ADA LAB Assignment Questions and Java Solutions (Clean Version)

ADA LAB Assignment Questions and Java Solutions

Question 1

1. Write a function of polynomial time complexity to convert a CNF-Boolean function into equivalent graph. Thereafter, write a program to read a CNF-Boolean function and check whether it is satisfiable or not by checking the corresponding graph having a clique of k vertices or not, where k is number of sum terms in the CNF-Boolean function.

Java Solution:

```
import java.util.*;
class Q1_CNFToGraphClique {
    static class LitNode { int clause; String lit; LitNode(int c, String
l){clause=c;lit=l;} }
    public static boolean isComplement(String a, String b){
        if(a.equals("-"+b) || b.equals("-"+a)) return true;
        return false;
    }
    public static boolean hasClique(List<List<String>> clauses){
        int m = clauses.size();

        List<LitNode> nodes = new ArrayList<>();
        for(int i=0;i<m;i++){
            for(String lit: clauses.get(i)) nodes.add(new LitNode(i, lit));
        }
        int N = nodes.size();
        boolean[][] adj = new boolean[N][N];
        for(int u=0;u<N;u++){
            for(int v=0;v<N;v++){
                if(nodes.get(u).clause != nodes.get(v).clause){
                    if(!isComplement(nodes.get(u).lit, nodes.get(v).lit))
adj[u][v]=true;
                }
            }
        }

        List<Integer>[] byClause = new List[m];
        for(int i=0;i<m;i++) byClause[i]=new ArrayList<>();
        for(int idx=0;idx<N;idx++) byClause[nodes.get(idx).clause].add(idx);

        return dfsChoose(0, m, byClause, new int[m], adj);
    }
    static boolean dfsChoose(int pos, int m, List<Integer>[] byClause, int[]
chosen, boolean[][] adj){
        if(pos==m){

            for(int i=0;i<m;i++) for(int j=i+1;j<m;j++)
```

```
if(!adj[chosen[i]][chosen[j]]) return false;
            return true;
        }
        for(int node: byClause[pos]){
            chosen[pos]=node;
            if(dfsChoose(pos+1,m,byClause,chosen,adj)) return true;
        }
        return false;
    }

    public static void main(String[] args){

        List<List<String>> clauses = new ArrayList<>();
        clauses.add(Arrays.asList("x1","x2","-x3"));
        clauses.add(Arrays.asList("-x1","x3","x4"));
        clauses.add(Arrays.asList("x2","-x4","x5"));
        System.out.println("Satisfiable (by clique method)?: "+
hasClique(clauses));
    }
}
```

Question 2

2. Write a function in polynomial time complexity to convert an undirected graph into equivalent CNF-Boolean function. Thereafter, write a program to read adjacency matrix of the graph and check whether it is having a clique of k vertices or not by checking the corresponding CNF-Boolean function's satisfiability, where k is entered by user.

Java Solution:

```
import java.util.*;
class Q2_GraphToCNF_CheckClique {

    static class Clause { List<String> lits = new ArrayList<>(); }
    static List<Clause> genCNF(int[][] adj, int k){
        int n = adj.length;
        List<Clause> clauses = new ArrayList<>();

        for(int p=0;p<k;p++){
            Clause c = new Clause();
            for(int v=0;v<n;v++) c.lits.add("x"+v+"_"+p);
            clauses.add(c);
        }

        for(int v=0;v<n;v++){
            for(int p=0;p<k;p++) for(int q=p+1;q<k;q++){
                Clause c = new Clause(); c.lits.add("-x"+v+"_"+p);
c.lits.add("-x"+v+"_"+q); clauses.add(c);
            }
        }
```

```
        for(int p=0;p<k;p++) for(int q=p+1;q<k;q++){
            for(int u=0;u<n;u++) for(int v=0;v<n;v++){
                if(u==v) continue;
                if(adj[u][v]==0){
                    Clause c = new Clause(); c.lits.add("-x"+u+"_"+p);
c.lits.add("-x"+v+"_"+q); clauses.add(c);
                }
            }
        }
        return clauses;
    }

    static boolean hasKClique(int[][] adj, int k){
        int n = adj.length;
        int[] comb = new int[k];
        return combine(0,0,n,k,comb,adj);
    }
    static boolean combine(int start,int idx,int n,int k,int[] comb,int[][]
adj){
        if(idx==k){
            for(int i=0;i<k;i++) for(int j=i+1;j<k;j++)
if(adj[comb[i]][comb[j]]==0) return false;
            return true;
        }
        for(int v=start;v<n;v++){
            comb[idx]=v;
            if(combine(v+1,idx+1,n,k,comb,adj)) return true;
        }
        return false;
    }
    public static void main(String[] args){
        int[][] adj = {{0,1,1,0},{1,0,1,1},{1,1,0,1},{0,1,1,0}};
        int k=3;
        System.out.println("Has "+k+"-clique?: "+ hasKClique(adj,k));
        List<Clause> cnf = genCNF(adj,k);
        System.out.println("Generated CNF clauses count: "+cnf.size());
    }
}
```

Question 3

3. Write a program for Traversing Sales Person (TSP) problem using best first search branch and bound method.

Java Solution:

```
import java.util.*;
class Q3_TSP_BestFirst {
    static class State implements Comparable<State>{
        List<Integer> path; double cost; boolean[] visited;
        State(List<Integer> p,double c,boolean[] v){path=p;cost=c;visited=v;}
```

```
        public int compareTo(State o){ return Double.compare(this.cost,o.cost);
}
    }
    static double tsp(double[][] dist){
        int n = dist.length;
        PriorityQueue<State> pq = new PriorityQueue<>();
        boolean[] vis0 = new boolean[n]; vis0[0]=true;
        pq.add(new State(new ArrayList<>(Arrays.asList(0)),0.0,vis0));
        double best = Double.POSITIVE_INFINITY;
        while(!pq.isEmpty()){
            State s = pq.poll();
            if(s.path.size()==n){
                double total = s.cost + dist[s.path.get(n-1)][0];
                best = Math.min(best, total);
                continue;
            }
            for(int v=0;v<n;v++) if(!s.visited[v]){
                List<Integer> np = new ArrayList<>(s.path); np.add(v);
                boolean[] nv = s.visited.clone(); nv[v]=true;

                double minEdge = Double.POSITIVE_INFINITY;
                for(int i=0;i<n;i++) for(int j=0;j<n;j++) if(i!=j) minEdge =
Math.min(minEdge, dist[i][j]);
                double lb = s.cost + dist[s.path.get(s.path.size()-1)][v] + (n
- np.size())*minEdge;
                if(lb < best) pq.add(new State(np, s.cost +
dist[s.path.get(s.path.size()-1)][v], nv));
            }
        }
        return best;
    }
    public static void main(String[] args){
        double[][] dist = {
            {0,10,15,20},
            {10,0,35,25},
            {15,35,0,30},
            {20,25,30,0}
        };
        System.out.println("TSP best tour cost: "+ tsp(dist));
    }
}
```

Question 4

4. Write a program using single one dimensional array of minimum size to get binomial coefficient "C(n, k)" by using dynamic programming approach.

Java Solution:

```
import java.util.*;
class Q4_Binomial1D {
```

```
    static long binomial(int n,int k){
        if(k<0 || k>n) return 0;
        k = Math.min(k, n-k);
        long[] C = new long[k+1];
        C[0]=1;
        for(int i=1;i<=n;i++){
            for(int j=Math.min(i,k); j>0; j--) C[j] = C[j] + C[j-1];
        }
        return C[k];
    }
    public static void main(String[] args){
        System.out.println("C(5,2) = "+ binomial(5,2));
        System.out.println("C(10,3) = "+ binomial(10,3));
    }
}
```

Question 5

5. Write a dynamic programming program for chained matrix multiplication problem to get the optimal order.

Java Solution:

```
import java.util.*;
class Q5_MatrixChain {
    static int matrixChainOrder(int[] p){
        int n = p.length-1;
        int[][] m = new int[n+1][n+1];
        for(int L=2;L<=n;L++){
            for(int i=1;i<=n-L+1;i++){
                int j=i+L-1;
                m[i][j]=Integer.MAX_VALUE;
                for(int k=i;k<=j-1;k++){
                    int q = m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
                    if(q < m[i][j]) m[i][j]=q;
                }
            }
        }
        return m[1][n];
    }
    public static void main(String[] args){
        int[] dims = {40,20,30,10,30};
        System.out.println("Minimum multiplications: "+
matrixChainOrder(dims));
    }
}
```

Question 6

6. Write a program to get minimum spanning tree of a given graph using Kruskal's algorithm.

Java Solution:

```
import java.util.*;
class Q6_Kruskal {
    static class Edge implements Comparable<Edge>{ int u,v,w; Edge(int a,int
b,int c){u=a;v=b;w=c;} public int compareTo(Edge o){return this.w-o.w;} }
    static int find(int[] p,int x){ return p[x]==x?x:p[x]=find(p,p[x]); }
    static List<Edge> kruskal(int n,List<Edge> edges){
        Collections.sort(edges);
        int[] parent = new int[n]; for(int i=0;i<n;i++) parent[i]=i;
        List<Edge> res = new ArrayList<>();
        for(Edge e: edges){
            int a = find(parent,e.u), b = find(parent,e.v);
            if(a!=b){ parent[a]=b; res.add(e); }
        }
        return res;
    }
    public static void main(String[] args){
        int n=4;
        List<Edge> edges = Arrays.asList(new Edge(0,1,10),new Edge(0,2,6),new
Edge(0,3,5),new Edge(1,3,15),new Edge(2,3,4));
        List<Edge> mst = kruskal(n, new ArrayList<>(edges));
        System.out.println("MST edges:");
        for(Edge e: mst) System.out.println(e.u+" - "+e.v+" : "+e.w);
    }
}
```

Question 7

7. Write an efficient program to print truth table of n Boolean variables.

Java Solution:

```
import java.util.*;
class Q7_TruthTable {
    static void printTruthTable(int n){
        int rows = 1<<n;
        for(int i=0;i<rows;i++){
            for(int j=n-1;j>=0;j--){
                System.out.print(((i>>j)&1)==1 ? "1 " : "0 ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args){
        printTruthTable(3);
```

```
    }
}
```

Question 8

8. Write a program to multiply two square matrices of order 2^n using Strassen's matrix multiplication algorithm.

Java Solution:

```
import java.util.*;
class Q8_Strassen {
    static int[][] add(int[][] A,int[][] B){
        int n=A.length; int[][] C=new int[n][n];
        for(int i=0;i<n;i++) for(int j=0;j<n;j++) C[i][j]=A[i][j]+B[i][j];
        return C;
    }
    static int[][] sub(int[][] A,int[][] B){
        int n=A.length; int[][] C=new int[n][n];
        for(int i=0;i<n;i++) for(int j=0;j<n;j++) C[i][j]=A[i][j]-B[i][j];
        return C;
    }
    static int[][] strassen(int[][] A,int[][] B){
        int n=A.length;
        if(n==1){ return new int[][]{{A[0][0]*B[0][0]}}; }
        int newN = n/2;
        int[][] A11=new int[newN][newN],A12=new int[newN][newN],A21=new
int[newN][newN],A22=new int[newN][newN];
        int[][] B11=new int[newN][newN],B12=new int[newN][newN],B21=new
int[newN][newN],B22=new int[newN][newN];
        for(int i=0;i<newN;i++) for(int j=0;j<newN;j++){
            A11[i][j]=A[i][j]; A12[i][j]=A[i][j+newN];
            A21[i][j]=A[i+newN][j]; A22[i][j]=A[i+newN][j+newN];
            B11[i][j]=B[i][j]; B12[i][j]=B[i][j+newN];
            B21[i][j]=B[i+newN][j]; B22[i][j]=B[i+newN][j+newN];
        }
        int[][] M1 = strassen(add(A11,A22), add(B11,B22));
        int[][] M2 = strassen(add(A21,A22), B11);
        int[][] M3 = strassen(A11, sub(B12,B22));
        int[][] M4 = strassen(A22, sub(B21,B11));
        int[][] M5 = strassen(add(A11,A12), B22);
        int[][] M6 = strassen(sub(A21,A11), add(B11,B12));
        int[][] M7 = strassen(sub(A12,A22), add(B21,B22));
        int[][] C = new int[n][n];
        int[][] C11 = add(sub(add(M1,M4),M5),M7);
        int[][] C12 = add(M3,M5);
        int[][] C21 = add(M2,M4);
        int[][] C22 = add(sub(add(M1,M3),M2),M6);
        for(int i=0;i<newN;i++) for(int j=0;j<newN;j++){
            C[i][j]=C11[i][j]; C[i][j+newN]=C12[i][j];
            C[i+newN][j]=C21[i][j]; C[i+newN][j+newN]=C22[i][j];
```

```
        }
        return C;
    }
    public static void main(String[] args){
        int[][] A = {{1,2,3,4},{5,6,7,8},{9,8,7,6},{5,4,3,2}};
        int[][] B = {{2,0,1,3},{1,2,0,4},{3,1,2,0},{2,3,1,1}};
        int[][] C = strassen(A,B);
        System.out.println("Result matrix:");
        for(int i=0;i<C.length;i++){ for(int j=0;j<C.length;j++)
System.out.print(C[i][j]+" "); System.out.println(); }
    }
}
```

Question 9

9. Write a program to get Longest Common Subsequence of two given sequences using dynamic programming approach.

Java Solution:

```
import java.util.*;
class Q9_LCS {
    static String lcs(String A,String B){
        int n=A.length(), m=B.length();
        int[][] dp = new int[n+1][m+1];
        for(int i=1;i<=n;i++) for(int j=1;j<=m;j++)
            if(A.charAt(i-1)==B.charAt(j-1)) dp[i][j]=dp[i-1][j-1]+1; else
dp[i][j]=Math.max(dp[i-1][j],dp[i][j-1]);
        StringBuilder sb = new StringBuilder();
        int i=n,j=m;
        while(i>0 && j>0){
            if(A.charAt(i-1)==B.charAt(j-1)){ sb.append(A.charAt(i-1)); i--; j-
-; }
            else if(dp[i-1][j]>dp[i][j-1]) i--; else j--;
        }
        return sb.reverse().toString();
    }
    public static void main(String[] args){
        System.out.println(lcs("AGGTAB","GXTXAYB"));
    }
}
```

Question 10

10. Write a program to get minimum spanning tree of a given graph using Prim's algorithm.

Java Solution:

```
import java.util.*;
class Q10_Prim {
    static void prim(int[][] graph){
        int n=graph.length;
        boolean[] inMST = new boolean[n];
        int[] key = new int[n]; Arrays.fill(key,Integer.MAX_VALUE);
        int[] parent = new int[n]; Arrays.fill(parent,-1);
        key[0]=0;
        for(int count=0;count<n-1;count++){
            int u=-1; int min=Integer.MAX_VALUE;
            for(int v=0;v<n;v++) if(!inMST[v] && key[v]<min){min=key[v];u=v;}
            inMST[u]=true;
            for(int v=0;v<n;v++) if(graph[u][v]!=0 && !inMST[v] &&
graph[u][v]<key[v]){ parent[v]=u; key[v]=graph[u][v]; }
        }
        System.out.println("Edge \tWeight");
        for(int i=1;i<n;i++) System.out.println(parent[i]+" - "+i+"
\t"+graph[i][parent[i]]);
    }
    public static void main(String[] args){
        int[][]
g={{0,2,0,6,0},{2,0,3,8,5},{0,3,0,0,7},{6,8,0,0,9},{0,5,7,9,0}};
        prim(g);
    }
}
```

Question 11

11. Write an efficient program to remove duplicate character from string and determine its worst case time complexity.

Java Solution:

```
import java.util.*;
class Q11_RemoveDuplicates {
    static String removeDuplicates(String s){
        boolean[] seen = new boolean[256];
        StringBuilder sb = new StringBuilder();
        for(char c: s.toCharArray()){
            if(!seen[c]){ sb.append(c); seen[c]=true; }
        }
        return sb.toString();
    }

    public static void main(String[] args){
        System.out.println(removeDuplicates("banana"));
    }
}
```

Question 12

12. Write a program for 0-1 knapsack problem using dynamic programming algorithm.

Java Solution:

```
import java.util.*;
class Q12_Knapsack01 {
    static int knapSack(int W,int[] wt,int[] val,int n){
        int[][] dp = new int[n+1][W+1];
        for(int i=1;i<=n;i++){
            for(int w=0;w<=W;w++){
                dp[i][w]=dp[i-1][w];
                if(wt[i-1]<=w) dp[i][w]=Math.max(dp[i][w], dp[i-1][w-wt[i-
1]]+val[i-1]);
            }
        }
        return dp[n][W];
    }
    public static void main(String[] args){
        int[] val={60,100,120}; int[] wt={10,20,30}; int W=50;
        System.out.println("Max value: "+ knapSack(W,wt,val,3));
    }
}
```

Question 13

13. Write a program for n-queen problem using backtracking algorithm.

Java Solution:

```
import java.util.*;
class Q13_NQueen {
    static boolean isSafe(int[] board,int row,int col){
        for(int i=0;i<row;i++){
            if(board[i]==col || Math.abs(board[i]-col)==row-i) return false;
        }
        return true;
    }
    static void solveNQUtil(int row,int n,int[] board,List<List<Integer>>
sols){
        if(row==n){ List<Integer> sol=new ArrayList<>(); for(int v:board)
sol.add(v); sols.add(sol); return; }
        for(int col=0;col<n;col++){
            if(isSafe(board,row,col)){ board[row]=col;
solveNQUtil(row+1,n,board,sols); }
        }
    }
    public static void main(String[] args){
        int n=8; int[] board=new int[n]; List<List<Integer>> sols=new
ArrayList<>();
```

```
        solveNQUtil(0,n,board,sols);
        System.out.println("Number of solutions for "+n+"-Queens:
"+sols.size());
    }
}
```

Question 14

14. Write an efficient program to remove multiple blank spaces from a given file and determine its time complexity.

Java Solution:

```
import java.io.*;
class Q14_RemoveExtraSpaces {
    static String normalize(String s){
        return s.replaceAll("\\\\s+"," ");
    }
    public static void main(String[] args) throws Exception{
        String input = "This   is   a    line.  Another    line.";
        String out = normalize(input);
        System.out.println(out);
    }
}
```

Question 15

15. Write a program to generate all valid shifts to match a pattern in a text using naïve algorithm.

Java Solution:

```
import java.util.*;
class Q15_PatternShifts {
    static List<Integer> naiveMatch(String text,String pat){
        List<Integer> res=new ArrayList<>();
        int n=text.length(), m=pat.length();
        for(int s=0;s<=n-m;s++){
            int j=0;
            while(j<m && text.charAt(s+j)==pat.charAt(j)) j++;
            if(j==m) res.add(s);
        }
        return res;
    }
    public static void main(String[] args){
        System.out.println(naiveMatch("ABABABABA","ABA"));
```

```
    }
}
```

Question 16

16. Write a program for 0-1 knapsack problem using best first search branch and bound algorithm.

Java Solution:

```
import java.util.*;
class Q16_KnapsackBFS_BB {
    static class Item{ int wt; int val; double ratio; Item(int w,int
v){wt=w;val=v; ratio=(double)v/w;} }
    static class Node implements Comparable<Node>{ int level,profit,weight;
double bound; Node(int l,int p,int w){level=l;profit=p;weight=w;} public int
compareTo(Node o){ return Double.compare(o.bound,o.bound); } }
    static double bound(Node u,int n,int W,Item[] items){
        if(u.weight>=W) return 0;
        double result = u.profit;
        int j=u.level+1; int totweight=u.weight;
        while(j<n && totweight+items[j].wt<=W){ totweight+=items[j].wt;
result+=items[j].val; j++; }
        if(j<n) result += (W - totweight) * items[j].ratio;
        return result;
    }
    static int knap(Item[] items,int W){
        int n=items.length;
        Arrays.sort(items, (a,b)->Double.compare(b.ratio,a.ratio));
        Comparator<Node> cmp = (a,b)-> Double.compare(b.bound,a.bound);
        PriorityQueue<Node> pq = new PriorityQueue<>(cmp);
        Node u = new Node(-1,0,0); u.bound = bound(u,n,W,items); pq.add(u);
        int maxProfit = 0;
        while(!pq.isEmpty()){
            u = pq.poll();
            if(u.bound <= maxProfit) continue;
            if(u.level==n-1) continue;
            Node v = new Node(u.level+1, u.profit + items[u.level+1].val,
u.weight + items[u.level+1].wt);
            if(v.weight <= W && v.profit > maxProfit) maxProfit = v.profit;
            v.bound = bound(v,n,W,items);
            if(v.bound > maxProfit) pq.add(v);
            Node v2 = new Node(u.level+1, u.profit, u.weight);
            v2.bound = bound(v2,n,W,items);
            if(v2.bound > maxProfit) pq.add(v2);
        }
        return maxProfit;
    }
    public static void main(String[] args){
        Item[] items = { new Item(10,60), new Item(20,100), new Item(30,120) };
        int W=50;
```

```
        System.out.println("Max profit (B&B): "+ knap(items,W));
    }
}
```

Question 17

17. Write a program for sum of subset problem using backtracking algorithm.

Java Solution:

```
import java.util.*;
class Q17_SumOfSubset {
    static boolean found=false;
    static void subsetSum(int[] arr,int idx,int sum,int target,List<Integer>
curr){
        if(sum==target){ System.out.println(curr); found=true; return; }
        if(idx==arr.length || sum>target) return;

        curr.add(arr[idx]); subsetSum(arr,idx+1,sum+arr[idx],target,curr);
        if(found) return;
        curr.remove(curr.size()-1);

        subsetSum(arr,idx+1,sum,target,curr);
    }
    public static void main(String[] args){
        int[] arr={3,34,4,12,5,2}; int target=9;
        subsetSum(arr,0,0,target,new ArrayList<>());
    }
}
```

Question 18

18. Write a program to generate the Huffman code for given list of characters and their frequencies.

Java Solution:

```
import java.util.*;
class Q18_Huffman {
    static class Node implements Comparable<Node>{
        char ch; int freq; Node left,right;
        Node(char c,int f){ch=c;freq=f;}
        public int compareTo(Node o){ return this.freq - o.freq; }
    }
    static void printCodes(Node root, String s){
        if(root.left==null && root.right==null && root.ch!='\0')
System.out.println(root.ch + ": " + s);
        if(root.left!=null) printCodes(root.left, s+"0");
```

```
        if(root.right!=null) printCodes(root.right, s+"1");
    }
    public static void main(String[] args){
        char[] chars = {'a','b','c','d','e','f'};
        int[] freq = {5,9,12,13,16,45};
        PriorityQueue<Node> pq = new PriorityQueue<>();
        for(int i=0;i<chars.length;i++) { Node nd = new Node(chars[i],
freq[i]); pq.add(nd); }
        while(pq.size()>1){
            Node x = pq.poll(); Node y = pq.poll();
            Node z = new Node('\0', x.freq + y.freq);
            z.left = x; z.right = y;
            pq.add(z);
        }
        printCodes(pq.peek(), "");
    }
}
```

Question 19

19. Write a program for m-coloring problem using backtracking algorithm.

Java Solution:

```
import java.util.*;
class Q19_MColoring {
    static boolean isSafe(int v,int[][] graph,int[] color,int c,int m){
        for(int i=0;i<graph.length;i++) if(graph[v][i]==1 && color[i]==c)
return false;
        return true;
    }
    static boolean graphColoringUtil(int v,int[][] graph,int[] color,int m){
        if(v==graph.length) return true;
        for(int c=1;c<=m;c++){
            if(isSafe(v,graph,color,c,m)){ color[v]=c;
if(graphColoringUtil(v+1,graph,color,m)) return true; color[v]=0; }
        }
        return false;
    }
    static boolean graphColoring(int[][] graph,int m){
        int V = graph.length; int[] color = new int[V];
        if(graphColoringUtil(0,graph,color,m)){ System.out.println("Coloring:
"+ Arrays.toString(color)); return true; }
        else return false;
    }
    public static void main(String[] args){
        int[][] graph = {{0,1,1,1},{1,0,1,0},{1,1,0,1},{1,0,1,0}};
        int m=3;
        System.out.println("Colorable with "+m+" colors? "+
graphColoring(graph,m));
    }
```

```
}
```