PROGRAM 11

Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

ALGORITHM

```
Heap_sort (arr[], n):
    Build_max_heap (arr, n)
    for i = n-1 to 1:
        swap arr[0] with arr[i]
        max_heapify (arr, 0, i)

Build_max_heap (arr, n)
    for i = n/2 - 1 down to 0:
        max_heapify (arr, i, n)

max_heapify (arr, i, n):
    left = 2*i + 1
    right = 2*i + 2
    largest = i
    if left < n & arr[left] > arr[largest]:
        largest = left
    if right < n & arr[right] > arr[largest]:
        largest = right
    if largest != i:
        swap arr[i] with arr[largest]
        max_heapify (arr, largest, n)
```

CODE

```
#include <stdio.h>
#include <time.h>

// Function to heapify a subtree rooted at index i
void max_heapify(int arr[], int i, int n) {
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    int largest = i;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;

        max_heapify(arr, largest, n);
    }
}

// Function to build a max heap
void build_max_heap(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--)
        max_heapify(arr, i, n);
}
```

```
// Heap Sort function
void heap_sort(int arr[], int n) {
    build_max_heap(arr, n);

    for (int i = n - 1; i >= 1; i--) {
        // Swap arr[0] with arr[i]
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        // Call max_heapify on the reduced heap
        max_heapify(arr, 0, i);
    }
}

// Function to print the array
void print_array(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[100], n;
    clock_t start, end;
    double time_taken;

    printf("Enter number of elements: ");
    scanf("%d", &n);
```

```
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    start = clock();  // Start timing

    heap_sort(arr, n);

    end = clock();    // End timing
    time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

    printf("Sorted array:\n");
    print_array(arr, n);

    printf("Time taken: %f seconds\n", time_taken);

    return 0;
}
```

OUTPUT

```
Enter number of elements: 6
Enter 6 elements:
45 20 35 10 50 25
Sorted array:
10 20 25 35 45 50
Time taken: 0.000003 seconds
```

TRACING

Tracing

Input: [45, 20, 35, 10, 50, 25]

S1: Build max Heap

Initial Array: [45, 20, 35, 10, 50, 25]

After heapify (2): [45, 20, 35, 10, 50, 25]

After heapify (1): [45, 50, 35, 10, 20, 25]

After heapify (0): [50, 45, 35, 10, 20, 25]

S2: Sorting Phase

Swap 50 <-> 25 : [25, 45, 35, 10, 20, 50]

Heapify : [45, 25, 35, 10, 20, 50]

Swap 45 <-> 20 : [20, 25, 35, 10, 45, 50]

Heapify : [35, 25, 20, 10, 45, 50]

Swap 35 <-> 10 : [10, 25, 20, 35, 45, 50]

Heapify : [25, 10, 20, 35, 45, 50]

Swap 25 <-> 20 : [20, 10, 25, 35, 45, 50]

Heapify : [20, 10, 25, 35, 45, 50]

Swap 20 <-> 10 : [10, 20, 25, 35, 45, 50]

Final Sorted Array

[10, 20, 25, 35, 45, 50].