模版·Prim(最小生成树)

```
class Node{
    int val;
    int weight;

    Node(int val,int weight){
        this.val=val;
        this.weight=weight;
        }
    }

class Graph{
    private Map<Integer,List<Node>> adjacencyList;

    Graph(int vertices){
        this.adjacencyList=new HashMap<>(vertices);
        for(int i=1;i<=vertices;i++){
            adjacencyList.put(i,new ArrayList<>());
        }
    }

    void addEdge(int source,int destination,int weight){
        adjacencyList.get(source).add(new Node(destination,weight));
        adjacencyList.get(destination).add(new Node(source,weight));
    }

    public List<Node> primMST(){
        PriorityQueue<Node> pq=new PriorityQueue<>(Comparator.comparingInt(node->node.weight));
        Set<Integer> visited=new HashSet<>();
        int weightSum=0;

        int startVertex=1;//选择顶点1作为起点
        visited.add(startVertex);

        for(Nodeneighbor:adjacencyList.get(startVertex)){
            pq.add(new Node(neighbor.val,neighbor.weight));
        }

        while(!pq.isEmpty()){
            NodecurrentNode=pq.poll();
            int currentVertex=currentNode.val;
            int currentWeight=currentNode.weight;

            if(visited.contains(currentVertex)){
                continue;
            }

            visited.add(currentVertex);
            weightSum+=currentWeight;
            for(Nodeneighbor:adjacencyList.get(currentVertex)){
                if(!visited.contains(neighbor.val)){
                    pq.add(new Node(neighbor.val,neighbor.weight));
                }
            }
        }
        return weightSum;
    }
}

public class PrimMST{
    public static void main(String[] args){
        Graph graph=new Graph(5);

        graph.addEdge(1,2,2);
        graph.addEdge(1,3,1);
        graph.addEdge(2,4,4);
        graph.addEdge(1,4,3);
        graph.addEdge(2,5,2);
        graph.addEdge(3,5,4);
        int weightSum=graph.primMST();
        System.out.println("最小生成树边权重之和： "+weightSum);
    }
}
```