Chapter 9 summary

* **Representation of graphs –** a way to represent a graph is using a two dimensional array, also called adjacency matrix. For each edge (u,v) we set A[u][v] to true, otherwise the array is false. Adjacency matrix is appropriate if the graph is dense, if the graph is not dense a solution is an adjacency list representation (for each vertex we keep a list of all adjacent vertices)
* **Digraph –** is a directed graph
* **Path –** a sequence of vertices in a graph
* **Length –** the number of edge on the path, which is equal to N-1
* **Simple Path –** a path such that all vertices are distinct, except that the first and last could be the same
* **Loop –** when there is a path from its vertex to itself
* **Cycle –** in a directed graph is a path of length at least 1, such w1=wN
* **Topological sort –** is an ordering of vertices in a directed acyclic graph, such that if there is a path from vi vj, then vj appears after vi in the ordering.
* **Dijkstra’s algorithm** – the general method to solving the single source shortest path problem. It is a prime example of a greedy algorithm, meaning it generally solves a problem in stages by doing what appears to be the best thing at each stage. Dijkstras algorithm proceeds in stages, just like the unweighted shortest path algorithm and at each stage, Dijkstras algorithm selects a vertex v, which has the smallest dv among all unknown vertices and declares that the shortest path from s to v is known.
* **Graphs with negative edge costs** – if this is true, Dijkstras algorithm doesn’t work. In this case, taking a path from s to v back to u is better than going from s to u without using v. A solution could be to add a constant to each edge cost, results in removing negative edges (calculates a shortest path on the new graph, then uses that result on the original.) A combination of weighted and unweighted algorithms will solve the problem but causes increased running time.
* **Acyclic graphs** – a directed graph with no cycles. If the graph is acyclic, we can improve Dijkstras algorithm by changing the order in which vertices are declared known, also known as the vertex selection rule. The new rule is to select vertices in topological order and the algorithm can be done in one pass.
* **All pairs shortest path** – when it is important to find the shortest paths between all pairs of vertices in the graph, use the O(lVl^3) algorithm to solve this problem for weighted graphs
* **Simple maximum flor problem** – involve finding a feasible flow through a single source, single sink flow network that is maximum.
* **Minimum spanning tree –** in an undirected graph, a minimum spanning tree of an undirected graph G is a tree formed from graph edges that connects all the vertices of G at lowest total cost. A minimum spanning tree exists if and only if G is connected.
* **Prims algorithm –** a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph, meaning it finds a subset of edges that forms a tree that includes every vertex where the total weight of all edges in the tree is minimized. Essentially identical to Dijkstras algorithm for shortest paths with exception that since the definition of dv is different, so is the update rule.
* **Kruskals Algorithm –** another greedy strategy, you continually select the edges in order of smallest weight and accept an edge if it does not cause a cycle. Kruskals algorithm maintains a forest (collection of trees) there are lVl single node trees. Adding an edge merges two trees into one, and when the algorithm terminates, there is only one tree left which is the minimum spanning tree.
* **Undirected Graphs –** an undirected graph is connected if and only if a depth first search starting from any node visits every node.
* **Biconnectivity –** a connected undirected graph is biconnected if there are no vertices whose removal disconnects the rest of the graph. If a graph is not biconnected, the vertices whose removal would disconnect the graph are known as articulation points.
* **Euler circuit –** a eulerian path is a trail in a graph which visits every edge exactly once. A eulerian circuit is an eulerian trail which starts and ends on the same vertex.
* **Directed Graphs –** is a graph or set of vertices connected by edges where the edges have a direction associated with them. Can be traversed in linear time, using depth first search. If the graph is not strongly connected, a depth first search starting at some node might not visit all nodes.
* **Class NP –** NP stands for nondeterministic polynomial time. A deterministic machine at each point in time is executing an instruction. Depending on the instruction, it then goes to some next instruction, which is unique. A nondeterministic machine has a choice of next steps, it is free to choose any that it wishes. If one of these steps leads to a solution, it will always choose the correct one.

Chapter 10 summary

* **Greedy algorithms –** works in phases, in each phase a decision is made that appears to be good, without regard of future consequences (local optimum chosen). When the algorithm terminates we hope that the local optimum is equal to the global optimum.
* **Huffman algorithm –** maintain a forest of trees, the weight of a tree is equal to the sum of the frequencies of its leaves. C-1 times, select the two trees T1 and T2, of the smallest weight breaking ties arbitrarily, and form a new tree with subtrees T1 and T2.
* **Online algorithms**
* **Bin packing problem –** these algorithms will run quickly but will not necessarily produce optimal solutions. Online bin packing (each item must be placed in a bin before the next item can be processed) and Offline bin packing (do not need to do anything until all the input has been read)
* **Next fit –** when processing and item, we check to see whether it fits in the same bin as the last item, if it does it is placed there, otherwise a new bin is created.
* **First fit –** scan the bins in order and place the new items in the first bin that is large enough to hold it. Thus a new bin is created only when the results of previous placements have left no other alternative.
* **Best fit –** instead of placing a new item in the first spot that is found, It is placed in the tightest spot amongst all bins
* **Divide and conquer –** consists of two parts, first divide (smaller problems are solved recursively, except, of course, base cases), and conquer (the solution to the original problem s then formed from the solutions to the subproblems).
* **Running time of divide and conquer algorithms –** O(nxlog(n))
* **Selection point problem –** requires us to find the kth smallest element in a collection S of N elements. Of the particular interests is the special case of finding the median. This occurs when k = [N/2].
* **Dynamic programming –** is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions ideally using a memory based data structure.
* **Optimal binary search tree –** a dynamic programming example which considers the following output: given a lost of words w1,w2….wn and fixed probabilities p1,p2,…pn of their occurrence. The problem is to arrange these words in a binary search tree in a way that minimizes the expected total access time.
* **All pairs shortest path –** another example of dynamic programming which is an algorithm that computes the shortest weighted paths between every pair of points in a directed graph G = (V,E).
* **Randomized algorithms –** an algorithm that employs a degree of randomness as part of its logic. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behavior, in the hope of achieving a good performance in the “average case” over all possible choices of random bits.
* **Skip lists –**  a randomized data structure that supports both searching and insertion in O(logN) expected time.
* **Primality testing -**  determining whether or not a number is prime.
* **Backtracking algorithms –** a general algorithm for finding all (or some) solutions to some computational problems, notably constraint satisfaction problems, that incrementally builds candidates to the solutions, and abandons each partial candidate C as soon as it determines that C cannot be completed to a valid solution
* **Turnpike reconstruction problem –** is to reconstruct a point set from the distances.
* **Games – minimax strategy :** when the value of the position is determined by recursively assuming optimal play by both sides. **Terminal position:** a position for which this assignment can be determined by examining the board.