L10: Recursion and Sorting

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

Euclid's Algorithm & Recursion

Greatest Common Divisor (GCD) in C

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

What is GCD?

- **GCD (Greatest Common Divisor)** of two integers `a` and `b`
  = the largest integer that divides both without remainder.

Examples:

- gcd(12, 8) = 4
- gcd(20, 28) = 4
- gcd(17, 5) = 1 (they are coprime)

Euclid's Algorithm (Concept)

- Based on this property:
  - `gcd(a, b) = gcd(b, a % b)`
  - Continue until remainder = 0
- When `b = 0`, `gcd(a, b) = a`

Euclid's Algorithm (Steps)

Find gcd(48, 18):

1. gcd(48, 18) = gcd(18, 48 % 18) = gcd(18, 12)
2. gcd(18, 12) = gcd(12, 6)
3. gcd(12, 6) = gcd(6, 0)
4. gcd = 6 ✅

Iterative GCD in C

```c
#include <stdio.h>

int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

Recursive GCD in C

```c
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int main() {
    int x = 48, y = 18;
    printf("GCD of %d and %d is %d\n", x, y, gcd(x, y));
    return 0;
}
```

Dry Run (Recursive Example)

gcd(48, 18):

gcd(48, 18) → gcd(18, 12)

gcd(18, 12) → gcd(12, 6)

gcd(12, 6) → gcd(6, 0)

Return 6 ✅

Ternary Operator

Recursive GCD in C

```c
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int gcd_single_line(int a, int b) {
  return b==0 ? a: gcd_single_line(b, a % b) ;
}
```

Factorial using Ternary operator?

```
int factorial_single_line(int n) {
  return n==0 ? 1: n * factorial_single_line(n-1);
}
```

Sorting

arranging elements of an array in ascending or descending order.

Sorting Algorithms

- Common sorting algorithms:
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort, Quick Sort (advanced)

Selection Sort

- Find the **minimum element** in the unsorted part of the array
- Swap it with the **first unsorted element**
- Move boundary forward
- Repeat until sorted

Recursive Idea

- Base case: if starting index = last index → done
- Recursive case:
    i. Find minimum element from `start` to `end`
    ii. Swap with element at `start`
    iii. Recur for subarray `start+1 ... end`

Recursive Selection Sort in C

```c
#include <stdio.h>

// Find index of minimum element
int minIndex(int arr[], int i, int j) {
    int min_index = i;
    for (int k = i + 1; k < j; k++) {
      if (arr[k] < arr[min_index]) min_index = k;
    }
    return min_index;
}

// Recursive Selection Sort
void selectionSortRecursive(int arr[], int n, int index) {
    if (index == n) return;  // base case

    // Find minimum element in subarray
    int k = minIndex(arr, index, n-1);

    // Swap
    if (k != index) {
        int temp = arr[k];
        arr[k] = arr[index];
        arr[index] = temp;
    }

    // Recur for the rest
lectionSortRecursive(arr, n, index+1);
}
```

Example Walkthrough

Array = [64, 25, 12, 22, 11]

Find min (11), swap with 64 → [11, 25, 12, 22, 64]

Recur on [25,12,22,64] → min=12 → [11,12,25,22,64]

Next → [11,12,22,25,64]

Sorted ✅

Insertion Sort in C (Iterative)

Idea of Insertion Sort

- Builds the sorted array **one element at a time**
- At each step:
  - Take the next element
  - Insert it into the correct position in the already sorted part

Example Walkthrough

Array = [5, 3, 4, 1, 2]

1. [**5**, 3, 4, 1, 2] → Sorted: [5]
2. Insert 3 → [3, 5, 4, 1, 2]
3. Insert 4 → [3, 4, 5, 1, 2]
4. Insert 1 → [1, 3, 4, 5, 2]
5. Insert 2 → [1, 2, 3, 4, 5] ✅ Sorted

Algorithm

1. Start from index `1` (first element is trivially sorted)
2. Pick the element (`key`)
3. Shift all larger elements to the right
4. Insert `key` at the correct position
5. Repeat until end of array

C Implementation

```c
#include <stdio.h>

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Shift elements greater than key
        while (j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}

int main() {
    int arr[] = {5, 3, 4, 1, 2};
    int n = sizeof(arr)/sizeof(arr[0]);

    insertionSort(arr, n);

    for (int i=0; i<n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Dry Run Example

Input: [9, 5, 1, 4, 3]

i=1 → key=5 → [5,9,1,4,3]

i=2 → key=1 → [1,5,9,4,3]

i=3 → key=4 → [1,4,5,9,3]

i=4 → key=3 → [1,3,4,5,9] ✅


INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD