Data Structures
Sorting Technique – Heap Sort

Team Emertxe

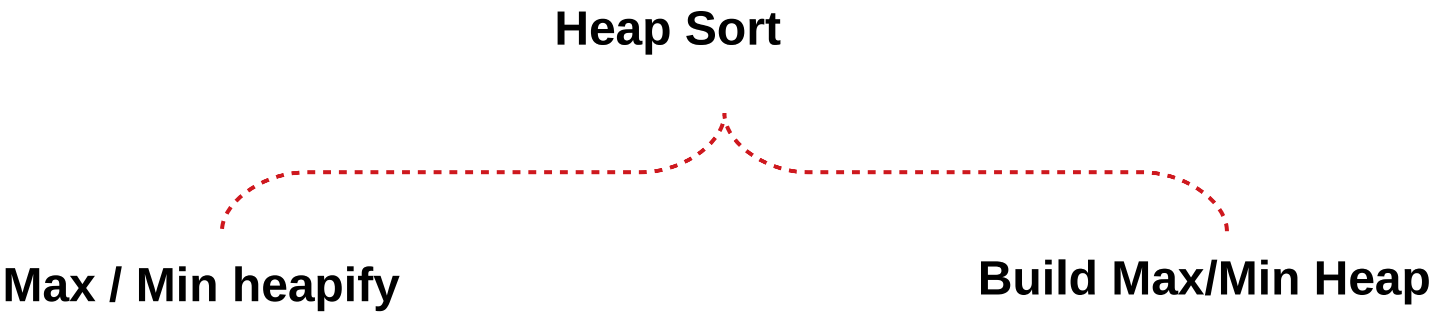
Dependency Algorithm

Introduction

Dependency Algorithm:

Introduction

Dependency Algorithm:



Heap Sort

Max / Min heapify

Build Max/Min Heap

Max heapify

Introduction

Heapify:

- Its a process of converting the given binary tree into heap. It can be either min heap or max heap

- arr[SIZE]

SIZE = 10

Introduction

Heapify:

- Its a process of converting the given binary tree into heap. It can be either min heap or max heap

- arr[SIZE]

SIZE = 10

1	14	10	8	7	9	3	2	4	6
arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]	arr[8]	arr[9]

Introduction

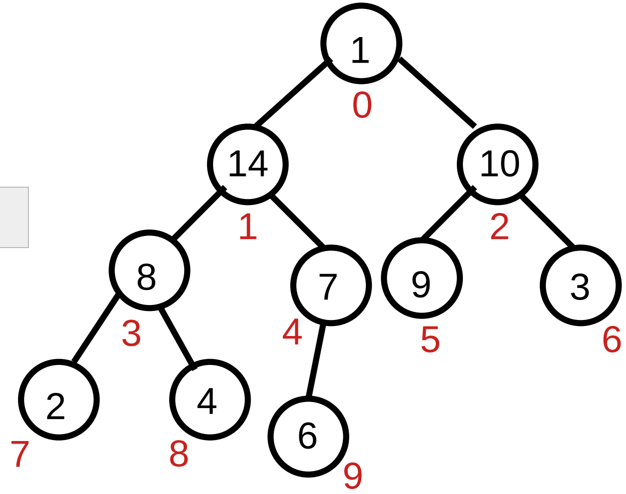
Heapify:

- Its a process of converting the given binary tree into heap. It can be either min heap or max heap

- arr[SIZE]

SIZE = 10

1	14	10	8	7	9	3	2	4	6
arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]	arr[8]	arr[9]



Algorithm

maxheapify(arr,index,size):

Input Specification:

arr	:	Array to hold elements
Index	:	Integer value
size	:	Length of the array

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND  arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0

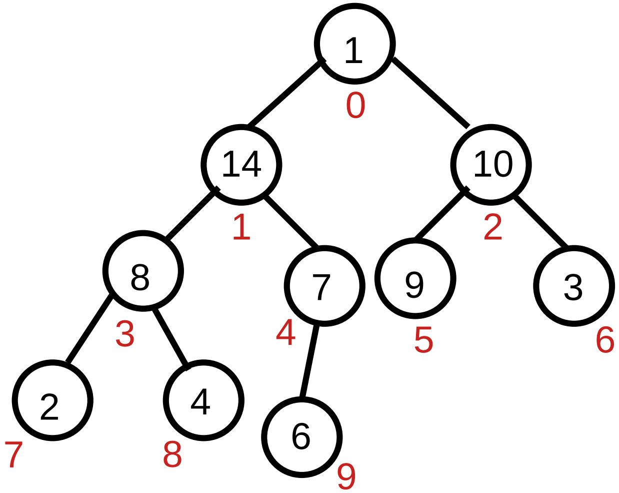
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND  arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0



Data Structure – Sorting Techniques

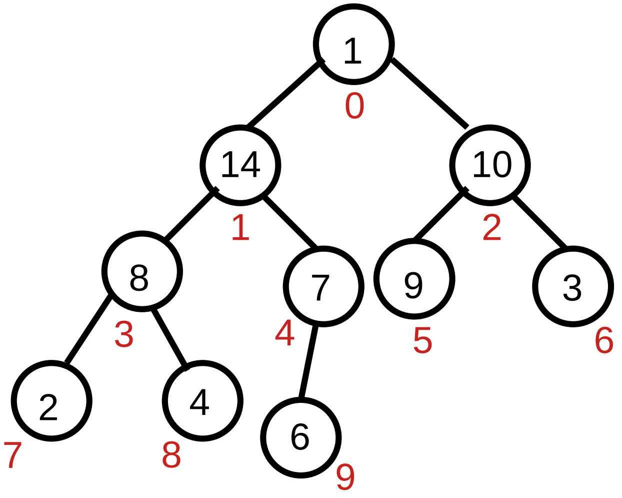
maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0

L = 1



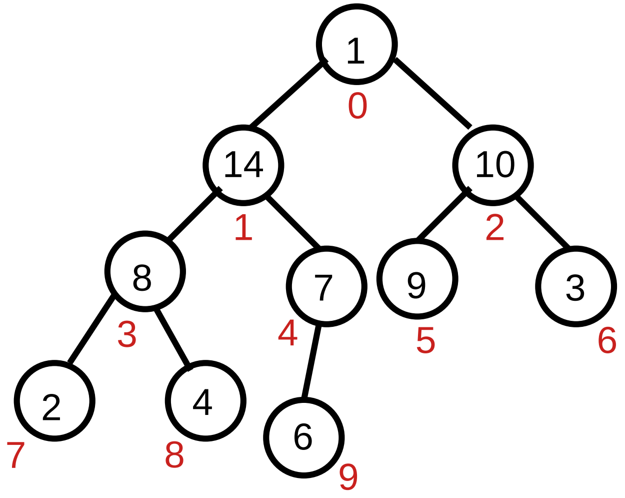
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0 L = 1 R = 2



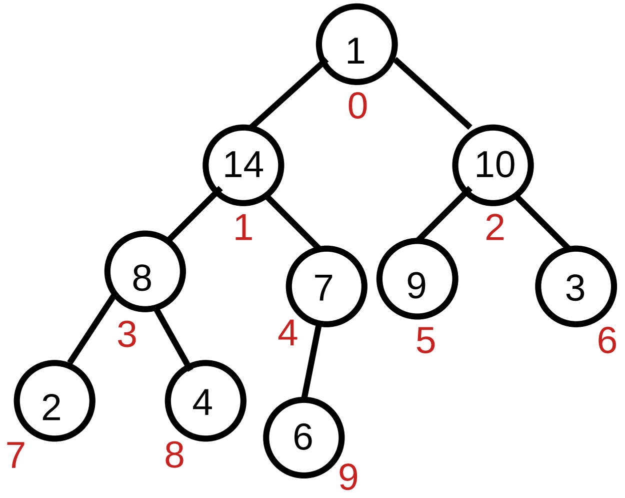
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0 L = 1 R = 2



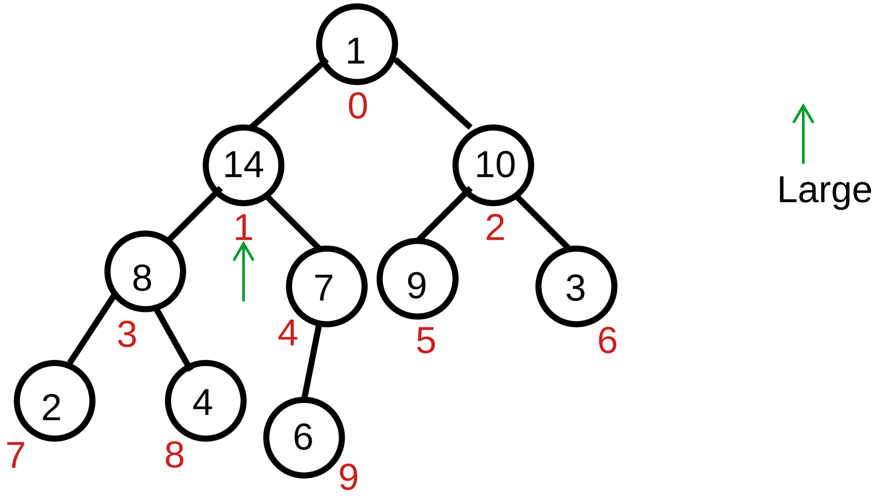
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0   L = 1   R = 2



maxheapify(arr,index,size)

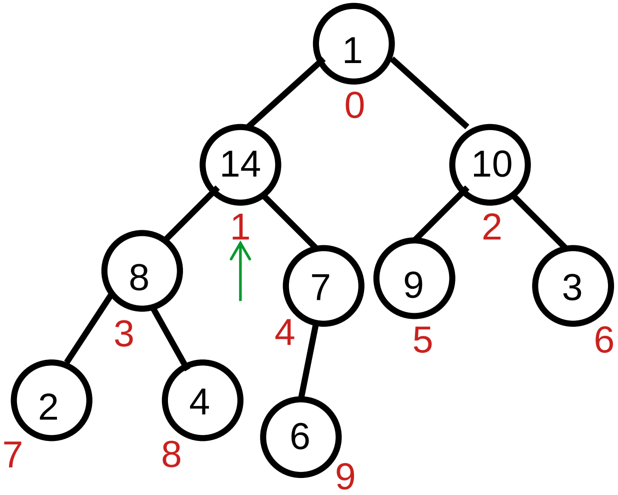
SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size  AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0

L = 1

R = 2



Large

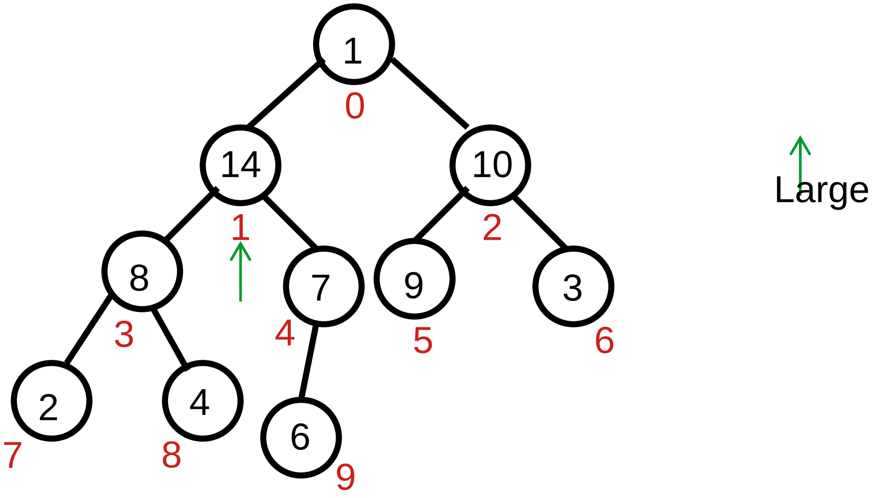
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size  AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0   L = 1   R = 2

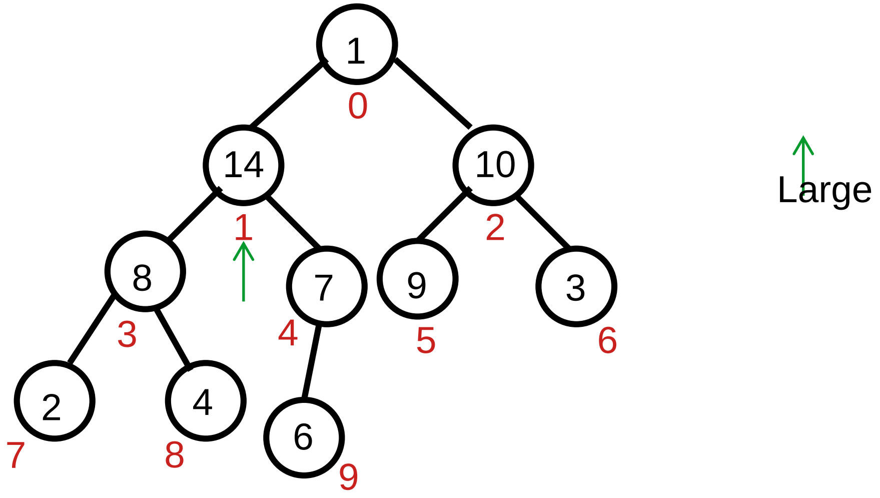


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0 L = 1 R = 2

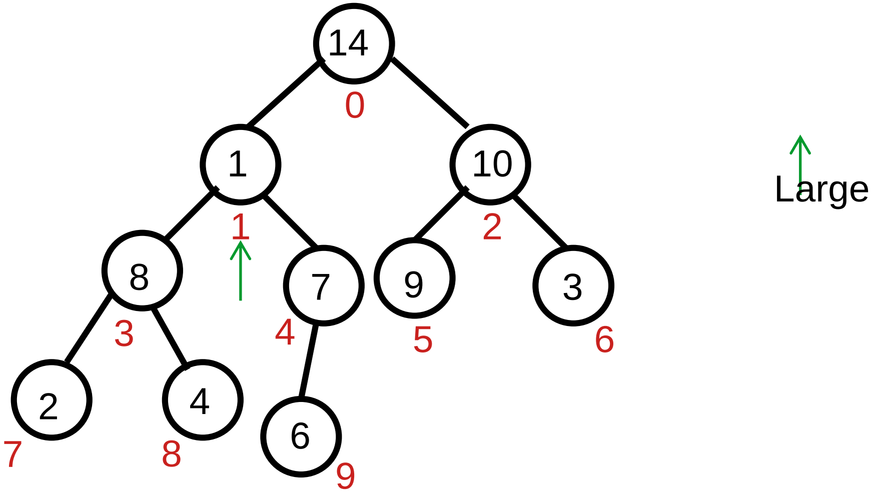


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0   L = 1   R = 2

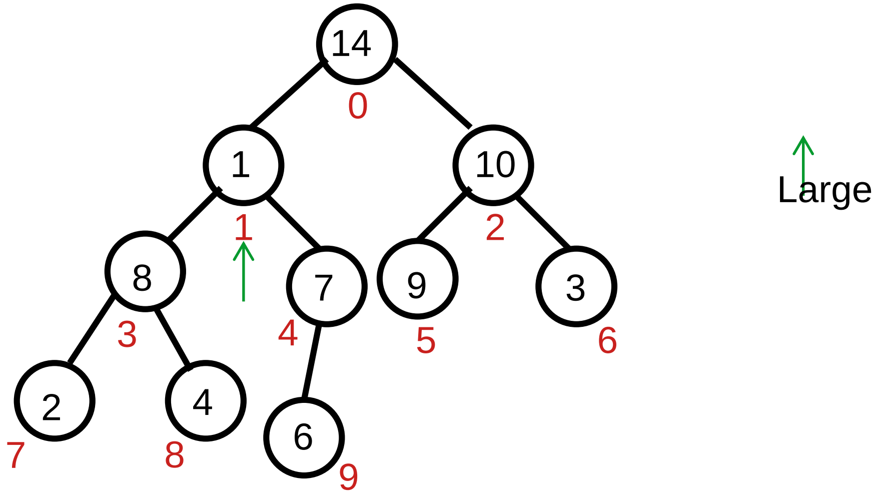


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 0 L = 1 R = 2

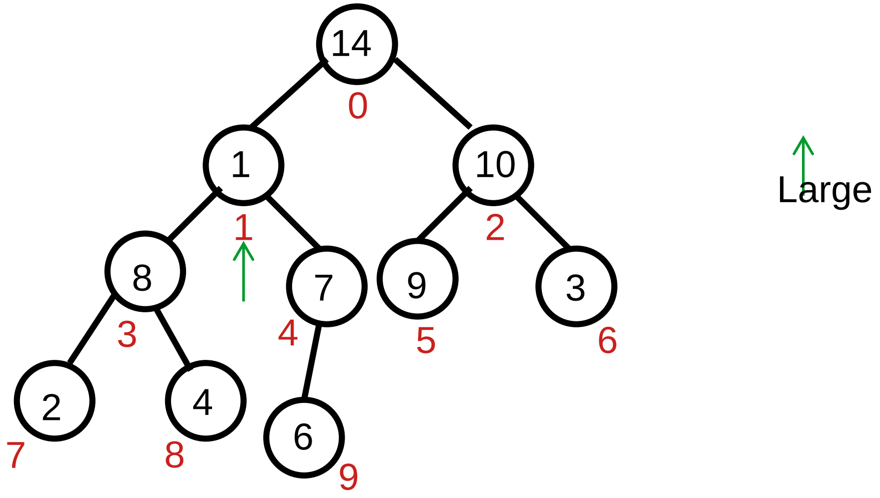


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1



Data Structure – Sorting Techniques

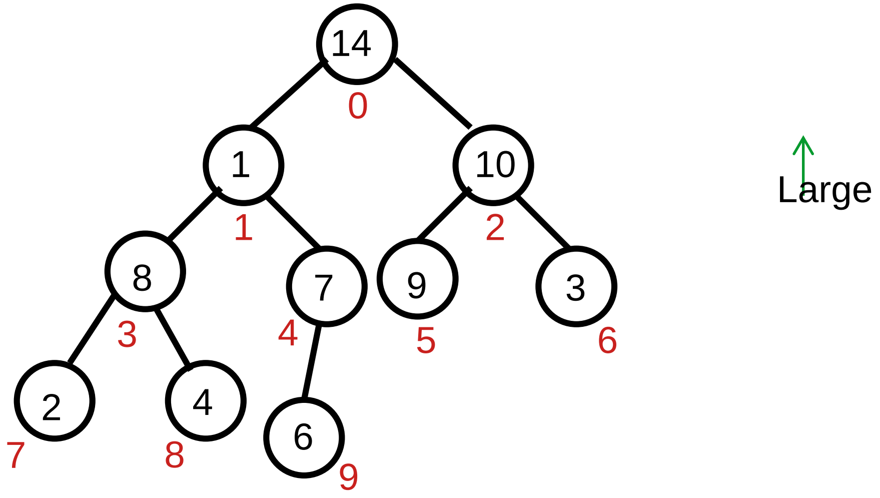
maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1

L = 3

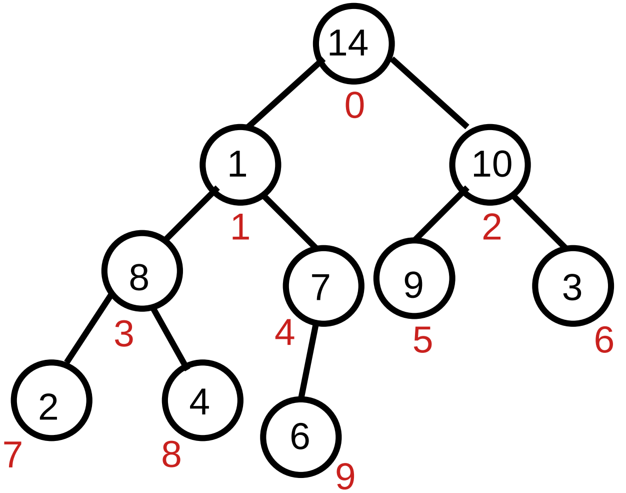


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1 L = 3 R = 4



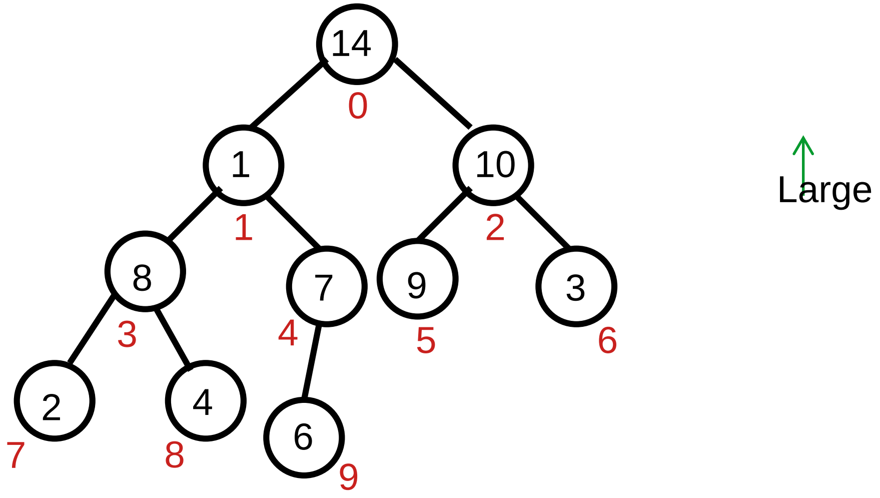
Large

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1 L = 3 R = 4



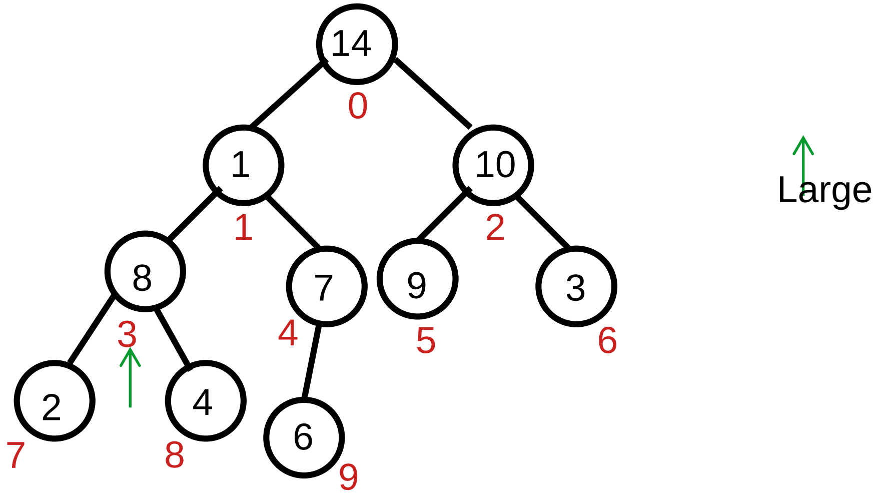
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1   L = 3   R = 4

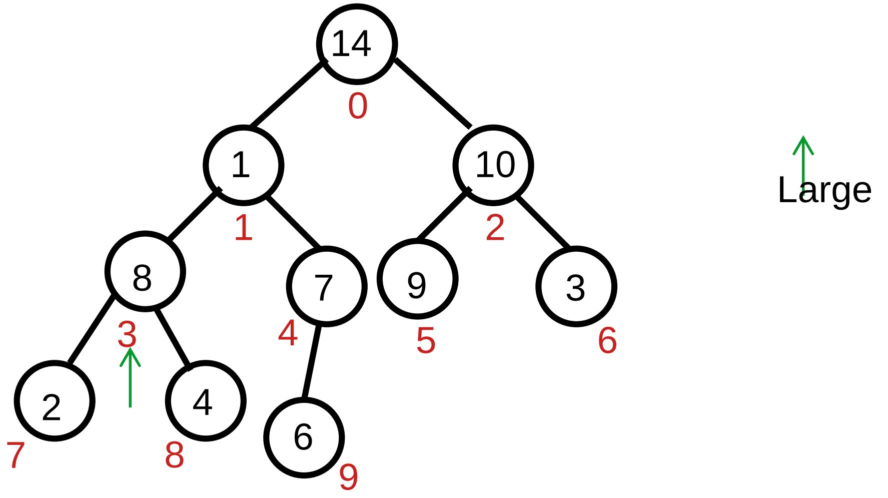


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1 L = 3 R = 4



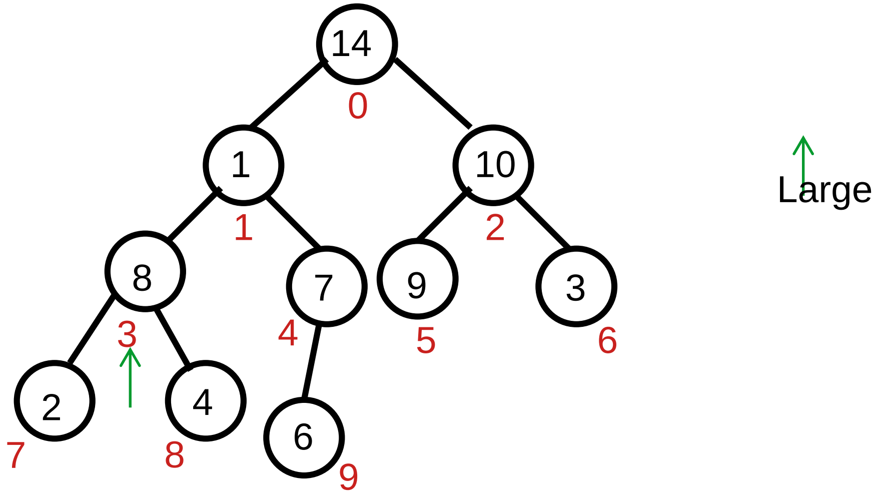
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1 L = 3 R = 4

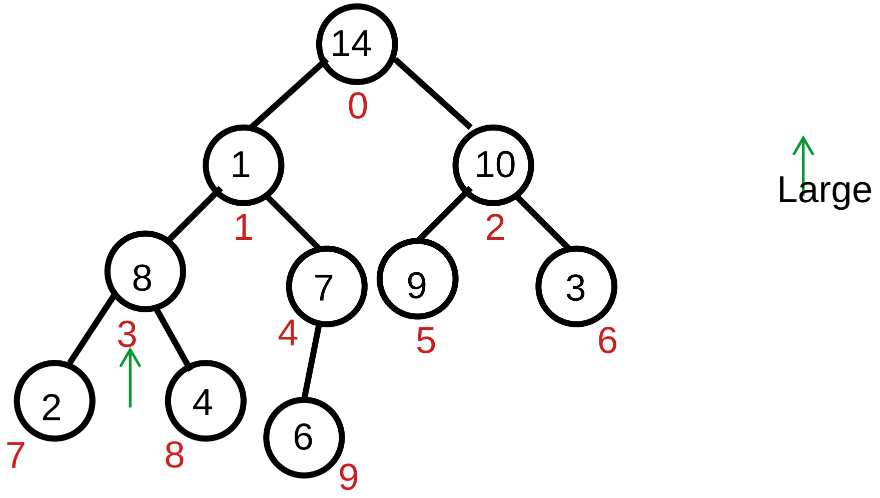


maxheapify(arr,index,size)

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

SIZE = 10

index = 1 L = 3 R = 4

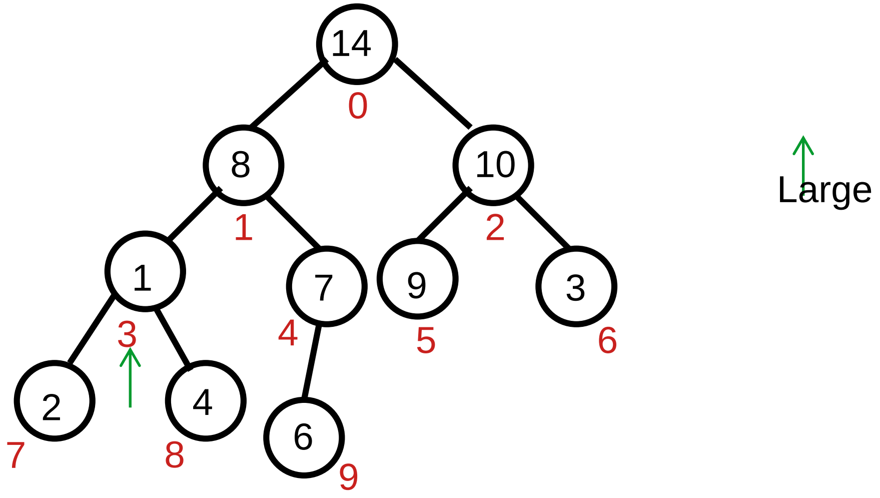


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1 L = 3 R = 4



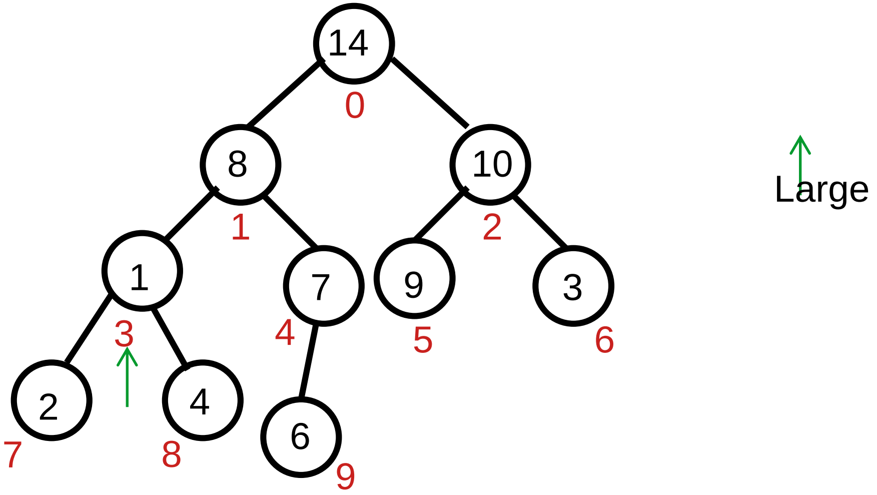
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 1　　L = 3　　R = 4

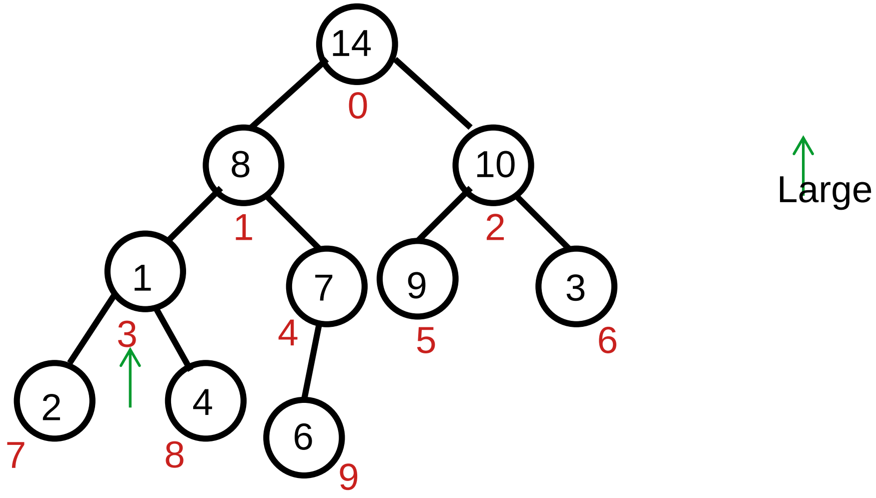


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3



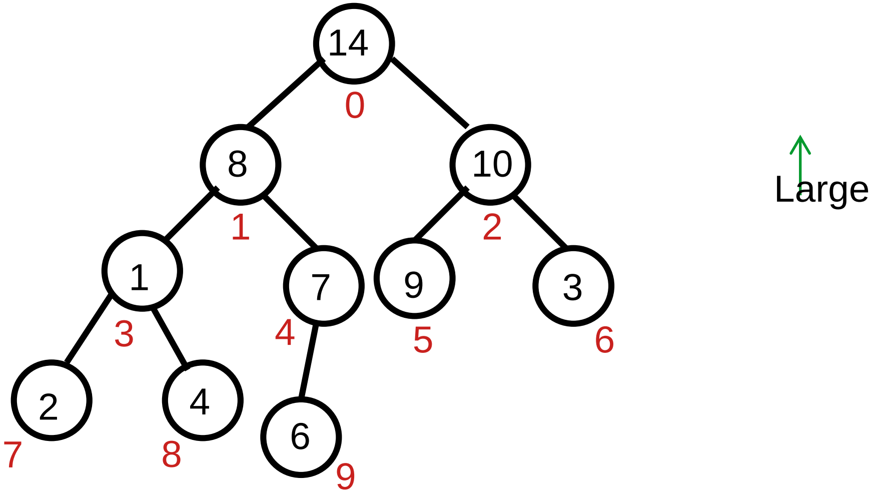
maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3

L = 7

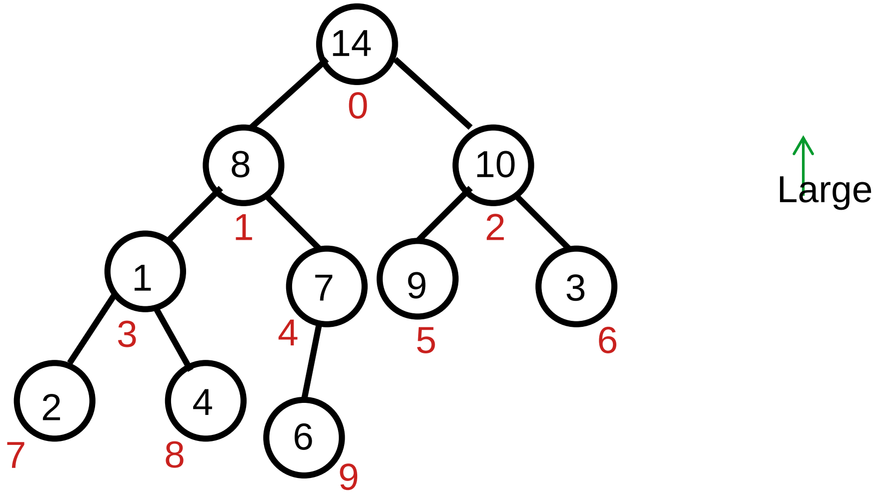


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3   L = 7   R = 8

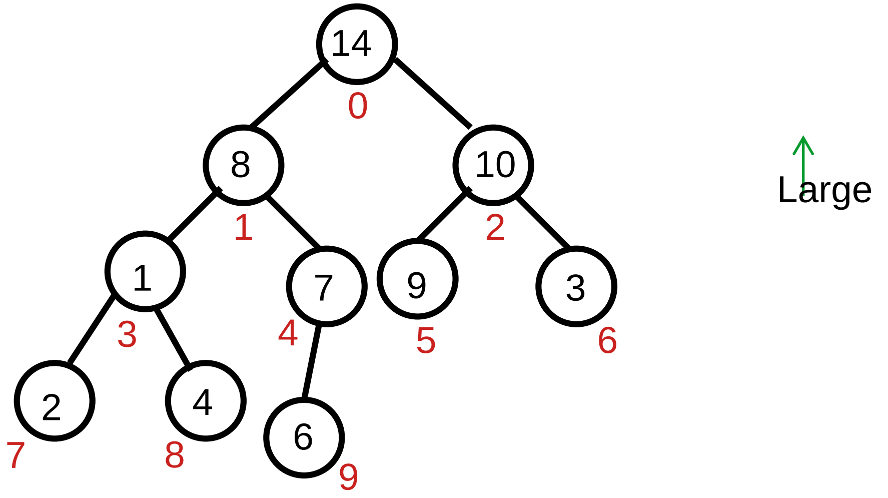


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3 L = 7 R = 8

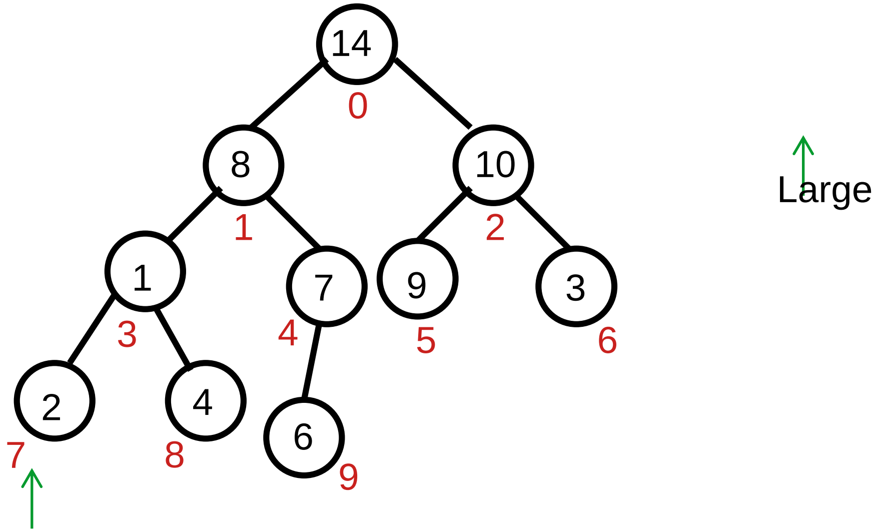


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3   L = 7   R = 8

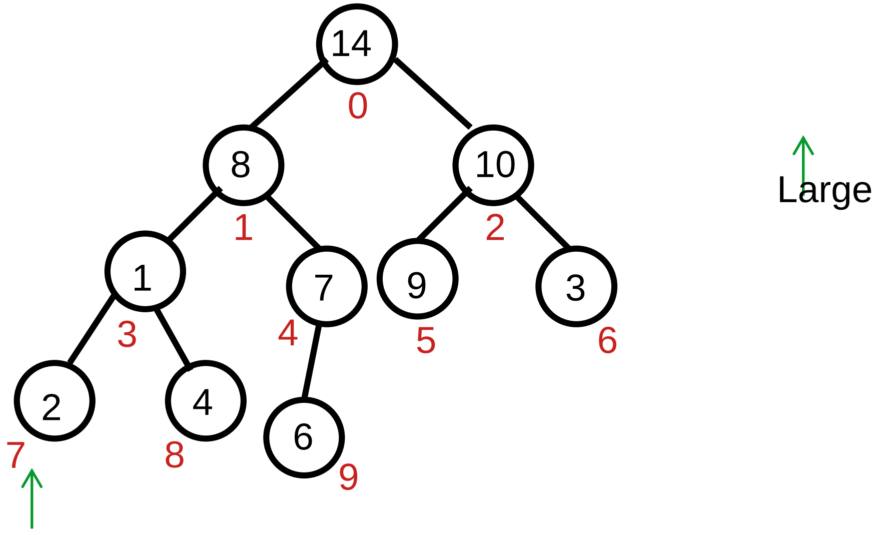


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3   L = 7   R = 8

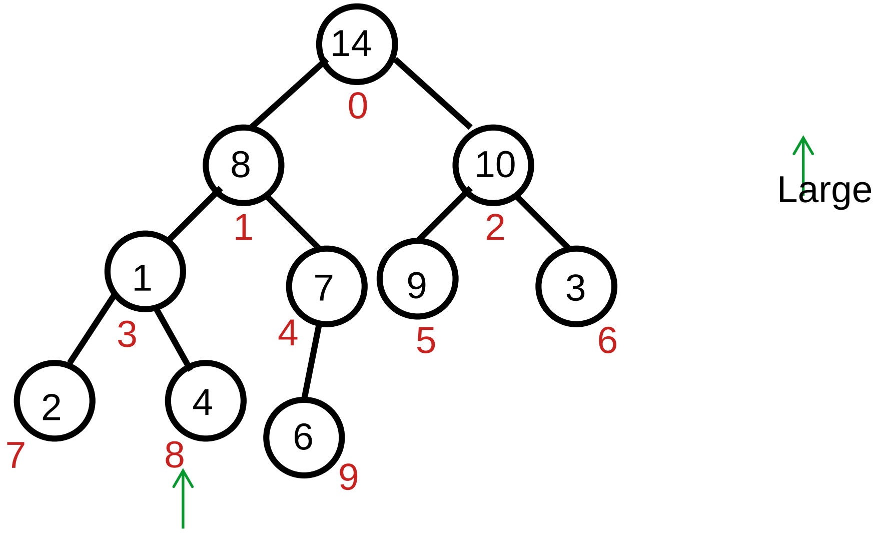


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3 L = 7 R = 8

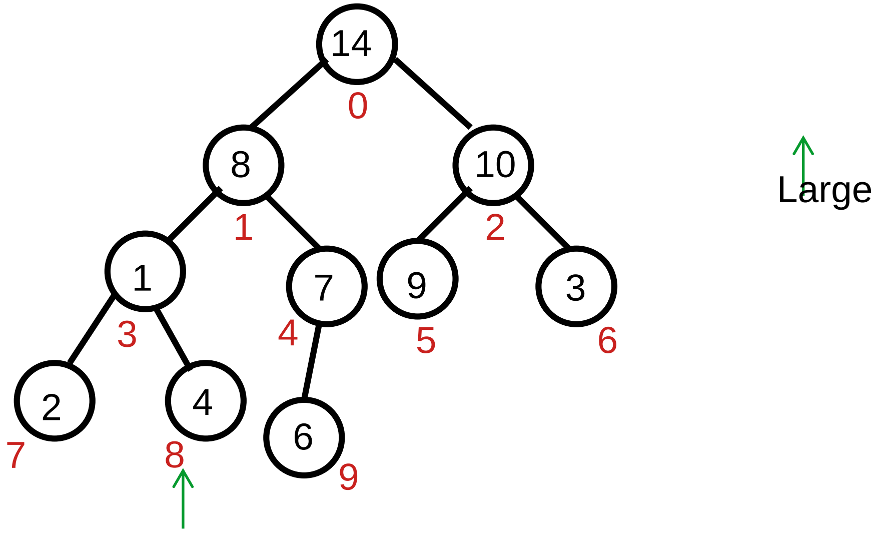


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3   L = 7   R = 8



maxheapify(arr,index,size)

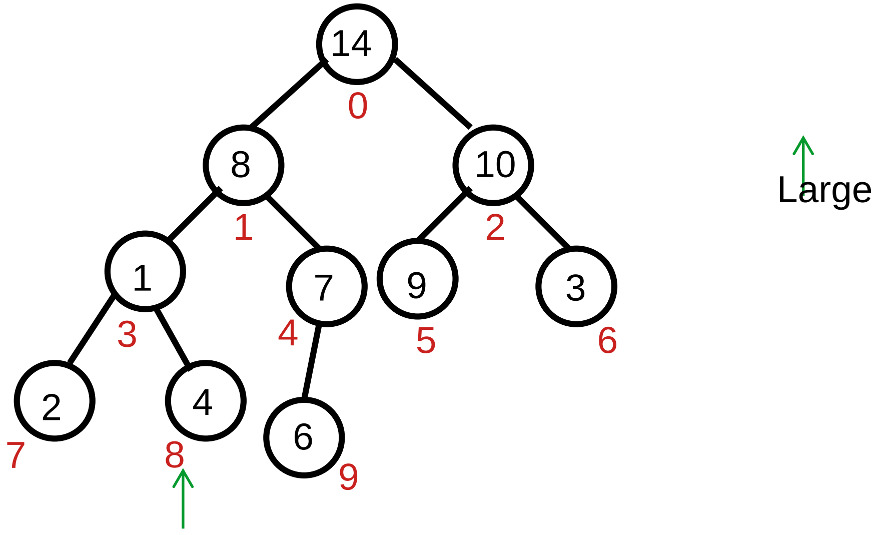
```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

SIZE = 10

index = 3

L = 7

R = 8

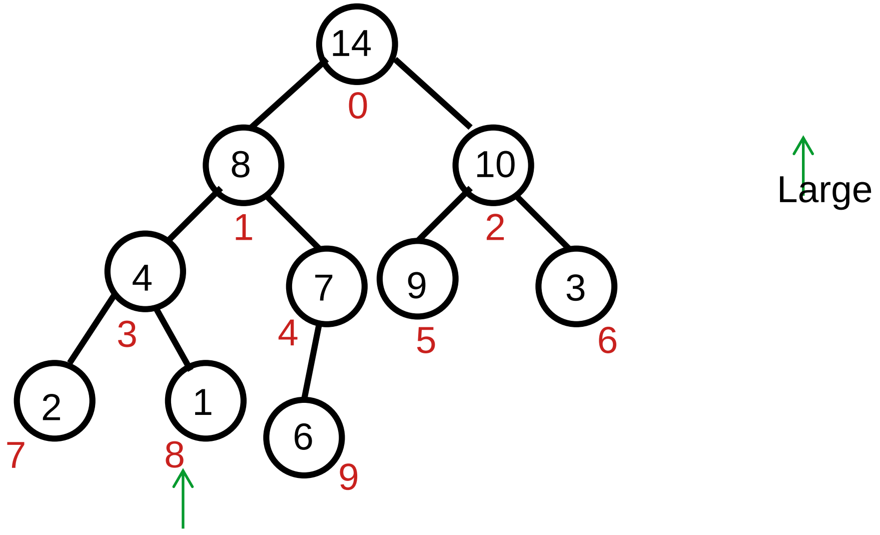


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3 L = 7 R = 8

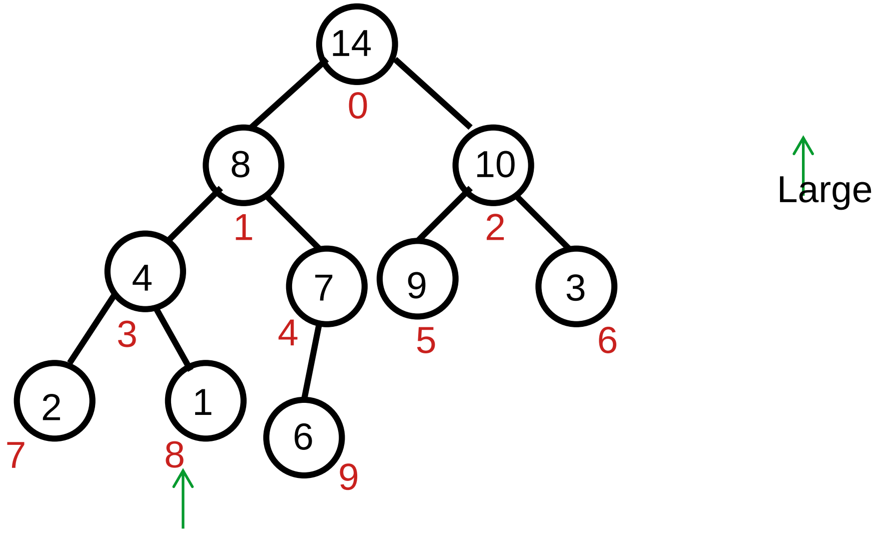


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 3 L = 7 R = 8

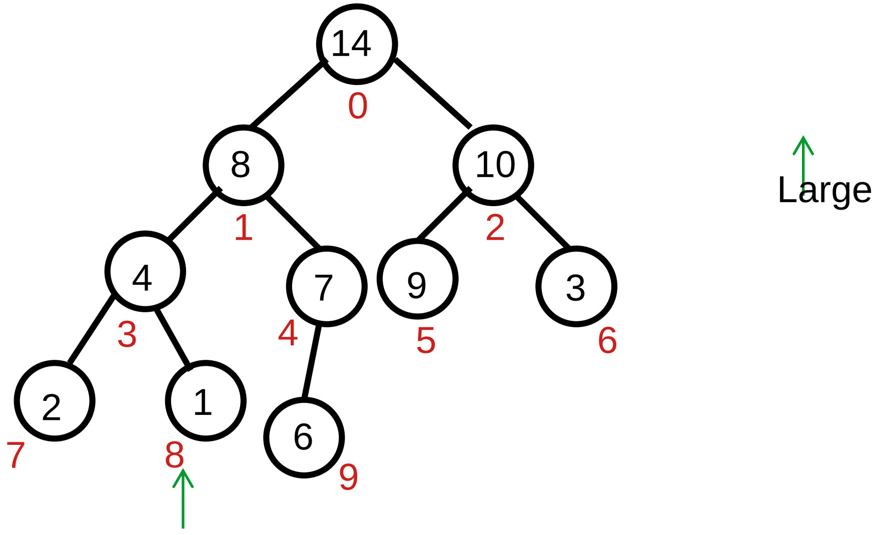


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8



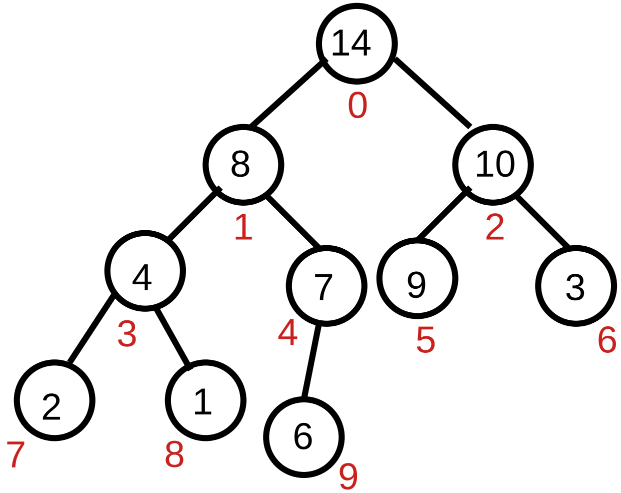
maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8

L = 17



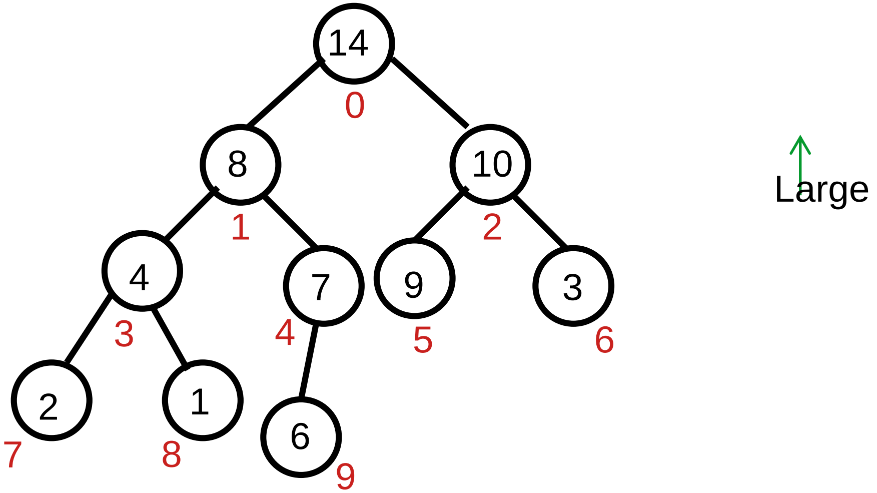
Large

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8 L = 17 R = 18



maxheapify(arr,index,size)

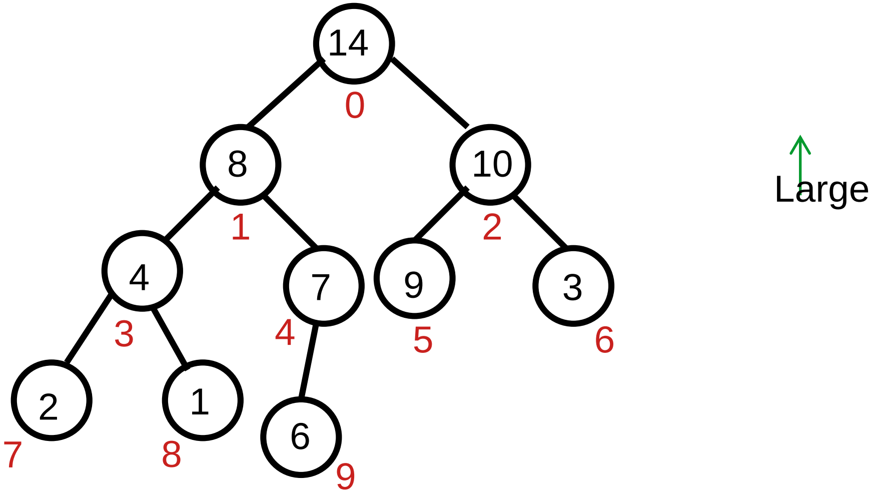
SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8

L = 17

R = 18

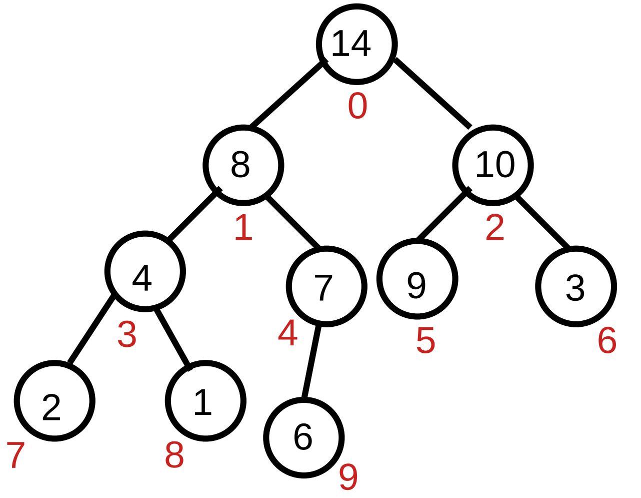


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8 L = 17 R = 18



Large

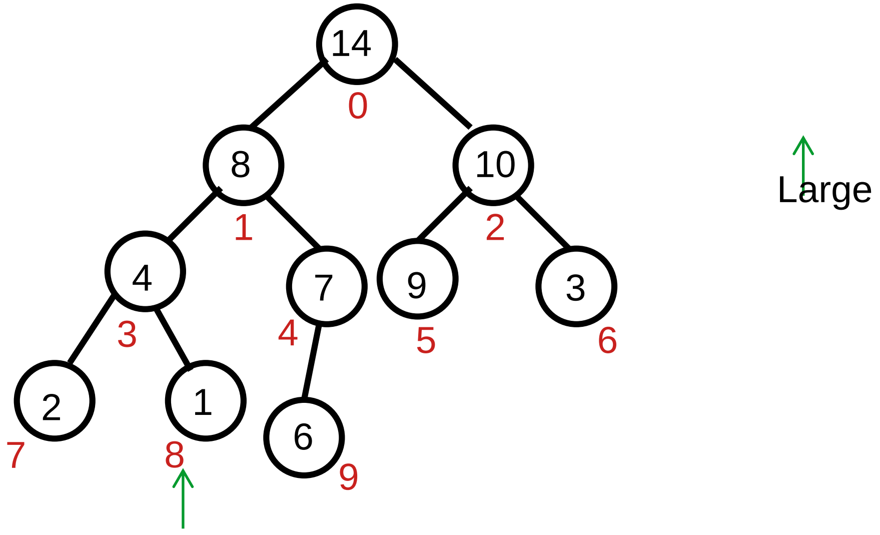
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8 L = 17 R = 18

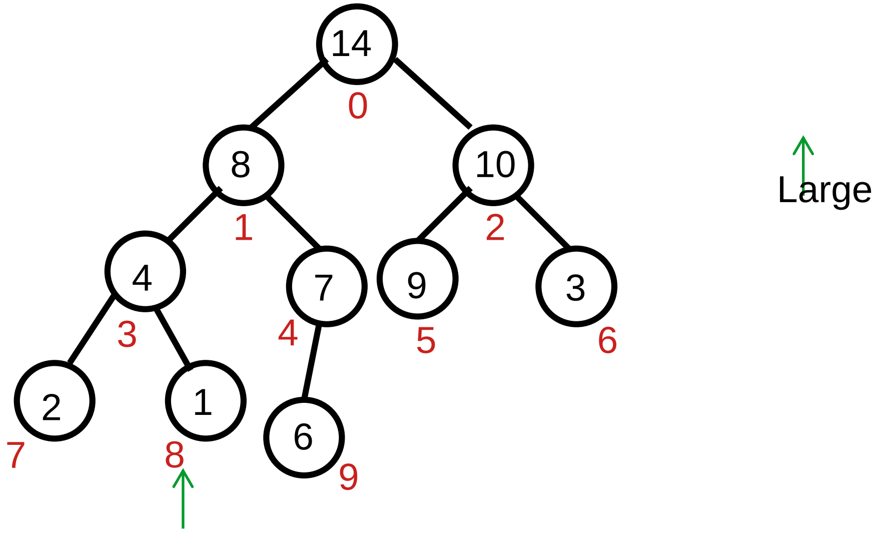


maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

index = 8   L = 17   R = 18



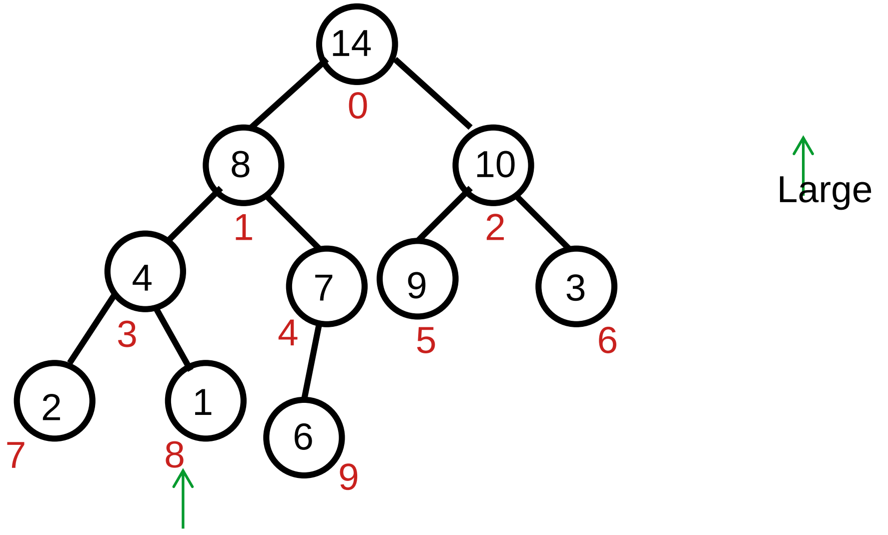
Data Structure – Sorting Techniques

maxheapify(arr,index,size)

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```

SIZE = 10

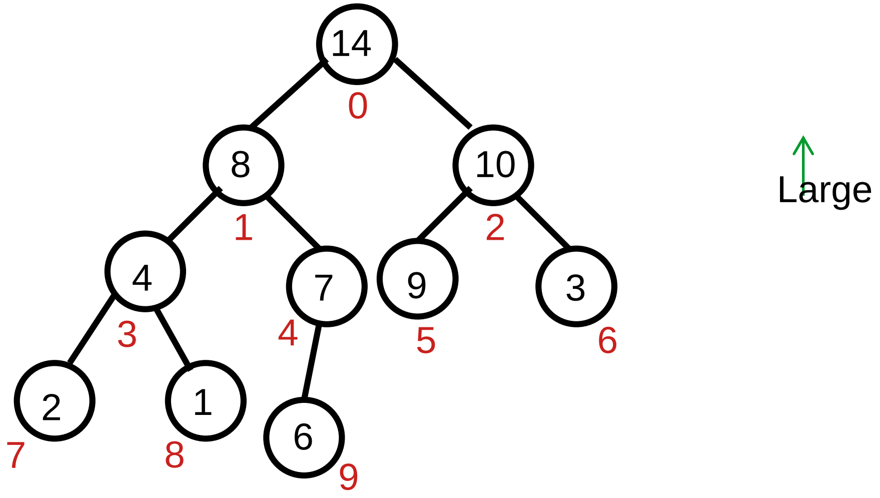
index = 8 L = 17 R = 18



maxheapify(arr,index,size)

SIZE = 10

```
L = 2*index +1
R = 2*index+2
if(L < size AND arr[index] < arr[L])
  large = L
else
  large = index
if(R < size AND arr[large]< arr[R])
  large = R
if(index != large)
  swap(arr[large] ,arr[index])
  maxheapify(arr,large,size)
```



Build Maxheap