Assignment 4
Jason Chen
CMSC 256

Questions

1.

2^{10}	$2^{\log n}$	nlogn	3n +100logn	4n	4nlogn+2n	n^2+10n	n^3	2^n

2.

```
public int span(int n){
if(n==0)                                  c1      1
return 0;                                 c2      1
else if (n==1)                            c3      1
return 1;                                 c4      1
else
return 2+span(n-1)+span(n-2);             c5      1
}
```

When n = 0
T(0) = c1 + c2
When n = 1
T(1) = c3 + c4
When n > 1
T(n) = c5 + T(n-1) + T(n-2)
= T(n-1) + T(n-2) + c
=2T(n-1) + c
T(n) = 2*T(n-1) + c
= 2 * (2*T(n-2)+c) + c
= 2 * (2* (2*T(n-3)+c) + c) + c
= 2^3 * T(n-3) + (22+21+20)*c (assuming n>2)
when substitution repeated i-1th times
= 2i * T(n-i) + (2i-1+ ... +21+20)*c
when i=n
= 2n * T(0) + (2n-1+ ... +21+20)*c
= 2n * c1 + ( )*c
= 2n * c1 + (2n-1)*c
= 2n*(c1+c) – c
So, the growth rate function is O(2^n)
O(2^n)

3.

Array – search is O(n), storage is very good, insertion is O(1), deletion is O(n). Use array when want good storage and fast insertion.
AVL tree – search is O(logn), storage is good, insertion is O(logn), deletion is O(logn). Use AVL tree when want fast search, fast insertion and deletion, and average storage.
Hash table – search is O(1), storage is not good, insertion is O(1), deletion is O(1). Use hash table when want quick search and quick access.

4.
Quicksort – pros: is O(nlogn), best for unsorted lists or arrays, average speed is same as merge sort, but no tricky merging like mergesort, recursive implementation is easy.
Cons: sorted lists or arrays is the worst case, which is $O(n^2)$, iterative implementation is not easy.
Mergesort – pro: worst case time complexity is O(nlogn), so it's guaranteed to be O(nlogn), simple to understand.
Cons: requires much more memory as the original array, not as fast as quicksort on average.

5. For merge sort, it will be O(nlogn), because the best, average, and worst case for merge sort is always O(nlogn). For quick sort, it will be $O(n^2)$, because there are only two values, so after the first round of breaking into sub arrays or sub lists, the sub arrays or sub lists are sorted with all 1s in one array or list and 0s in another. Sorted and semi-sorted lists or arrays are the worst case for quick sort, which is $O(n^2)$.

6. First, use quicksort to sort the collection. Then, loop or iterate through the collection and remove any consecutive identical elements. To do that, you compare the current element with the next element, if they are the same, remove the second element and shift the index up, then compare the next element. If different move the pointer of the current element index to the index of element after the current. Keep doing this until end of collection.

Bonus Question: n/logn is the maximum number of inputs. Because to sort n elements, comparison sorts must take Ω(nlogn) comparison in the worst case.

References

Fung, Carol. Lecture.

Goodrich, Michael T., and Roberto Tamassia. Data Structures and Algorithms in Java. New York: John Wiley, 2014. Print.