1.

push(x) = O(log n)

top() = O(1)

pop() = O(1)

buildHeap(vector<int>{1...N}) = O(n)

2.

Any application which could normally be done with queues; however, sometimes they need a new element to jump the line. For example Oss. Programs have to wait their turn for CPU time or other limited resources. If you have a program (take a game for example) that needs the resource no matter when it is called, using a priority queue allows the game to jump the line and get the needed resources.

3.

Parent: located at i/2 index.

Children:

Right Child: located at 2*i+1 index

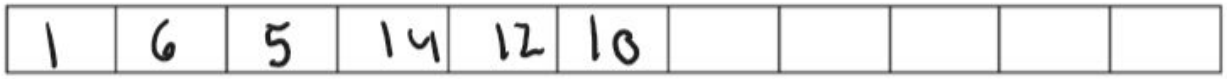
Left Child: located at 2*i index

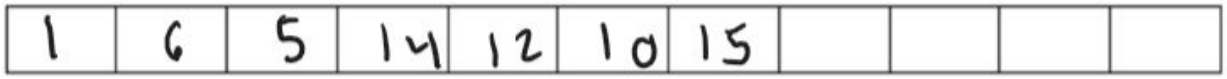
4.

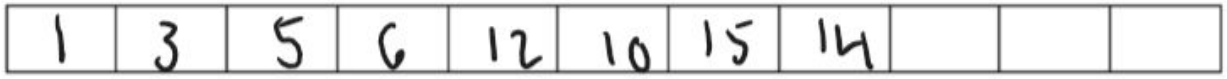
After insert(10):

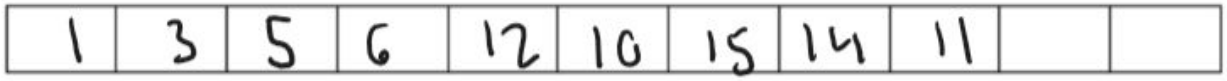
After insert (12):

Etc.









5.

1,3,5,11,6,10,15,14,12

6.

3,6,5,11,12,10,15,14

5,6,10,11,12,14,15

6,11,10,15,12,14

7.

Bubble: O(n^2) Yes

Insert: O(n^2) Yes

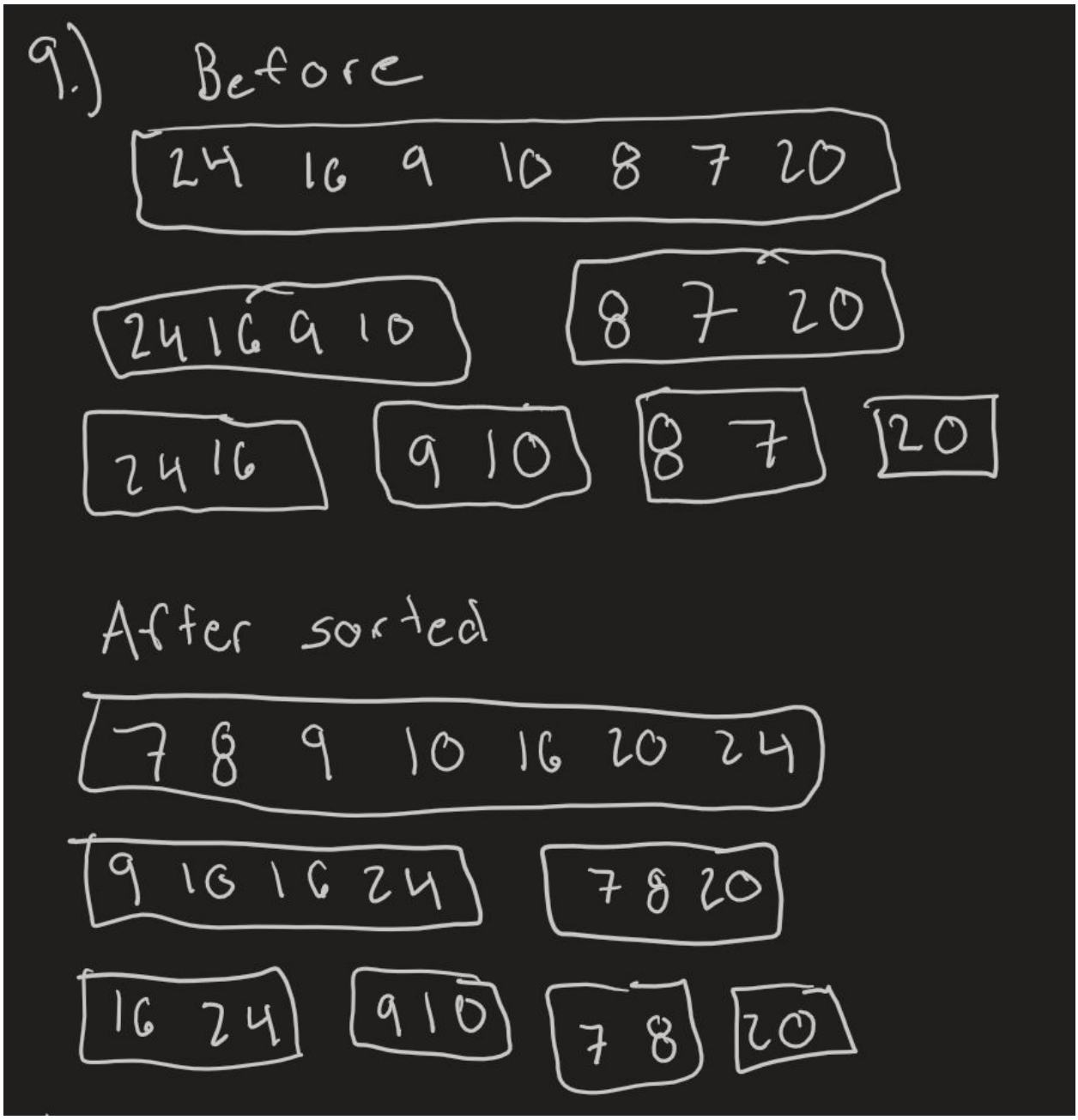
Merge: O(n log(n)) Yes

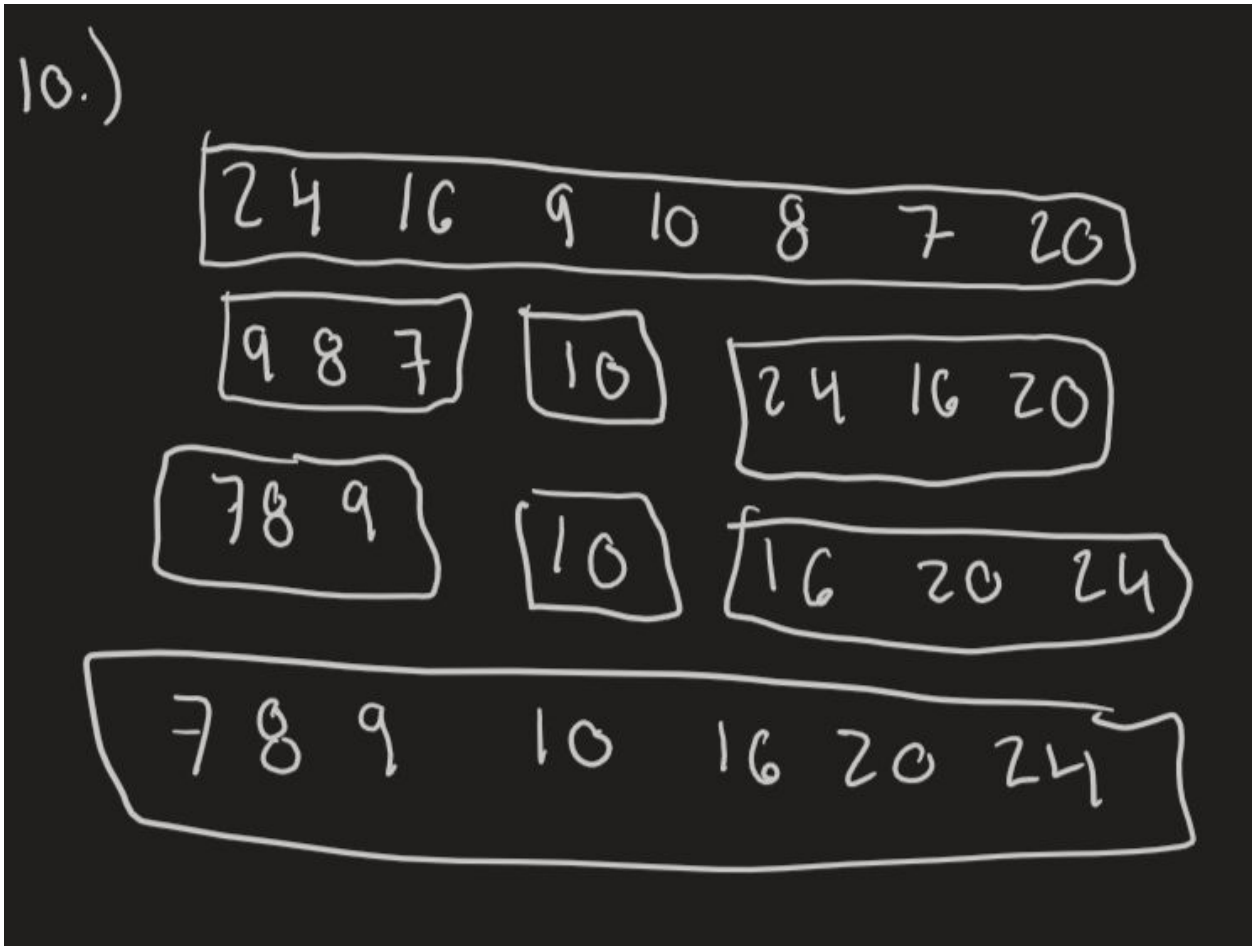
Radix: O(nk) Yes

Quick: O(n log(n)) No

8.

Merge sort splits in half always, merge sort can split into any ratio. Quick sort requires more memory and is very consistent with any amount of data O(n * log(n)). Quick sort requires less memory, but does not scale well with larger data sets $O(n^2)$.


9.) Before
24 16 9 10 8 7 20
24 16 9 10
8 7 20
24 16
9 10
8 7
20
After sorted
7 8 9 10 16 20 24
9 10 16 24
7 8 20
16 24
9 10
7 8
20


10.)
24 16 9 10 8 7 20
9 8 7
10
24 16 20
7 8 9
10
16 20 24
7 8 9 10 16 20 24