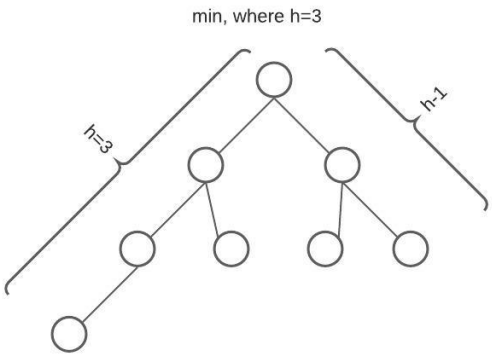
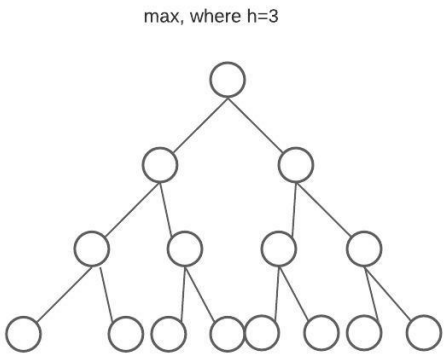
CSCI 4041 Assignment 1

Created	@Sep 24, 2020 9:47 AM
Email	neis@umn.edu
Name	Kelsey Neis
Section	001

6-1

1) mix and max numbers of elements in heap height h





Min: compute n for h-1, then add 1:

2^(h-1+1) -1 = 2^(h) - 1

2^(h) - 1 + 1 (add the node on the last row)

= 2^(h)

Max: follow formula

2^(h+1) - 1

Answer: 2^(h) ≤ n ≤ 2^(h+1) - 1

2) Show that an n-element heap has height lg n.

Given n = 2^(h) is min, **h = [log(n)]**

Given n = 2^(h+1) - 1 is max:

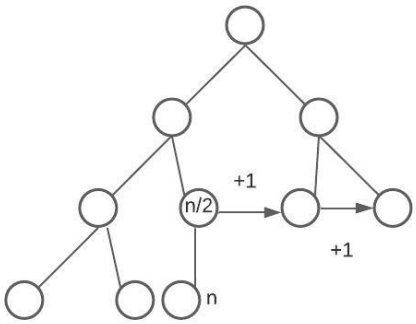
n+1 = 2^(h+1)

(n+1)/2 = 2^(h)

log(n+1) - 1 = h

[log(n+1) - 1] = [log(n)]

7) show that leaves are nodes indexed by [n/2] +1, [n/2] +2,...., n



to get the first leaf, calculate n's parent + 1, since that will be the first node without a child

`parent(n) = [n/2]`

first leaf: [n/2] +1

second leaf: [n/2] + 2

...and so on until we reach the last leaf, n

6-2

3)

The value at A[i] remains in its position, and no further call is made to MaxHeapify, since the heap property is satisfied

4)

Since i > A.heapsize/2, the element A[i] is a leaf, so it will remain in its position, as there are no children to compare it to.

5)

```
MAX-HEAPIFY(A, i)

  l=LEFT(i)
  r=RIGHT(i)
  broken_heap=true
  largest = i

  while broken_heap == true:
    if l > A.heap-size and A[l] > A[i]
      largest = l
    if r <= A.heap-size and A[r] > A[largest]
      largest = r
    if largest == i
      broken_heap = false
    else
      exchange A[i] with A[largest]
      i = largest
```

6-5

8)

```
MAX-HEAP-DELETE(A, i)
  exchange A[i] with A[A.heap_size]
  A.heap_size = A.heap_size - 1
  MAX-HEAPIFY(A, i)
```

Since runtime for MAX-HEAPIFY is 0(lg n), and the first two lines take constant time, the total runtime is 0(lg n)

9)

```
// k-arrays: array of k-arrays
// n: number of total elements

MERGE-SORT( k-arrays, n )
```

```
  merged_array = []
  for i in k-arrays:
    k-heap.push(original-index: i, value: k-arrays[i].pop)

  BUILD-K-HEAP(k-heap) // change any reference to A[i] to A[i][value]
  last-max = k-heap[1]

  while(merged_array.length < n)
    merged_array.push(HEAP-EXTRACT-MAX-MODIFIED(k-heap, last-max.value))
    for j in merged_array.length down to 1 do:
      select-k = merged-array[j][original-index]
      if(k-arrays[select-k].length > 0)
        last-max = k-arrays[select-k].pop
        break


BUILD-K-HEAP(A)   // Time: 0(k)
  for i in [A.length/2] down to 1 do:
    MAX-HEAPIFY(A, i)



HEAP-EXTRACT-MAX-MODIFIED(A, next-elem)
  if A.heap-size < 1
    error “heap underflow”
  max = A[1]
  A[1] = A[next-elem]
  MAX-HEAPIFY(A, 1) // change any reference to A[i] to A[i][value]
  return max
```

Runtime: 0(n*lgk)

BUILD-K-HEAP = 0(k) → constant time

HEAP-EXTRACT-MAX-MODIFIED = 0(n* lgk)