Lösningsförslag Brevoptimering

Vi vill för varje person lista ut hur många brev som skickas till den per sekund, I_b. Låt W_ab beteckna andelen av sina brev person a skickar till person b. Värdet I_b kan vi då räkna ut som summan av $U_a \cdot W_ab$ för alla personer a som skickar något till person b. Om vi nu definerar en funktion `getOutput(v)` som ger person v:s output, kan vi räkna ut värdet på den den rekursivt, genom att anropa `getOutput(u)` för alla personer u som skickar något till v.

För att detta ska bli tillräckligt snabbt krävs dock en viktigt insikt: vi behöver inte räkna ut `getOutput(v)` mer än en gång för varje person v. Så fort vi kommit till slutet att `getOutput(v)` för en viss person v, sparar vi svaret i som `output[v]` i en global array `output` innan vi returnerar det. I början av funktionen `getOutput(v)` kollar vi nu ifall värdet redan är uträknat, och returnerar det i så fall direkt. Att komma ihåg värden för en rekursiv funktion för att slippa räkna ut dem igen kallas för "memoization". Med den här optimeringen blir tidskomplexiteten $O(N + S)$, eftersom vi kommer att gå igenom varje kant i grafen exakt en gång, och varje person exakt en gång.

```
#include<bits/stdc++.h>

using namespace std;

vector<vector<pair<int,double>>> gettingLettersFrom;
//Här sparar vi för varje person v en lista av alla
    personer u den får brev från,
//tillsammans med ett flyttal som beskriver andelen av u:
    s brev som den skickar till v

vector<double> M;
vector<double> output;

double getOutput(int v){
        if(output[v]!=-1) { //Ifall output inte är -1 har
            vi redan räknat ut den
                return output[v];
        }
        if(gettingLettersFrom[v].size()==0) return M[v];

    double in = 0;
    for(int i = 0; i<gettingLettersFrom[v].size(); i++){
        int u = gettingLettersFrom[v][i].first;
        double share = gettingLettersFrom[v][i].second;

        in += getOutput(u)*share;
    }
```

```
    double out = min(in,M[v]);

    return output[v] = out; //spara värdet i output och
        returnera det
}

int main(){
        cin.sync_with_stdio(false);
    int n;
    cin>>n;
    gettingLettersFrom.resize(n);
    M.resize(n);
        output.resize(n,-1); //Vi sätter -1 som defaultvä
            rde innan det riktiga värdet är uträknat

    for(int i = 0; i<n;i++) {
        int k;
        cin>>M[i]>>k;
        for(int l = 0; l<k;l++) {
            int j;
            double w;
            cin>>j>>w;
            gettingLettersFrom[j-1].emplace_back(i,w
                /100.0);
        }
    }

    rep(i,0,n){
        if(getOutput(i)==M[i]) cout<<i+1<<" ";
    }
    cout<<endl;
}
```