[3.3]
4.3. HEAPSORT

max-heaps can be used for sorting in $O(N \log N)$ time!

idea:
- build max-heap $O(N)$
- deleteMax, repeated N times $O(N \log N)$

$\Big\}\ O(N \log N)$

to avoid use of additional memory, swap max element (after deleting) into last array position and then percolate down (to restore max-heap)

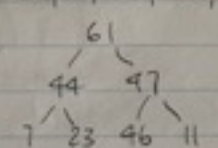
algorithm: given array list of length N

```
1) buildMaxHeap(list)                              // O(N)
2) for (j=N-1; j>0; j--)
     { swap list[0] and list[j];                   // O(1)      } for j=0,...,N-1
       percolateDown(list[0], list[0..j-1]);       // O(log j)  }
     }
```

[24.]
EXAMPLE 34.

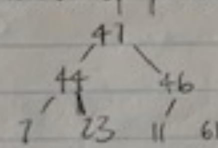
list: 11, 23, 47, 7, 44, 46, 61

after (1):



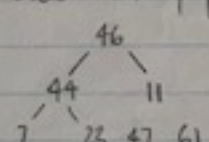
61, 44, 47, 7, 23, 46, 11

after first execution of for loop:



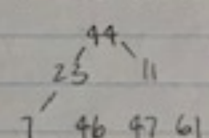
47, 44, 46, 7, 23, 11 | 61

after second execution of for loop:



46, 44, 11, 7, 23 | 47, 61

third:



44, 23, 11, 7 | 46, 47, 61

analysis
worst case: step 1) $O(N)$
step 2) $O(N \log N)$ → $T_{worst}(N) = O(N \log N)$
(one can show $T_{worst}(N) = \Theta(N \log N)$)
average case: one can show that $T_{avg}(N) = \Theta(N \log N)$.
(2.4) (07)

4.4 MERGESORT

recursive sorting: the "divide-and-conquer" idea:
- break a problem instance of size N into smaller problem instances
- solve the smaller instances (typically recursively)
- construct a solution to the larger problem instances from the solutions to the smaller ones

Mergesort for array of length N:
- break into two arrays of size $\frac{N}{2}$: array1, array2
- Mergesort(array1) (rec.) and Mergesort(array2) (rec.)
- merge the two resulting arrays

```
Mergesort(list): (given List of length N)
  if          N=1 do nothing, else:
  {   middle = (N-1)/2;
    array1=Mergesort(list[0..middle]);
    array2=Mergesort(list[middle+1..N-1]);
    return merge(array1, array2);
  }
```

how to merge? compare the first non-copied elem. of array1 to the first nc. elem of array2
array1: 2, 12, 30, 31, 33   array2: 1, 15, 35, 39, 40
copied: 1, 2, 12, 15, 30, 31, 33, 35, 39, 40

7 comparisons:

array1	↔ array2
2	1
2	15
12	15
30	15
30	35
31	35
33	35

when one of the two arrays has been copied, no more comparisons are needed.

CS 340

// Heapsort

Use max heaps to sort in $O(N \log N)$ time

1. Build max heap $O(N)$
2. deleteMax, repeat N times $O(N \log N)$

Swap max element into last array position

Percolate down to restore max heap

Example

61	44	47	7	23	46	11
1	2	3	4	5	6	7

11 44 47 7 23 46 | 61

47 44 11 7 23 46 | 61

47 44 46 7 23 11 | 61

11 44 46 7 23 | 47 61

46 44 11 7 23 | 47 61

23 44 11 7 | 46 47 61

44 23 11 7 | 46 47 61

7 23 11 | 44 46 47 61

23 7 11 | 44 46 47 61

11 7 | 23 44 46 47 61

7 11 23 44 46 47 61

$\Theta(N \log N)$ for both worst case and average case

// Merge Sort

Divide and Conquer approach

- break a problem into smaller problem instances
- solve smaller instances
- Construct solution to large problem by combining the solutions to the smaller ones

How to merge smaller arrays?

- show example

~~2~~ ~~12~~ ~~30~~ ~~31~~ ~~33~~

~~1~~ ~~15~~ 35 39 40

1 2 12 15 30 31 33 35 39 40

~~2 31 30 12 33 1 39 35 40 15~~

2 31 12 30 1 33 ~~35 39 15 40~~

~~2 31 30 12 33 1 39 35 40 15~~

~~2 12~~

2 31 30 12 33 | 1 39 35 40 15

2 31 30 | 12 33 | 1 39 35 | 40 15

2 | 31 30 | 12 33 | 1 | 39 35 | 40 15 |

2 30 31 | 12 33 | 1 | 35 39 | 15 40 |

2 12 30 31 33 | 1 35 39 | 15 40 |

2 12 30 31 33 | 1 15 35 39 40

1 2 12 15 30 31 33 35 39 40