Jason Chen
Assignment 5
CMSC 256

Questions

1. Quick sort – use it when don't need stable sort, wants good storage space, and average case matters more than worst case. Best for unsorted lists or arrays. Because average time complexity is O(nlogn), worst case is $O(n^2)$. Storage is O(logn).

Counting sort – use it when the number of possible values elements can be is small compared to the total number of elements and want a stable sort. Because time complexity is O(n+k) with k being the possible number of values of elements. Storage is O(n+k).

Bucket sort – use it when inputs are uniformly distributed across the range. Because time complexity is O(n) when inputs are uniformly distributed. Storage is O(n).

Radix sort – use it when inputs are taken from a large set of numbers and the number of digits are small. Because time complexity is O(d(n+k)) with d being number of digits and k being number of possible values for each digit. Storage is O(n+k).

Insertion sort – use it when you need a stable sort, wants excellent space complexity, and when number of elements is small. Because the best case time complexity is O(n) and space complexity is O(1).

2. n*(n-1)/2 number of edges.
Induction
Base case: if n=0, then n(n-1)/2 = 0(0-1)/2=0. That's true.
Induction: show n+1 is true.
(n+1)(n+1-1)/2 = n(n+1)/2
Add another vertex to the graph
n(n+1)/2 + n
= n(n+1)/2 +2n/2
= $(n^2+n)/2$
=n(n+1)/2
It is true for all n greater or equal to 0.

3.

	0	1	2	3	5
0	0	0	0	1	1
1	0	0	1	0	1
2	0	0	0	0	1
3	0	0	1	0	0
5	0	0	0	1	0

Adj[0] = {3,5}
Adj[1] = {2,5}
Adj[2] = {5}
Adj[3] = {2}
Adj[5] = {3}

4.
1. Compute all indegrees
2. Put all indegree 0 nodes into a Collection
3. Print and remove a node from Collection
4. Decrement indegrees of the node's neighbors
5. If any neighbor has indegree 0, place in Collection. Go to step 3

a - 2
b - 1
c - 1
d – 0
0 – 2
1 – 1
2 – 2
3 – 3
5 - 3
6 - 3
7 - 2
8 - 1
9 - 2

Collection: d, 1, 0, 2, 5, c, b, 9, 3, 6, 7, 8, a
Result: d, 1, 0, 2, 5, c, b, 9, 3, 6, 7, 8, a

5.
K = 0

	0	1	2	3	4
0	-	-	-	3	1
1	1	-	6	4	2
2	-	1	-	-	5
3	-	-	1	-	-
4	-	-	-	2	-

K=1

	0	1	2	3	4
0	-	-	-	3	1
1	1	-	6	4	2
2	2	1	7	5	4
3	-	-	1	-	-

4	-	-	-	2	-

K=2

	0	1	2	3	4
0	-	-	-	3	1
1	1	7	6	4	2
2	2	1	7	5	4
3	3	2	1	6	5
4	-	-	-	2	-

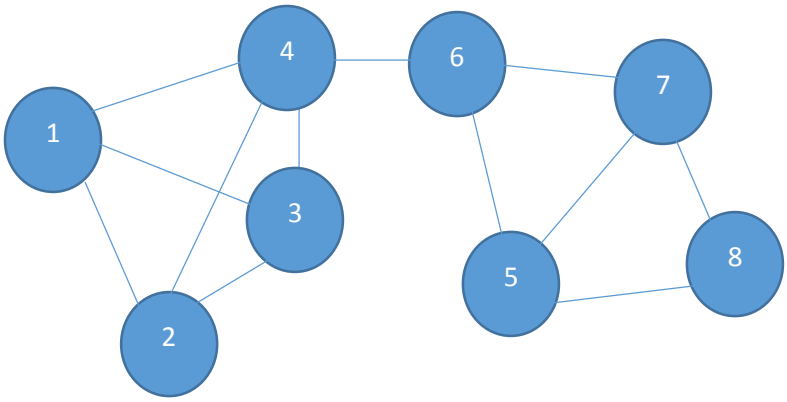
K=3

	0	1	2	3	4
0	6	5	4	3	1
1	1	7	6	4	2
2	2	1	7	5	4
3	3	2	1	6	5
4	5	4	3	2	6

K=4

	0	1	2	3	4
0	6	5	4	3	1
1	1	7	6	4	2
2	2	1	7	5	4
3	3	2	1	6	5
4	5	4	3	2	6

6.
a)



b) BFS traversal: 1, 2, 3, 4, 6, 5, 7, 8

Bonus Question:
Put p into list L 1
Following parent pointer h

Add q to L	1
Update parent pointer	h*h

Total: 1+h+1+h*h
$\Omega(h^2)$

References

Fung, Carol. Lecture.

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