DSC 20
Discussion Section 6

ARDA CANKAT BATI

Today's Plan

1. General Notes On Recursion
2. Recursion examples
3. Talking about HW06 (if we have time)

Recursion Notes

1. (Almost) everything you can do with recursion, you can do with for loops
2. However some problems are much easier and straightforward to solve with recursions
3. Some questions might be actually (almost) impossible to do with for loops, but easy with recursion.

Recursion Notes

Some problems that are recursive in nature

1. Given a string, print all possible palindromic partitions
2. Check if a number is Palindrome
3. Print all possible combinations of elements in a given array of size n (power set)
4. Print all increasing sequences of length k from first n natural numbers
5. Print all leaf nodes of a Binary Tree (Binary Tree Search)
6. and many more...

https://www.geeksforgeeks.org/recursion-practice-problems-solutions/

Recursion examples

```
print(minus_plus(1))
```

```
-
```

```
print(minus_plus(2))
```

```
-
++
```

```
print(minus_plus(5))
```

```
-
++
---
++++
-----
```

Recursion examples

```
print(minus_plus(1))
```

```
-
```

```
print(minus_plus(2))
```

```
-
++
```

```
print(minus_plus(5))
```

```
-
++
---
++++
-----
```

```
def minus_plus(n):
    if n == 0:
        return ''
```

Recursion examples

```
def minus_plus(n):
    if n == 0:
        return ''
    if n % 2 == 0:
        return minus_plus(n - 1) + '+' * (n) + '\n'
    if n % 2 == 1:
        return minus_plus(n - 1) + '-' * (n) + '\n'
```

Recursion examples

```
print(minus_plus_star(1))
```

```
-
```

```
print(minus_plus_star(2))
```

```
-
++
```

```
print(minus_plus_star(6))
```

```
-
++
***
----
+++++
******
```

Recursion examples

```python
def minus_plus_star(n):
    parity_fac = 3
    if n == 0:
        return ''
    if n % parity_fac == 0:
        return minus_plus_star(n - 1) + '*' * (n) + '\n'
    if n % parity_fac == 1:
        return minus_plus_star(n - 1) + '-' * (n) + '\n'
    if n % parity_fac == 2:
        return minus_plus_star(n - 1) + '+' * (n) + '\n'
```

Recursion examples

```
print(minus_plus_starV2(6))
```

```
-
++
***
----
+++++
******
```

```
symbols = ['*', '-', '+']
parity_fac = 3
def minus_plus_starV2(n):
    """
    Uses only a single if statement to
    solve the problem
    """
    if n == 0:
        return ''
```

Recursion examples

```python
symbols = ['*', '-', '+']
parity_fac = 3
def minus_plus_starV2(n):
    """
    Uses only a single if statement to
    solve the problem
    """
    if n == 0:
        return ''
    index = n % parity_fac
    return minus_plus_star(n - 1) + symbols[index] * (n) + '\n'
```

```python
print(minus_plus_starV2(6))
```

```
-
++
***
----
+++++
******
```

Recursion examples

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

may simply be written as

$$a + ar + ar^2 + ar^3 + \cdots$$, with $a = \frac{1}{2}$ and $r = \frac{1}{2}$.

Recursion examples

```
# a
geo_sum(a = 1/2, r = 1/2, n = 1)
```

```
0.5
```

```
# a + a * (r ** 1)
geo_sum(a = 1/2, r = 1/2, n = 2)
```

```
0.75
```

```
# a + a * (r ** 1) + a * (r ** 2)
geo_sum(a = 1/2, r = 1/2, n = 3)
```

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

may simply be written as

$$a + ar + ar^2 + ar^3 + \cdots$$, wi

Recursion examples

```python
def geo_sum(a, r, n):
    assert isinstance(n, int)
    assert n > 0
    if n == 1:
        return a
```

$$\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots$$

may simply be written as

$$a+ar+ar^2+ar^3+\cdots$$, wi

Recursion examples

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

may simply be written as

$$a + ar + ar^2 + ar^3 + \cdots$$, wi

```
def geo_sum(a, r, n):
    assert isinstance(n, int)
    assert n > 0
    if n == 1:
        return a
    else:
        return a * (r ** (n - 1)) + geo_sum(a, r, n - 1)
```

Recursion examples

```
p_tri(3, 1)
```

3

```
p_tri(3, 2)
```

3

```
p_tri(5, 3)
```

10

```
p_tri(5, 4)
```

5

```
p_tri(6, 3)
```

20

```
p_tri(7, 3)
```

35

$$
\begin{array}{c}
1 \\
1 \quad 1 \\
1 \quad 2 \quad 1 \\
1 \quad 3 \quad 3 \quad 1 \\
1 \quad 4 \quad 6 \quad 4 \quad 1 \\
1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \\
1 \quad 6 \quad 15 \quad 20 \quad 15 \quad 6 \quad 1 \\
1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1
\end{array}
$$

Recursion examples

```
p_tri(0, -1)
```

0

```
p_tri(0, 0)
```

1

```
p_tri(0, 1)
```

0

```
p_tri(3, -1)
```

0

```
p_tri(3, 0)
```

1

```
p_tri(3, 4)
```

0

```
0
0 1 0
0 1 1 0
0 1 2 1 0
0 1 3 3 1 0
0 1 4 6 4 1 0
0 1 5 10 10 5 1 0
0 1 6 15 20 15 6 1 0
0 1 7 21 35 35 21 7 1 0
```

Recursion examples

```
def p_tri(row, column):

    # Base Cases
    if column < 0:
        return ?
    elif column > row:   row - 1
        return ?                                  row - 1
    elif column == 0 or column == row:
        return ?
    elif row == 0:
        return ?
```

0
0 1 0
0 1 1 0
0 1 2 1 0
0 1 3 3 1 0
0 1 4 6 4 1 0
0 1 5 10 10 5 1 0
0 1 6 15 20 15 6 1 0
0 1 7 21 35 35 21 7 1 0

Recursion examples

```
def p_tri(row, column):

    # Base Cases
    if column < 0:
        return 0
    elif column > row:   row - 1
        return 0
                                               row - 1
    elif column == 0 or column == row:
        return 1
    elif row == 0:
        return 0
```

0

0 1 0

0 1 1 0

0 1 2 1 0

0 1 3 3 1 0

0 1 4 6 4 1 0

0 1 5 10 10 5 1 0

0 1 6 15 20 15 6 1 0

0 1 7 21 35 35 21 7 1 0

Recursion examples

```
def p_tri(row, column):

    # Base Cases
    if column < 0:
        return 0
    elif column > row:  row - 1
        return 0
    elif column == 0 or column == row:   row - 1
        return 1
    elif row == 0:
        return 0

    # Recursion
    else:
        return p_tri(row - 1, column) + p_tri(row - 1, column - 1)
```

Recursion examples

Let n represent the row and k represent the column. We know that $\binom{n}{0} = 1$ and $\binom{1}{k} = k$. To compute the diagonal containing the elements $\binom{n}{0}$, $\binom{n+1}{1}$, $\binom{n+2}{2}$, . . ., we can use the formula

$$\binom{n+k}{k} = \binom{n+k-1}{k-1} \times \frac{n+k}{k}, \; k > 0.$$

To calculate the diagonal ending at $\binom{7}{2}$, the fractions are $\frac{6}{1}$, $\frac{7}{2}$, and the elements are

$$\begin{aligned}\binom{5}{0} &= 1,\\ \binom{6}{1} &= \binom{5}{0} \times \frac{6}{1} = 1 \times \frac{6}{1} = 6,\\ \binom{7}{2} &= \binom{6}{1} \times \frac{7}{2} = 6 \times \frac{7}{2} = 21.\end{aligned}$$

Recursion examples

```
def p_tri(row, column):
    # Base Cases
    if column < 0:
        return 0
    elif column > row: row - 1
        return 0
    elif column == 0 or column == row: row - 1
        return 1
    elif row == 0:
        return 0
```

Recursion examples

```
def p_tri(row, column):
    # Base Cases
    if column < 0:
        return 0
    elif column > row:  row - 1
        return 0
    elif column == 0 or column == row:   row - 1
        return 1
    elif row == 0:
        return 0

    # Recursion
    return int(p_tri(row - 1, column - 1) * (row / column))
```