```cpp
class Maxflow {
private:
  struct edge_t {
    int cap;
  };
  int n, source, sink;
  int result;
  vector<bool> done;
  vector<unordered_map<int, edge_t>> lst;
  int dfs(int a, int t) {
    if (a == t) return 1;
    done[a] = true;
    Loopitr(itr, lst[a]) {
      int b = itr->fst;
      int cap = itr->snd.cap;
      if (!done[b] && cap > 0) {
        if (dfs(b, t)) {
          lst[a][b].cap--;
          lst[b][a].cap++;
          return 1;
        }
      }
    }
    return 0;
  }
  int run_flow(int s, int t, int f) {
    int ret = 0;
    Loop(i, f) {
      done = vector<bool>(n, false);
      if (dfs(s, t)) ret++;
      else break;
    }
    return ret;
  }
public:
  Maxflow(const vvi &lst, const vvi &cap, int s, int t) {
    n = lst.size();
    this->lst.resize(n);
    Loop(i, n) {
      Loop(j, lst[i].size()) {
        this->lst[i][lst[i][j]].cap += cap[i][j];
        this->lst[lst[i][j]][i].cap += 0;
      }
    }
    source = s;
    sink = t;
    result = 0;
    update();
  }
  void add_cap(int s, int t, int dcap, bool update_flag = true) {
    lst[s][t].cap += dcap;
    // program not be ensured when cap. becomes negative
    if (lst[s][t].cap < 0) {
      int df = -lst[s][t].cap;
      run_flow(s, source, df);
      run_flow(sink, t, df);
      lst[s][t].cap += df;
      lst[t][s].cap -= df;
      result -= df;
    }
    if (update_flag) update();
  }
  void update() {
    result += run_flow(source, sink, INT_MAX);
  }
  int get_maxflow() {
    return result;
  }
};
```