Codeforces Round #179 (Div. 1)

Problem B – Greg and Graph

Prof. Edson Alves

Faculdade UnB Gama

Greg has a weighed directed graph, consisting of n vertices. In this graph any pair of distinct vertices has an edge between them in both directions. Greg loves playing with the graph and now he has invented a new game:

- The game consists of n steps.
- On the i-th step Greg removes vertex number x_i from the graph. As Greg removes a vertex, he also removes all the edges that go in and out of this vertex.
- Before executing each step, Greg wants to know the sum of lengths of the shortest paths between all pairs of the remaining vertices. The shortest path can go through any remaining vertex. In other words, if we assume that $d(i, v, u)$ is the shortest path between vertices v and u in the graph that formed before deleting vertex x_i, then Greg wants to know the value of the following sum: $\sum_{v,u,v \neq u} d(i, v, u)$

Help Greg, print the value of the required sum before each step.

Greg tem um grafo ponderado, composto por n vértices. Neste grafo qualquer par de vértices distintos tem uma arestas entre eles em ambas direções. Greg ama brincar com o grafo e agora ele inventou um novo jogo:

- O jogo é composto de n etapas.
- Na i-ésima etapa Greg remove o vértice x_i do grafo. Quando Greg remove um vértice, ele também remove todas as arestas que chegam ou partem deste vértice.
- Antes de cada etapa, Greg quer saber a soma dos custos dos caminhos mínimos entre todos os pares de vértices restantes. O caminho mais curto pode passar por qualquer vértice restante. Em outras palavras, se $d(i, v, u)$ é o custo do caminho mínimo entre os vértices v e u no grafo antes da exclusão do vértice x_i, então Greg quer saber o valor da seguinte soma: $\sum_{v,u,v\neq u} d(i, v, u)$

Help Greg, print the value of the required sum before each step.

Input

The first line contains integer n $(1 \leq n \leq 500)$ – the number of vertices in the graph.

Next n lines contain n integers each – the graph adjacency matrix: the j-th number in the i-th line a_{ij} $(1 \leq a_{ij} \leq 10^5, a_{ii} = 0)$ represents the weight of the edge that goes from vertex i to vertex j.

The next line contains n distinct integers: $x_1, x_2, \ldots, x_n$ $(1 \leq x_i \leq n)$ – the vertices that Greg deletes.

Output

Print n integers – the i-th number equals the required sum before the i-th step.

Entrada

A primeira linha contém o inteiro n $(1 \leq n \leq 500)$ – o número de vértices no grafo.

As próximas n linhas contém n inteiros cada – a matriz de adjacências do grafo: o j-ésimo número na i-ésima linha a_{ij} $(1 \leq a_{ij} \leq 10^5, a_{ii} = 0)$ representa o peso da aresta que vai do vértice i ao vértice j.

A próxima linha contém n inteiros distintos: $x_1, x_2, \ldots, x_n$ $(1 \leq x_i \leq n)$ – os vértices que Greg apagará.

Saída

Imprima n inteiros – o i-ésimo número corresponde à soma requerida antes da i-ésima etapa.

Exemplo de entrada e saída

Exemplo de entrada e saída

2

Exemplo de entrada e saída

2

de vértices

Exemplo de entrada e saída

2

Exemplo de entrada e saída

```
2
0 5
4 0
```

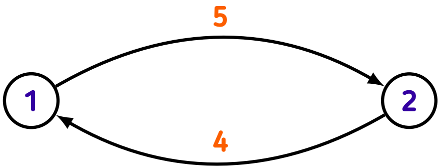
Exemplo de entrada e saída

```
2
0 5
4 0
```

matriz de adjacências

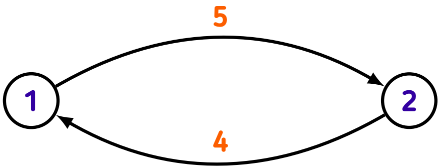
Exemplo de entrada e saída

```
2
0 5
4 0
```



Exemplo de entrada e saída

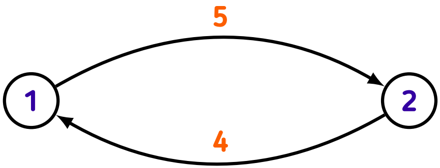
```
2
0 5
4 0
1 2
```



Exemplo de entrada e saída

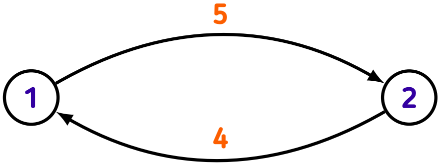
```
2
0 5
4 0
1 2
```

vértices excluídos



Exemplo de entrada e saída

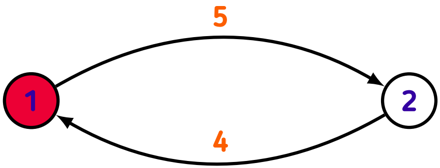
```
2
0 5
4 0
1 2
```



$$\sum_{u,v,u\neq v} d(1,u,v) = d(1,1,2) + d(1,2,1) = 5 + 4 = 9$$

Exemplo de entrada e saída

```
2
0 5
4 0
1 2
```



Exemplo de entrada e saída

```
2
0 5
4 0
1 2
```

```
2
```

Exemplo de entrada e saída

```
2
0 5
4 0
1 2
```

2

$$\sum_{u,v,u\neq v} d(2,u,v) = 0$$

Exemplo de entrada e saída

```
2
0 5
4 0
1 2
```

90

2

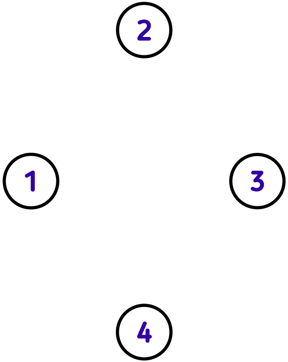
Exemplo de entrada e saída

Exemplo de entrada e saída

```
4
```

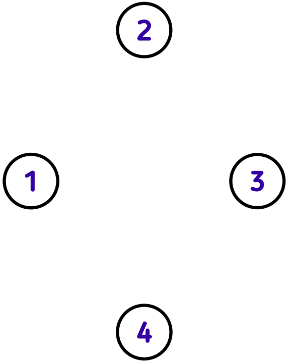
Exemplo de entrada e saída

4



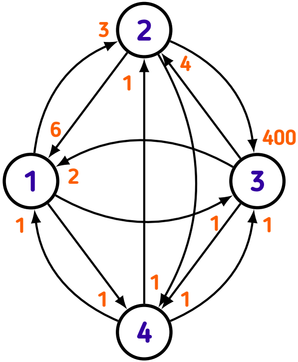
Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
```



Exemplo de entrada e saída

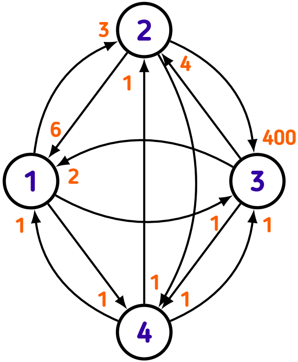
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
```



	1	2	3	4
1	0	3	1	1
2	6	0	400	1
3	2	4	0	1
4	1	1	1	0

Exemplo de entrada e saída

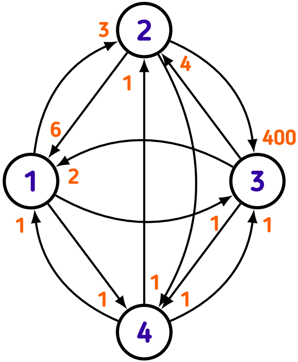
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	3	1	1
2	6	0	400	1
3	2	4	0	1
4	1	1	1	0

Exemplo de entrada e saída

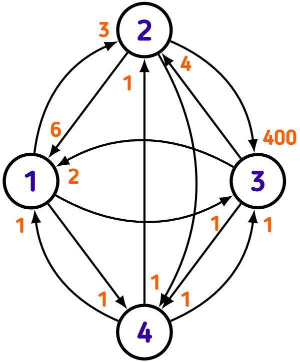
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	**2**	1	1
2	**2**	0	**2**	1
3	2	**2**	0	1
4	1	1	1	0

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

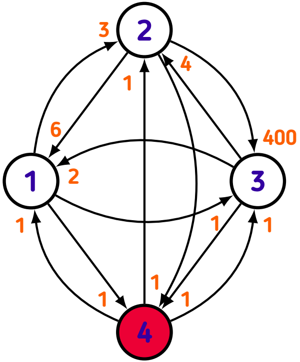


	1	2	3	4
1	0	2	1	1
2	2	0	2	1
3	2	2	0	1
4	1	1	1	0

$$\sum_{u,v,u\neq v} d(1,u,v) = 17$$

Exemplo de entrada e saída

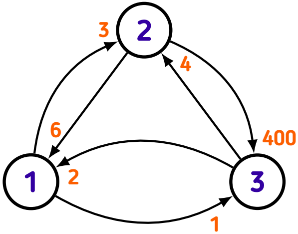
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	2	1	1
2	2	0	2	1
3	2	2	0	1
4	1	1	1	0

Exemplo de entrada e saída

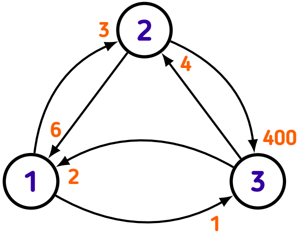
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	2	1	1
2	2	0	2	1
3	2	2	0	1
4	1	1	1	0

Exemplo de entrada e saída

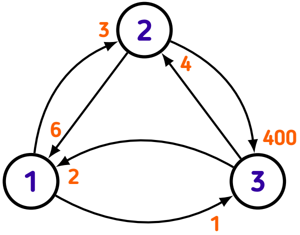
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	3	1	∞
2	6	0	7	∞
3	2	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

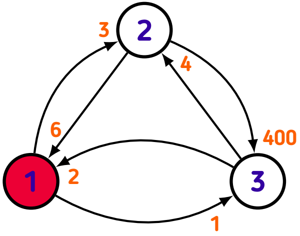


	1	2	3	4
1	0	**3**	1	∞
2	**6**	0	**7**	∞
3	2	**4**	0	∞
4	∞	∞	∞	∞

$$\sum_{u,v,u\neq v} d(2,u,v) = 23$$

Exemplo de entrada e saída

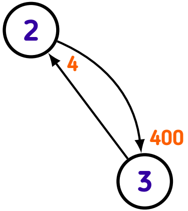
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	3	1	∞
2	6	0	7	∞
3	2	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

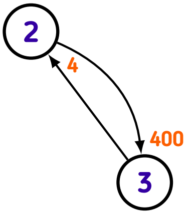
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	0	3	1	∞
2	6	0	7	∞
3	2	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

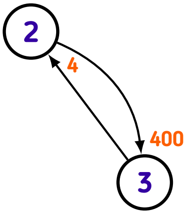
```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	∞	∞	∞	∞
2	∞	0	**400**	∞
3	∞	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

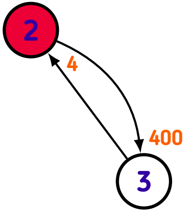


	1	2	3	4
1	∞	∞	∞	∞
2	∞	0	**400**	∞
3	∞	4	0	∞
4	∞	∞	∞	∞

$$\sum_{u,v,u\neq v} d(3,u,v) = 404$$

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```



	1	2	3	4
1	∞	∞	∞	∞
2	∞	0	400	∞
3	∞	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

3

	1	2	3	4
1	∞	∞	∞	∞
2	∞	0	400	∞
3	∞	4	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

3

	1	2	3	4
1	∞	∞	∞	∞
2	∞	∞	∞	∞
3	∞	∞	0	∞
4	∞	∞	∞	∞

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

3

	1	2	3	4
1	∞	∞	∞	∞
2	∞	∞	∞	∞
3	∞	∞	0	∞
4	∞	∞	∞	∞

$$\sum_{u,v,u\neq v} d(4,u,v) = 0$$

Exemplo de entrada e saída

```
4
0 3 1 1
6 0 400 1
2 4 0 1
1 1 1 0
4 1 2 3
```

17 23 404 0

3

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & \infty & \infty & \infty & \infty \\
2 & \infty & \infty & \infty & \infty \\
3 & \infty & \infty & 0 & \infty \\
4 & \infty & \infty & \infty & \infty
\end{array}
$$

$$\sum_{u,v,u\neq v} d(4,u,v) = 0$$

Solução

Solução

$\star$ **As arestas de um vértice podem ser removidas da matriz de adjacências em** $O(N)$

Solução

- $\star$ As arestas de um vértice podem ser removidas da matriz de adjacências em $O(N)$

- $\star$ Em cada etapa, os caminhos mínimos entre todos os pares podem ser computados com Floyd-Warshall em $O(N^3)$

Solução

- $\star$ As arestas de um vértice podem ser removidas da matriz de adjacências em $O(N)$

- $\star$ Em cada etapa, os caminhos mínimos entre todos os pares podem ser computados com Floyd-Warshall em $O(N^3)$

- $\star$ Como são N etapas, esta solução tem complexidade $O(N^4)$

Solução

- As arestas de um vértice podem ser removidas da matriz de adjacências em $O(N)$

- Em cada etapa, os caminhos mínimos entre todos os pares podem ser computados com Floyd-Warshall em $O(N^3)$

- Como são N etapas, esta solução tem complexidade $O(N^4)$

- **Veredito:** TLE!

Solução

Solução

- **Para reduzir a complexidade, é preciso compreender o funcionamento do algoritmo de Floyd-Warshall**

Solução

- ⋆ **Para reduzir a complexidade, é preciso compreender o funcionamento do algoritmo de Floyd-Warshall**

- ⋆ **A cada iteração, as distâncias são relaxadas usando o vértice k como intermediário**

Solução

$\star$ Para reduzir a complexidade, é preciso compreender o funcionamento do algoritmo de Floyd-Warshall

$\star$ A cada iteração, as distâncias são relaxadas usando o vértice k como intermediário

$\star$ Assim, basta começar com o grafo vazio e, a cada etapa, em ordem reversa, adicionar o vértice x_i e suas arestas a G, e relaxar as distâncias usando x_i

Solução

- $\star$ **Para reduzir a complexidade, é preciso compreender o funcionamento do algoritmo de Floyd-Warshall**

- $\star$ **A cada iteração, as distâncias são relaxadas usando o vértice k como intermediário**

- $\star$ **Assim, basta começar com o grafo vazio e, a cada etapa, em ordem reversa, adicionar o vértice x_i e suas arestas a G, e relaxar as distâncias usando x_i**

- $\star$ **Complexidade:** $O(N^3)$

```
vector<ll> solve(int N, vector<int>& xs)
{
    reverse(xs.begin(), xs.end());

    for (int i = 1; i <= N; ++i)
        for (int j = 1; j <= N; ++j)
            dist[i][j] = A[i][j];

    unordered_set<int> included;
    vector<ll> ans;

    for (auto x : xs)
    {
        included.insert(x);

         for (int u = 1; u <= N; ++u)
            for (int v = 1; v <= N; ++v)
                    dist[u][v] = min(dist[u][v], dist[u][x] + dist[x][v]);
```

```
        ll sum = 0;

        for (int u = 1; u <= N; ++u)
            for (int v = 1; v <= N; ++v)
                sum += (included.count(u) and included.count(v) ? dist[u][v] : 0);

         ans.emplace_back(sum);
    }

    reverse(ans.begin(), ans.end());

    return ans;
}
```