Chapter 35: Heap Sort

Section 35.1: C# Implementation

```
public class HeapSort
{
    public static void Heapify(int[] input, int n, int i)
    {
        int largest = i;
        int l = i + 1;
        int r = i + 2;

        if (l < n && input[l] > input[largest])
            largest = l;

        if (r < n && input[r] > input[largest])
            largest = r;

        if (largest != i)
        {
            var temp = input[i];
            input[i] = input[largest];
            input[largest] = temp;
            Heapify(input, n, largest);
        }
    }

    public static void SortHeap(int[] input, int n)
    {
        for (var i = n - 1; i >= 0; i--)
        {
            Heapify(input, n, i);
        }
        for (int j = n - 1; j >= 0; j--)
        {
            var temp = input[0];
            input[0] = input[j];
            input[j] = temp;
            Heapify(input, j, 0);
        }
    }

    public static int[] Main(int[] input)
    {
        SortHeap(input, input.Length);
        return input;
    }
}
```

Section 35.2: Heap Sort Basic Information

Heap sort is a comparison based sorting technique on binary heap data structure. It is similar to selection sort in which we first find the maximum element and put it at the end of the data structure. Then repeat the same process for the remaining items.

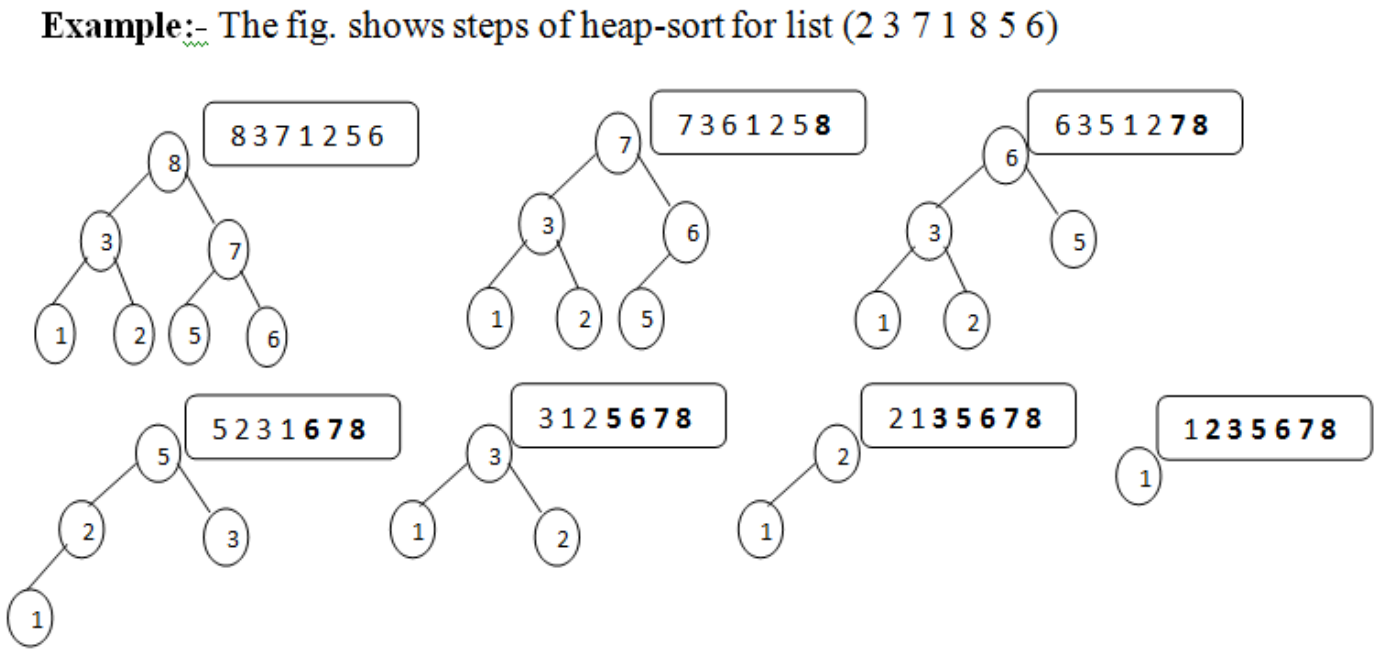
Pseudo code for Heap Sort:

```
function heapsort(input, count)
```

```
    heapify(a,count)
    end <- count - 1
    while end -> 0 do
    swap(a[end],a[0])
    end<-end-1
    restore(a, 0, end)

function heapify(a, count)
    start <- parent(count - 1)
    while start >= 0 do
        restore(a, start, count - 1)
        start <- start - 1
```

Example of Heap Sort:



Auxiliary Space: O(1)
Time Complexity: O(nlogn)