```cpp
class Dijkstra {
private:
  struct node {
    int id; bool done; vi to; vll cst; int from; ll d;
  };
  struct pq_t {
    int id; ll d;
    bool operator<(const pq_t & another) const {
      return d != another.d ? d > another.d : id > another.id;
    }
  };
  vector<node> nodes;
  int n, m, source;
public:
  Dijkstra(const vvi &lst, const vvll &cst, int start) {
    n = lst.size();
    nodes.resize(n);
    Loop(i, n) nodes[i] = { i, false, {}, {}, -1, LLONG_MAX };
    Loop(i, n) {
      Loop(j, lst[i].size()) {
        nodes[i].to.push_back(lst[i][j]);
        nodes[i].cst.push_back(cst[i][j]);
      }
    }
    source = start;
    nodes[source].d = 0;
    priority_queue<pq_t> pq;
    pq.push({ nodes[source].id, nodes[source].d });
    while (pq.size()) {
      int a = pq.top().id;
      pq.pop();
      if (nodes[a].done) continue;
      nodes[a].done = true;
      Loop(j, nodes[a].to.size()) {
        int b = nodes[a].to[j];
        if (nodes[b].done) continue;
        ll buf = nodes[a].d + nodes[a].cst[j];
        if (buf < nodes[b].d) {
          nodes[b].d = buf;
          nodes[b].from = a;
          pq.push({ b, nodes[b].d });
        }
      }
    }
    return;
  }
  vi get_path(int v) {
    stack<int> stk;
    stk.push(v);
    int a = v;
    while (nodes[a].from != -1) {
      stk.push(nodes[a].from);
      a = nodes[a].from;
    }
    if (a != source) return {};
    vi ret;
    while (stk.size()) {
      ret.push_back(stk.top());
      stk.pop();
    }
    return ret;
  }
  ll get_dist(int v) {
    return nodes[v].d;
  }
};
```