Laboratory work # 7

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114
Mail: jhlxhrq@163.com

Problem # 1806. Mobile Telegraph

Screenshot from Timus:

9877861	15:44:55 17 May 2022	hduads2022_20321114	1806. Mobile Telegraphs	Java 1.8	Accepted		1.281	27 868 KB

Explanation of algorithm:

1. Use the CMP model to sort the queue.
2. Use the DFS algorithm to do the searching.

Computational complexity of algorithm:

$$O(V)$$

Source code:

```
import java.io.*;
import java.util.*;

public class MobileTelegraph {
    static int SIZE = 50005, c = 0, n;
    static boolean[] isConv = new boolean[SIZE];
    static boolean f = false;
    static int[] a = new int[SIZE];
    static long[] V = new long[SIZE];

    static Map<String, Integer> hm = new HashMap<>();
    static Comparator<Integer> cmp = (o1, o2) -> (int) (V[o1] - V[o2] != 0 ?
V[o1] - V[o2] : o2 - o1);
    static Queue<Integer> Q = new PriorityQueue<>(cmp);
    static String[] S = new String[SIZE];
    static T[] t = new T[SIZE];

    public static void main(String[] args) {
        new MobileTelegraph().run();
    }

    static Scanner sc;
    static PrintWriter out;
    static StringBuilder sb;
```

```
    void run() {
        sc = new Scanner(System.in);
        out = new PrintWriter(new OutputStreamWriter(System.out));
        sb = new StringBuilder();
        s();
        out.flush();
    }

    void s() {
        int[] tg = new int[10];
        int now = 0;
        char c;
        n = sc.nextInt();
        for (int i = 0; i < 10; ++i) {
            tg[i] = sc.nextInt();
        }
        for (int i = 0; i < n; ++i) {
            t[i] = new T(i);
            S[i] = sc.next();
            V[i] = Long.MAX_VALUE;
            hm.put(S[i], i + 1);
        }
        V[0] = 0;
        isConv[0] = true;
        a[MobileTelegraph.c] = 1;
        MobileTelegraph.c++;
        for (int i = 1; i < n || now != n - 1; ++i) {
            sb.append(S[now]);
            for (int j = 0; j < 10; ++j) {
                for (int k = 0; k < 10; ++k) {
                    if ((char) (k + 48) != S[now].charAt(j)) {
                        sb.setCharAt(j, (char) (k + 48));
                        if (hm.get(sb.toString()) != null
&& !isConv[hm.get(sb.toString()) - 1]) {
                            st(tg, now);
                        }
                    }
                }
                sb.setCharAt(j, S[now].charAt(j));
            }
            for (int j = 0; j < 10; ++j) {
                for (int k = j + 1; k < 10; ++k) {
                    c = sb.charAt(j);
                    sb.setCharAt(j, sb.charAt(k));
                    sb.setCharAt(k, c);
                    if (hm.get(sb.toString()) != null
&& !sb.toString().equals(S[now]) && !isConv[hm.get(sb.toString()) - 1]) {
                        st(tg, now);
                    }
                    sb.delete(0, sb.length());
                    sb.append(S[now]);
                }
            }
            sb.delete(0, sb.length());
            if (now == n - 1) break;
            if (Q.peek() == null) break;
            now = Q.poll();
            isConv[now] = true;
        }
        pr();
    }

    void st(int[] tg, int now) {
```

```
            int temp = hm.get(sb.toString()) - 1;
            if (V[temp] > V[now] + tg[v(S[now], sb.toString())]) {
                V[temp] = V[now] + tg[v(S[now], sb.toString())];
                Q.add(temp);
                t[now].add(temp, tg[v(S[now], sb.toString())]);
            }
        }

        void pr() {
            if (V[n - 1] == Long.MAX_VALUE) {
                out.print(-1);
            } else {
                out.println(V[n - 1]);
                dfs(0, n - 1);
                out.println(c);
                for (int i = 0; i < c; ++i) {
                    out.print(a[i] + " ");
                }
            }
        }

        int v(String s1, String s2) {
            int n = 0, i = 0;
            while (true) {
                if (s1.charAt(i) == s2.charAt(i)) {
                    i++;
                    n++;
                } else {
                    break;
                }
            }
            return n;
        }

        boolean dfs(int n, int end) {
            boolean back = false;
            if (n == end) {
                f = true;
                back = true;
            }
            for (P r : t[n].l) {
                if (f) break;
                if (V[n] + r.w > V[r.to.r] || !isConv[r.to.r]) {
                    continue;
                }
                a[c] = r.to.r + 1;
                c++;
                back = back || dfs(r.to.r, end);
            }
            if (!back) {
                c--;
                return false;
            } else {
                return true;
            }
        }

        static class P {
            int w;
            T to;

            P(int l, int w) {
                this.w = w;
                this.to = t[l];
```

```
        }
    }

    static class T {
        int r;
        ArrayList<P> l = new ArrayList<>();

        T(int r) {
            this.r = r;
        }

        void add(int l, int w) {
            this.l.add(new P(l, w));
        }
    }
}
```