- Kruskal算法

```java
import java.util.Arrays;
import java.util.Scanner;

class Edge implements Comparable<Edge> {
    int a, b, w;

    public Edge(int a, int b, int w) {
        this.a = a;
        this.b = b;
        this.w = w;
    }

    public int compareTo(Edge e) {
        return Integer.compare(w, e.w);
    }

}

public class Kruskal {

    static final int N = 100010;
    static Edge[] edges = new Edge[N * 2];
    static int[] p = new int[N];
    static int n, m;
```

```java

    static int find(int x) {
        if(p[x] != x) p[x] = find(p[x]);
        return p[x];
    }

    static void kruskal() {
        int res = 0;
        int cnt = 0;

        Arrays.sort(edges, 0, m);
        for(int i = 1; i <= n; i++) p[i] =
i;

        for(int i = 0; i < m; i++) {
            int a = find(edges[i].a);
            int b = find(edges[i].b);
            if(a != b) {
                p[a] = b;
                res += edges[i].w;
                cnt ++;
            }
        }

      if(cnt < n - 1) {
        System.out.println("impossible");
      }
      else System.out.println(res);
```

```
52
53      }
54
55      public static void main(String[] args)
{
56          Scanner sc = new
Scanner(System.in);
57          n = sc.nextInt();
58          m = sc.nextInt();
59          for(int i = 0; i < m; i++) {
60              int a = sc.nextInt();
61              int b = sc.nextInt();
62              int w = sc.nextInt();
63              edges[i] = new Edge(a, b, w);
64          }
65
66          kruskal();
67
68      }
69
70  }
```

- 拓扑排序

```
1  import java.util.*;
```

```java

public class TopologicalSort {

    static final int N = 100010;
    static int[] h = new int[N];
    static int[] e = new int[N];
    static int[] ne = new int[N];
    static int[] in = new int[N];
    static int n, m, idx;
    static Queue<Integer> queue = new LinkedList<>();
    static List<Integer> list = new ArrayList<>();

    static void add(int a, int b) {
        e[idx] = b;
        in[b] ++;
        ne[idx] = h[a];
        h[a] = idx ++;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        m = sc.nextInt();
        Arrays.fill(h, -1);
```

```
26          while(m -- > 0) {
27              int a = sc.nextInt();
28              int b = sc.nextInt();
29              add(a, b);
30          }
31
32          if(toptsort()) {
33              for(int num : list) {
34                  System.out.print(num + "
"");
35              }
36          }else System.out.println("-1");
37
38      }
39
40      static boolean toptsort() {
41          for(int i = 1; i <= n; i++) {
42              if(in[i] == 0) {
43                  queue.offer(i);
44              }
45          }
46
47          while(!queue.isEmpty()) {
48              int t = queue.poll();
49              list.add(t);
50              for(int i = h[t]; i != -1; i =
ne[i]) {
51                  int j = e[i];
```

```
                in[j] --;
                if(in[j] == 0) {
                    queue.offer(j);
                }
            }
        }

        return list.size() == n;

    }

}
```