Chase Conaway

Github link:

HW3

1.

Chainging

	3		0	12 ➡ 1 ➡ 98		9 ➡ 42	70			

Linear Probing

	3		0	12	1	98	9	42	70	

Quadratic Probing

	42		0	12		3	9	70	1	98

2. The best size to start with in a hash table would be a size of 101 because it offers ample storage space to start and the size is a prime number, which helps the hash function, and collisions happen less.
3.
   a. The load factor is $\frac{53{,}491}{106{,}963}$ which is 0.500088..., or about 0.5.
   b. Yes, we should rehash. With linear probing, approaching a load factor around 0.5 to 0.6, our performance will start to drop as the likelihood of several elements being right next to each other increases. Since linear probing usually only increases the index by 1, this will give us worst case behavior O(n) .

c. No, for separate chaining, rehashing is usually done with a load factor at 0.8-ish because the lists usually are not too long.

4.

Function	Big O complexity
Insert(x)	O(n)
Rehash()	O(n)
Remove(x)	O(1)
Contains(x)	O(1)

5. Your hash table rehash function sucks for a few reasons. The first reason is that doubling the array takes up an unnecessary amount of space, and the table size will never be prime (which means there are probably more collisions). Second, you have two arrays that get massive each time you resize taking up a lot of memory. The third is you loop through an empty array that should initialize the .info field to false, to make sure it is set to false.

6.

Function	Big O
Push(x)	O(log(n))
Top()	O(1)
Pop()	O(log(n))
PriorityQueue(Collection<? extends E> c) // BuildHeap	O(n)

7. I would use a Priority Queue for a hospital. A PQ is a good option for a hospital because a regular list would not be able to attend to the direst situations first. With a PQ,

someone who is seriously wounded would take priority over someone with a small scratch. So, the PQ would sort how serious the injuries are so the doctors can see to the most serious problems first.

8.
   a. A node's parent in a heap is at $\lfloor \frac{i+1}{2} \rfloor$
   b. The node's left child is at 2i + 1 and the right child at 2i + 2

9. Max Heap

10										

12	10									

12	10	1								

14	12	1	10							

14	12	1	10	6						

14	12	5	10	6	1					

15	12	14	10	6	1	5				

15	12	14	10	6	1	5	3			

15	12	14	11	6	1	5	3	10		

10.

15	12	14	11	6	1	5	3	10		

11.

14	12	10	11	6	1	5	3			

12	11	10	3	6	1	5				

11	6	10	3	5	1					

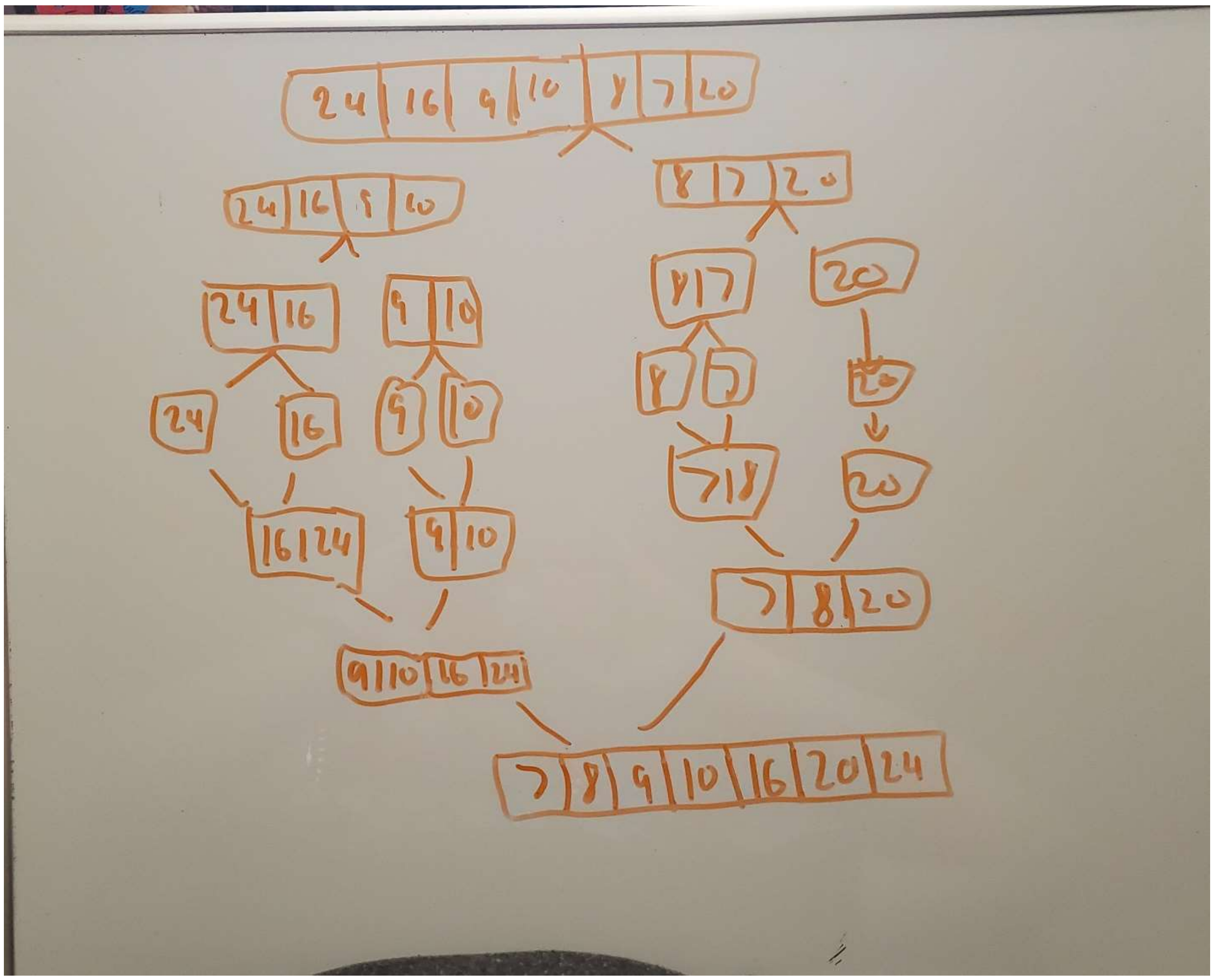
12.

Algorithm	Avg Complexity	Stable (yes/no)
Bubble sort	O(n^2)	Yes
Insertion sort	O(n^2)	Yes
Heap sort	O(nlogn)	No
Merge sort	O(nlogn)	Yes
Radix sort	O(k*n)	Yes
Quicksort	O(nlogn)	no

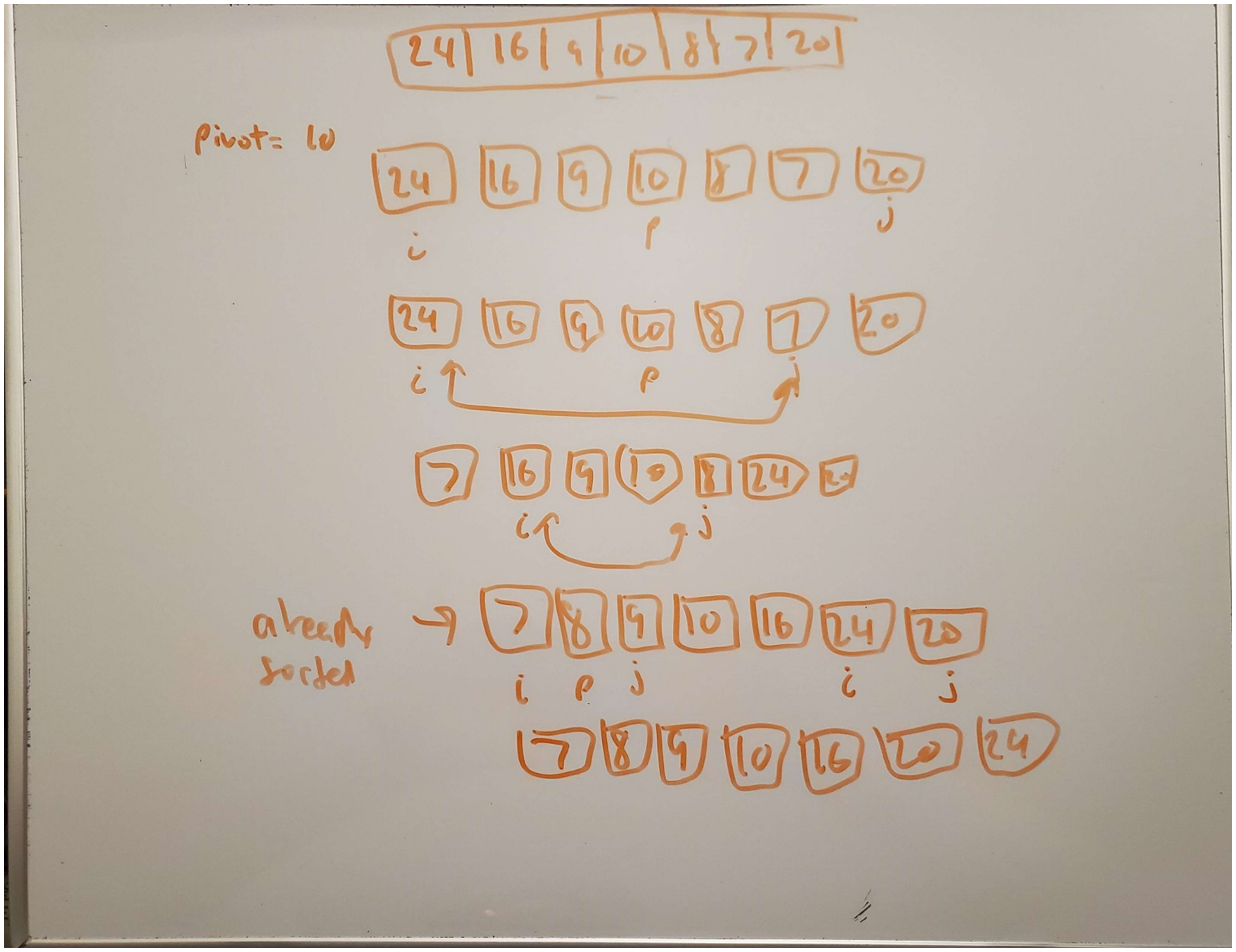
13. In QuickSort, you pick a pivot position to divide the array into two and then put every element great than it above it and every element less than it below it. Recursively do this until you have array size 0 or one and put it back together, MergeSort divides the array into 2 sub arrays and then repeats that until you have all arrays of size one. And then it merges the arrays back together until you have one array again. Quicksort is an in place sorting algorithm, which means it does not need any extra memory

and quick sort is not stable. Since quicksort requires no extra memory and is slightly faster than mergesort, languages often chose quicksort over mergesort.

14.



15.



My pivot picking just took the idle of the algorithm.