Recursion

- Recursion is a programming technique where a function calls itself. Such function is called recursive function.

Why do we need recursion ?

Recursion provides a way to systematically break one big problem into smaller instances of the problems. We can define the solution to problems using recurrence relation.

It makes the code very concise and easy to understand.

Why is recursion important for Dynamic programming ?

First step in solving DP problems from scratch is to come up with a naive recursive solution and them optimize optimize it with dynamic programming techniques.

Dynamic programming = Recursion + Caching

Examples

1. Given an array of integers, write a function which uses recursion to find the maximum.

Note: In case of Python, the input will be list of integers.

Example

Input=[4,3,6,7,0,9,2] Output=9

A = [4,3,6,7,0,9,2]

↑

Max = -1

A = [4,3,6,7,0,9,2]

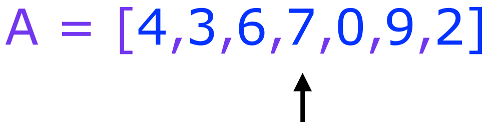
↑

Max = 4

A = [4,3,6,7,0,9,2]

↑

Max = 4



A = [4,3,6,7,0,9,2]

Max = 6

A = [4,3,6,7,0,9,2]

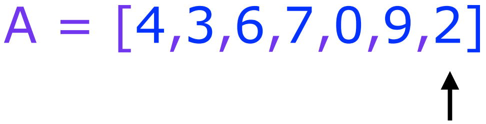
↑

Max = 7

A = [4,3,6,7,0,9,2]

↑

Max = 7



A = [4,3,6,7,0,9,2]

Max = 9

A = [4,3,6,7,0,9,2]

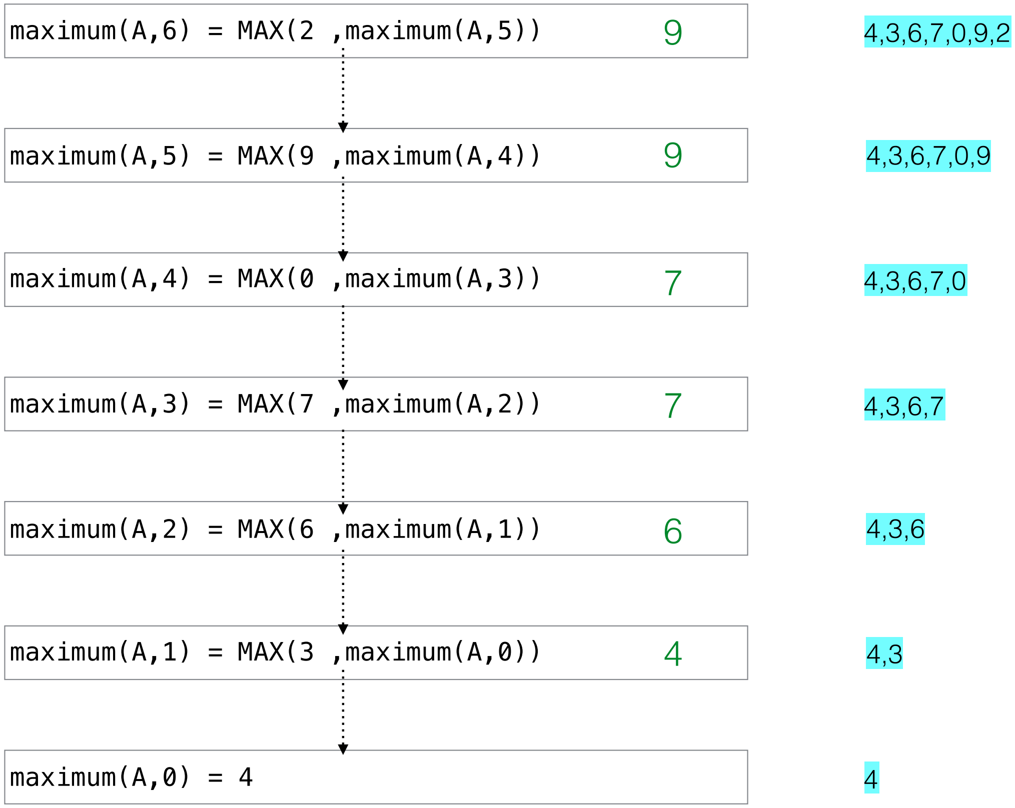
Max = 9

Recurrence relation

maximum(A,i) = MAX(A[i], maximum(A,i-1))

maximum(A,0) = A[0]

Find maximum



maximum(A,6) = MAX(2 ,maximum(A,5)) 9 4,3,6,7,0,9,2

maximum(A,5) = MAX(9 ,maximum(A,4)) 9 4,3,6,7,0,9

maximum(A,4) = MAX(0 ,maximum(A,3)) 7 4,3,6,7,0

maximum(A,3) = MAX(7 ,maximum(A,2)) 7 4,3,6,7

maximum(A,2) = MAX(6 ,maximum(A,1)) 6 4,3,6

maximum(A,1) = MAX(3 ,maximum(A,0)) 4 4,3

maximum(A,0) = 4 4

Java

```java
public static int maximum(int[] nums, int i){
    if(n == 0){
        return nums[0];
    }
    return Math.max(nums[i], maximum(nums,i-1));
}
```

Python

```python
def maximum(A, i):
    if i == 0: return A[0]
    return max(A[i], maximum(A, i - 1))
```

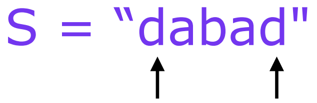
2. Given a string, write a function which uses recursion to check if its palindrome. A palindrome is a string which is same when read from either direction.

Example:

Input=“dabad” , output=true

Input=“xyyx” , output=true

Input=“ppq” , output=false



S = “dabad"

S = “dabad"

↑ ↑

S = “dabad"

↑

Recurrence relation

isPalindrome(S,i,j) = isPalindrome(S,i+1,j-1) if S[i] = S[j] else false

isPalindrome(S,i,j) = true if i >= j

Java

```java
public static boolean isPalindrome(String input, int i, int j)
{
    if (i >= j) {
        return true;
    }
    return input.charAt(i) == input.charAt(j) &&
isPalindrome(input, i + 1, j - 1);
}
```

Python

```python
def is_palindrome(S, i, j):
    if i >= j:
        return True
    return S[i] == S[j] and is_palindrome(S, i + 1, j - 1)
```

Exercise

1. Given an array, write a recursive function to check if the elements of array are in sequence.

Input=[2,3,4,5,6,7] , Output=true

Input=[2,4,5,6,7] , Output=false , because 3 is missing in the sequence

Recurrence relation

isInSequence(A,i) = isInSequence(A,i+1) if A[i] == A[i+1] -1

isInSequence(A,i) = true , if i == A.length-1

Java

```java
public static boolean isInSequence(int[] input,int index){
    return index == input.length-1 || (input[index] ==
input[index+1]-1 && isInSequence(input,index+1));
}
```

Python

```python
def check_sequence(nums, i):
    return i == len(nums)-1 or (nums[i] == nums[i+1]-1 and 
check_sequence(nums,i+1))
```

2. Given an integer, write a recursive function to return the sum of its digits.

Example:

Input = 123456 , output = 21

S=123456

123456 % 10 = 6

S=12345 **6**

Sum = 6

123456 / 10 = 12345

S = 12345

S=1234 **5**

Sum = 6+5 = 11

Recurrence relation

digitsSum(num) = (num%10)+digitsSum(num/10)

digitsSum(num) = 0, if num = 0

Java

```java
public static int digitsSum(int num){
    if(num == 0){
        return 0;
    }
    int digit = num%10;
    return digit+ digitsSum(num/10);
}
```

Python

```python
def digits_sum(n):
    if n == 0:
        return 0
    return n%10 + digits_sum(int(n/10))
```