Kartheeka.Repalle

192372289

CSE-AI

7/8/202024

INSERTION SORT:

```
#include <stdio.h>

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

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }

        arr[j + 1] = key;
    }
}

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

```
int main() {
    int arr[] = {7, 3, 10, 4, 1, 11};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    printArray(arr, n);

    insertionSort(arr, n);

    printArray(arr, n);

    return 0;
}
```

Output:

Original array: 7 3 10 4 1 11

Sorted array: 1 3 4 7 10 11

MERGE SORT:

```
#include <stdio.h>
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int leftArr[n1], rightArr[n2];
```

```
    for (int i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        rightArr[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;

    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = leftArr[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = rightArr[j];
        j++;
        k++;
    }
}
```

```
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {16, 9, 2, 20, 14, 3, 10, 7};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    printArray(arr, size);

    mergeSort(arr, 0, size - 1);

    printf("Sorted array: ");
    printArray(arr, size);
```

```
    return 0;
}
```

Output:

Original array: 16 9 2 20 14 3 10 7

Sorted array: 2 3 7 9 10 14 16 20

RADIX SORT:

Here is an example of radix sort in C:

```
#include <stdio.h>

void radixSort(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    for (int exp = 1; max / exp > 0; exp *= 10) {
        int output[n];
        int count[10] = {0};

        for (int i = 0; i < n; i++)
            count[(arr[i] / exp) % 10]++;
```

```
        for (int i = 1; i < 10; i++)
            count[i] += count[i - 1];

        for (int i = n - 1; i >= 0; i--) {
            output[count[(arr[i] / exp) % 10] - 1] = arr[i];
            count[(arr[i] / exp) % 10]--;
        }

        for (int i = 0; i < n; i++)
            arr[i] = output[i];
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {81, 901, 100, 12, 150, 77, 55, 23};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    printArray(arr, n);

    radixSort(arr, n);

    printf("Sorted array: ");
    printArray(arr, n);
```

```
    return 0;
}
```

Output:

Original array: 81 901 100 12 150 77 55 23

Sorted array: 12 23 55 77 81 100 150 901