```

class Increase {
public:
  int n, flowMax = 1000000;
  vector <vector<int>> p;
  vector<int> parent;

  Increase(int _n, vector <vector<int>> _p) {
    n = _n;
    p = _p;
    parent.resize(n);
  }

  int dfs(int id, int from, int flow) {
    if (parent[id] != id) return 0;
    parent[id] = from;
    if (id == n - 1) return flow;
    for (int i = 0; i < n; i++)
      if (p[id][i] > 0) {
        int next = dfs(i, id, min(flow, p[id][i]));
        if (next > 0) return next;
      }
    return 0;
  }

  int solve() {
    int ret = 0, flow;
    for (int i = 0; i < n; i++) parent[i] = i;

    while ((flow = dfs(0, -1, flowMax)) > 0) {
      int now = n - 1;
      ret += flow;
      while (parent[now] >= 0) {
        p[parent[now]][now] -= flow;
        p[now][parent[now]] += flow;
        now = parent[now];
      }
      for (int i = 0; i < n; i++) parent[i] = i;
    }
    return ret;
  }
};


class Increase {
public:
  int n, flowMax = 1000000;
  vector <map<int, int>> p;
  vector<int> parent;
  vector<int> stack;

  Increase(int n, vector <map<int, int>> &p) : n(n), p(p) {
    parent.resize(n);
  }

  int dfs(int id, int from, int flow) {
    if (parent[id] != id) return 0;
    stack.push_back(id);
    parent[id] = from;
    if (id == n - 1) return flow;
    for (auto &v : p[id])
      if (v.second > 0) {
        int next = dfs(v.first, id, min(flow, v.second));
        if (next > 0) return next;
      }
    return 0;
  }

  int solve() {
    int ret = 0, flow;
    for (int i = 0; i < n; i++) parent[i] = i;

    while ((flow = dfs(0, -1, flowMax)) > 0) {
      int now = n - 1;
      ret += flow;
      while (parent[now] >= 0) {
        p[parent[now]][now] -= flow;
        p[now][parent[now]] += flow;
        now = parent[now];
      }
      for (auto &v: stack) parent[v] = v;
      stack.clear();
    }
    return ret;
  }
};
```