$\sum_{i=1}^{n} = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Proof. *Step #1* Prove by induction that for $n = 0$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

We choose the summation form as $f(x)$ and the explicit form $g(x)$.

Step #2 Check the base case and two other values,

$$\sum_{i=1}^{1} n^2 = 1 = \frac{1(1+1)(2*1+1)}{6}$$
$$\sum_{i=1}^{2} n^2 = 5 = \frac{1(1+1)(2*1+1)}{6}$$
$$\sum_{i=1}^{3} n^2 = 14 = \frac{2*(2+1)(2*2+1)}{6}$$

Step #3 $\forall n > 0$, prove that $P(n)$ is true if $P(n-1)$ is true,

$$\begin{aligned}
f(x) &= \sum_{i=1}^{n} i^2 && \text{by definition}\\
&= f(x-1) + n^2 && \text{by recursion}\\
&= g(x-1) + n^2 && \text{by hypothesis}\\
&= \frac{(n-1)(n-1+1)(2(n+1)+1)}{6} + n^2\\
&= \frac{n(n+1)(2n+1)}{6} && \text{algebraic simplification}
\end{aligned}$$

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows,namely, for all natural numbers, $f(n) = g(n)$.

□

Question #8

Theorem 4. *For an arbitrary $n \in Z$, such that $n \geq 0$,*

$$11^{n+2} + 12^{2n+1} \mod 133 = 0$$

Proof. *Step #1* Prove by induction that for $n > 0$

$$11^{n+2} + 12^{2n+1} \mod 133 = 0$$

Step #2 Check the base case and two other values,

$$\begin{aligned} 11^3 + 12^3 &\mod 133 = 0 \\ 11^4 + 12^5 &\mod 133 = 0 \end{aligned}$$

Step #3 $\forall n > 0$, prove that $P(n)$ is true if $P(n-1)$ is true,

$$\begin{aligned} f(x) &= 11^{n+2} + 12^{2n+1} \\ &= 11^{n+1} + 12^{2n-1} + 11^{n+2} + 12^{2n+1} \end{aligned}$$

Because we assume $11^{n+2} + 12^{2n+1}$ to be divisible, we can rewrite it to be $133a_1$, signifying it is evenly divisible by 133.

$$\begin{aligned} f(x) &= 11 * 11^n + 12^{-1} * 144^n + 133a_1 \\ &= a_2 11^n + a_3 144^n + 133a_1 \end{aligned}$$

We know the superposition of all of these numbers to be divisible by 133.

Step #4 Since the hypotheses of Simple Induction are true, the conclusion follows, namely, for all natural numbers, $f(n) \mod 133 = 0$.

□

Question #9

Let $F(0)$ denote the root of the tree T. If we start at $F(0)$, there must be an adjacent vertex to it, namely, v_1. There are infinitely many vertexes by going through the tree (this must be the case, or the tree T would be finite).

We can repeat this process, for at any vertex v_n, there exists a vertex v_{n+1}. Because the tree is infinite, there are infinitely many v_{n+1}.

By induction, for any n, there exists a path of length L in T.

Question #10

```
class Binary_Search_Tree:
    def __init__(self, data):
        if data is None:
            self.data = [0, [], []]
        else:
            self.data = data
```

```python

    def add_node(self, node):
        if node.data[0] < self.data[0]:
            self.__add_node_left(node)
        else:
            self.__add_node_right(node)

    def __add_node_left(self, node):
        if self.data[1] == []:
            self.data[1] = node
        else:
            self.data[1].add_node(node)

    def __add_node_right(self, node):
        if self.data[2] == []:
            self.data[2] = node
        else:
            self.data[2].add_node(node)

    def leaf_count(self):
        if self.data[1] == []:
            if self.data[2] == []:
                return 1
            else:
                return self.data[2].leaf_count()
        elif self.data[2] == []:
            return self.data[1].leaf_count()
        else:
            return self.data[1].leaf_count() + self.data[2].leaf_count()

    def internal_count(self):
        if self.data[1] == []:
            if self.data[2] == []:
                return 0
        elif self.data[2] == []:
            return 1 + self.data[1].internal_count()
        else:
            return 1 + self.data[1].internal_count() + self.data[2].internal_count()


def main():
    T = Binary_Search_Tree([
        42,
        Binary_Search_Tree([29, [], []]),
        Binary_Search_Tree([51, [], []])
    ])

    print("Leaf: {0}, Internal: {1}".format(T.internal_count(), T.leaf_count()))

    T.add_node(Binary_Search_Tree([25, [], []]))
```

```
    T.add_node(Binary_Search_Tree([22, [], []]))
    T.add_node(Binary_Search_Tree([45, [], []]))
    T.add_node(Binary_Search_Tree([8, [], []]))
    T.add_node(Binary_Search_Tree([1000, [], []]))
    T.add_node(Binary_Search_Tree([3, [], []]))
    T.add_node(Binary_Search_Tree([1, [], []]))

    print("Leaf: {0}, Internal: {1}".format(T.internal_count(), T.
   leaf_count()))


if __name__ == "__main__":
    main()
```

From the results, we can very much see that Internal Nodes = Leaf Nodes − 1.

10.1 Proof

Proof. Take n to be the number of nodes, $I(T)$ the number of internal nodes and $L(T)$ to be the number of leaf nodes. We know the internal nodes to be the sum of previous internal nodes (i.e., $I(t) = I(t_1) + I(t_2) + 1$).

From this, we get

$$
\begin{aligned}
I(T) &= I(t_1) + I(t_2) + 1 && (2)\\
I(t_1) &= L(t_1) - 1 && (3)\\
I(t_2) &= L(t_2) - 1I(T) \quad = L(t_1) - 1 + L(t_2) - 1 && (4)\\
&= L(T) - 1 && (5)
\end{aligned}
$$

□