Heap

Parent = (i-1) / 2
Left = 2*i + 1
Right = 2*i + 2

Max Heap

```
void heapifyUp(int Size = N,int i)
{
    while (i > 0 && A[i] > A[parent(i)])
    {
        swap(A[i],A[parent(i)]);
        i = parent(i);
    }
}
```

```
void heapifyDown(int Size = N,int i)
{
    int smallest = i;
    int left = 2*i + 1;
    int right = 2*i + 2;

    if (left < Size && A[left] > A[smallest])
        smallest = left;

    if (right < Size && A[right] > A[smallest])
        smallest = right;

    if (smallest != i)
    {
        swap(A[smallest],A[i]);
        heapifyDown(smallest);
    }
}
```

Min Heap

```
void heapifyUp(int Size = N,int i)
{
    while (i > 0 && A[i] < A[parent(i)])
    {
        swap(A[i],A[parent(i)]);
        i = parent(i);
    }
}
```

```
void heapifyDown(int Size = N,int i)
{
    int smallest = i;
    int left = 2*i + 1;
    int right = 2*i + 2;

    if (left < Size && A[left] < A[smallest])
        smallest = left;

    if (right < Size && A[right] < A[smallest])
        smallest = right;

    if (smallest != i)
    {
        swap(A[smallest],A[i]);
        heapifyDown(smallest);
    }
}
```

Rest of the functions remain same, only heapify logic changes

```
void insert_element(int X)
{
    if (end_idx < N)
    {
        A[end_idx++] = X;
        heapifyUp(end_idx-1);
    }
}
```

```
void build_heap()
{
    for (int i = N/2 - 1; i >= 0; i--)
        heapifyDown(N, i);
}
```

```
void delete_element(int index)
{
    swap(A[index],A[end_idx]);
    end_idx--;
    heapifyDown(index);
}
```

```
void sortHeap()
{
    build_heap();

    for (int i = N-1; i >= 1; i--)
    {
        swap(A[0],A[i]);
        heapifyDown(i,0);
    }
}
```