25-2 이니로
알고리즘 멘토링

멘토 - 김수성

파일 합치기 3 / 13975

백준 13975 / https://www.acmicpc.net/problem/13975

소설가인 김대전은 소설을 여러 장(chapter)으로 나누어 쓰는데, 각 장은 각각 다른 파일에 저장하곤 한다. 소설의 모든 장을 쓰고 나서는 각 장이 쓰여진 파일을 합쳐서 최종적으로 소설의 완성본이 들어있는 한 개의 파일을 만든다. 이 과정에서 두 개의 파일을 합쳐서 하나의 임시파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로는 하나의 파일로 합친다. 두 개의 파일을 합칠 때 필요한 비용(시간 등)이 두 파일 크기의 합이라고 가정할 때, 최종적인 한 개의 파일을 완성하는데 필요한 비용의 총 합을 계산하시오.

예를 들어, C1, C2, C3, C4가 네 개의 장을 수록하고 있는 파일이고, 파일 크기가 각각 40, 30, 30, 50 이라고 하자. 이 파일들을 합치는 과정에서, 먼저 C2와 C3를 합쳐서 임시파일 X1을 만든다. 이때 비용 60이 필요하다. 그 다음으로 C1과 X1을 합쳐 임시파일 X2를 만들면 비용 100이 필요하다. 최종적으로 X2와 C4를 합쳐 최종파일을 만들면 비용 150이 필요하다. 따라서, 최종의 한 파일을 만드는데 필요한 비용의 합은 60+100+150=310 이다. 다른 방법으로 파일을 합치면 비용을 줄일 수 있다. 먼저 C1과 C2를 합쳐 임시파일 Y1을 만들고, C3와 C4를 합쳐 임시파일 Y2를 만들고, 최종적으로 Y1과 Y2를 합쳐 최종파일을 만들 수 있다. 이때 필요한 총 비용은 70+80+150=300 이다.

소설의 각 장들이 수록되어 있는 파일의 크기가 주어졌을 때, 이 파일들을 하나의 파일로 합칠 때 필요한 최소비용을 계산하는 프로그램을 작성하시오.

입력

프로그램은 표준 입력에서 입력 데이터를 받는다. 프로그램의 입력은 T개의 테스트 데이터로 이루어져 있는데, T는 입력의 맨 첫 줄에 주어진다.각 테스트 데이터는 두 개의 행으로 주어지는데, 첫 행에는 소설을 구성하는 장의 수를 나타내는 양의 정수 K (3 ≤ K ≤ 1,000,000)가 주어진다. 두 번째 행에는 1장부터 K장까지 수록한 파일의 크기를 나타내는 양의 정수 K개가 주어진다. 파일의 크기는 10,000을 초과하지 않는다.

출력

프로그램은 표준 출력에 출력한다. 각 테스트 데이터마다 정확히 한 행에 출력하는데, 모든 장을 합치는데 필요한 최소비용을 출력한다.

파일 합치기 3 / 13975

예제 입력 1 복사

```
2
4
40 30 30 50
15
1 21 3 4 5 35 5 4 3 5 98 21 14 17 32
```

예제 출력 1 복사

```
300
826
```

40 30 30 50

30 + 30 -> 60

40 50 60

40 + 50 -> 90

60 90

60 + 90 -> 150

60 + 90 + 150 = 300

파일 합치기 3 / 13975

40 30 30 50
30 + 30 -> 60

60 90
60 + 90 -> 150

40 50 60
40 + 50 -> 90

60 + 90 + 150 = 300

항상 제일 작은 것부터 합치는 것이 이득일 것 같음

-> 증명

파일 합치기 3 / 13975

증명
비용이 A <= B <= C인 파일 3개가 남았다고 할 때

A + B + A + B + C -> 2 * A + 2 * B + C
A + C + A + C + B -> 2 * A + B + 2 * C
B + C + B + C + C -> A + 2 * B + 2 * C

파일 합치기 3 / 13975

증명
비용이 A <= B <= C인 파일 3개가 있다고 할 때

2 * A + 2 * B + C -> A + B
2 * A + B + 2 * C -> A + C
A + 2 * B + 2 * C -> B + C

항상 A + B가 최소이므로 A B C 순으로 합치는 것이 최적이다

파일 합치기 3 / 13975

C++

```cpp
#include <iostream>
#include <queue>
using namespace std;
using ll = long long;

const ll MAX = 1010101;
ll n, t, a[MAX];
priority_queue <ll, vector<ll>, greater<ll>> pq;
```

```cpp
void solve(){
    cin >> n; ll result = 0;
    for(int i = 1;i <= n;i++) cin >> a[i], pq.push(a[i]);
    while(pq.size() > 1){
        ll fi = pq.top(); pq.pop();
        ll se = pq.top(); pq.pop();
        result += fi + se;
        pq.push(fi + se);
    }

    pq.pop();
    cout << result << "\n";
}

int main(){
    cin >> t;
    while(t--) solve();
    return 0;
}
```

파일 합치기 3 / 13975

Python

```python
t = int(input())
for _ in range(t):
    solve()
```

```python
def solve():
    n = int(input())
    a = list(map(int, input().split()))

    pq = []
    for i in range(n):
        heapq.heappush(pq, a[i])

    result = 0
    while len(pq) > 1:
        fi = heapq.heappop(pq)
        se = heapq.heappop(pq)

        result += fi + se
        heapq.heappush(pq, fi + se)

    print(result)
```

질문?

구두 수선공 / 14908

백준 14908 / https://www.acmicpc.net/problem/14908

문제

지금 구두 수선공에게는 손님으로부터 주문 받고 제작해야 할 작업이 N개 쌓여있다. 구두 수선공은 하루에 한 작업만 수행할 수 있고, i번째 작업을 완료하는 데 T_i일이 걸린다. 이때 T_i는 정수이고 $1 \le T_i \le 1000$이다.

i번째 작업을 시작하기 전에 하루가 지연될 때마다 구두 수선공은 보상금 S_i센트를 지불해야 한다. 이때 S_i는 정수이고 $1 \le S_i \le 10000$이다. 구두 수선공을 돕기 위해 최저 보상금을 지불하는 작업 순서를 정해야 한다.

하루에 2개 이상의 작업을 동시에 수행할 수 없다. 작업 i를 수행하고 있는 경우, 작업 i를 마칠 때 까지 작업 i 외의 다른 작업을 수행할 수 없다.

입력

$1 \le N \le 1000$ 범위의 정수 N이 첫 번째 줄에 주어진다. 다음 N개 줄에 걸쳐서 첫 번째 열에는 T_1 ... T_N이 입력되며, 두 번째 열에는 S_1 ... S_N이 주어진다.

출력

최소 보상금을 지불하는 작업 순서를 출력해야 한다. 모든 작업은 입력에서의 번호(1~N)로 표시해야 한다. 모든 정수는 한 줄로 표시해야 하며, 각 작업은 공백 문자로 구분한다. 여러 가지 해답이 나올 수 있다면 오름차순 정렬에 의해 가장 첫 번째 해답을 출력한다.

구두 수선공 / 14908

예제 입력 1 복사

```
4
3 4
1 1000
2 2
5 5
```

예제 출력 1 복사

```
2 1 3 4
```

2 -> 1 -> 3 -> 4

1000 * 0 + 4 * 1 + 2 * 4 + 6 * 5 -> 0 + 4 + 8 + 30

구두 수선공 / 14908

t1, s1인 작업과 t2, s2인 작업 2개가 있고
이전까지 들은 시간이 T, 비용이 C라고 하자

(t1, s1) -> (t2, s2)
C + T * s1 + (T + t1) * s2 -> C + T * s1 + T * s2 + t1 * s2

(t2, s2) -> (t1, s1)
C + T * s2 + (T + t2) * s1 -> C + T * s2 + T * s1 + t2 * s1

구두 수선공 / 14908

(t1, s1) -> (t2, s2)
C + T * s1 + (T + t1) * s2 -> C + T * s1 + T * s2 + t1 * s2

(t2, s2) -> (t1, s1)
C + T * s2 + (T + t2) * s1 -> C + T * s2 + T * s1 + t2 * s1

두 항에 C + T * s1 + T * s2를 빼고 양변에 s1 * s2를 나누면
t1 / s1, t2 / s2이 남는다
-> ti / si 가 작은 순으로 정렬하는 것이 최적

구두 수선공 / 14908

C++

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
using ll = long long;
using ld = long double;

const ll MAX = 1010;
ll n, t[MAX], s[MAX];
pair <ld, ll> a[MAX];
```

```cpp
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> t[i] >> s[i];
        a[i].first = (ld)t[i] / (ld)s[i];
        a[i].second = i;
    }
    sort(a + 1, a + n + 1);

    for(int i = 1;i <= n;i++) cout << a[i].second << " ";
    return 0;
}
```

구두 수선공 / 14908

Python

```python
import sys
input = sys.stdin.readline

n = int(input())
a = []
for i in range(1, n + 1):
    t, s = map(int, input().split())
    a.append((t / s, i))

a.sort()

for _, idx in a:
    print(idx, end=" ")
```

질문?

6주차 - MST

MST

스패닝 트리

어떤 그래프에서 간선을 부분적으로 선택해 만들 수 있는
그래프의 정점 개수와 같은 정점 개수를 가지는 트리

최소 스패닝 트리(MST)

가장 가중치가 작은 스패닝 트리

MST

MST 구성 알고리즘

프림 알고리즘 -> 그리디 + 우선 순위 큐

크루스칼 알고리즘 -> 그리디 + 유니온 파인드

솔린 알고리즘 -> 그리디 + 유니온 파인드

MST

트리의 성질

N개의 정점을 가지는 트리는 N - 1개의 간선을 가짐
사이클이 존재하지 않음

-> 사이클을 만들지 않는 선에서 가장 비용이 적은
N - 1개의 간선을 선택

MST

프림 알고리즘

1. 시작 정점으로 아무 정점 선택 및 방문 처리
   시작 정점과 인접한 간선 우선순위 큐에 삽입

2. 우선순위 큐 top 방문

2 – 1. top이 방문 처리 되어 있으면 이미 연결된 정점
   -> 건너 뜀

2 – 2. 아니면 정답에 추가 및 방문 처리
   top과 인접한 간선 우선순위 큐에 삽입

MST

C++

```cpp
#include <iostream>
#include <queue>
using namespace std;
using ll = long long;
using pll = pair<ll, ll>;

const ll MAX = 10101;
ll n, m, result;
bool v[MAX];
vector <pll> adj[MAX];
```

```cpp
class node{
public:
    ll s, e, c;
    // pq를 위한 > 연산자 오버로딩
    // 혹은 tuple 사용해서 간선이 앞으로 오게 하면 됨
    bool operator > (const node& ot) const{
        return c > ot.c;
    }
};
priority_queue <node, vector<node>, greater<node>> pq;
```

```cpp
int main(){
    cin >> n >> m;
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        adj[s].push_back({e, c});
        adj[e].push_back({s, c});
    }

    // 시작 정점과 연결된 간선을 pq에 삽입
    for(auto& [e, c] : adj[1]) pq.push({1, e, c});
    v[1] = 1; // 시작 정점 방문 처리

    while(!pq.empty()){
        auto[s, e, c] = pq.top(); pq.pop();
        // 다음 정점을 방문 했으면 -> 같은 집합이면
        // 건너 뜀
        if(v[e] == 1) continue;

        // 정답에 비용 추가 및 방문 처리
        result += c; v[e] = 1;

        for(auto& [nxt, co] : adj[e]){
            // 이미 방문한 정점이면 건너 뜀
            if(v[nxt]) continue;

            // pq에 간선 삽입
            pq.push({e, nxt, co});
        }
    }

    cout << result;
    return 0;
}
```

MST

Python

```python
import sys, heapq
input = sys.stdin.readline

MAX = 10101
n, m = map(int, input().split())
result = 0
v = [0] * (n + 1)
adj = [[] for _ in range(n + 1)]
pq = []

for i in range(m):
    s, e, c = map(int, input().split())
    adj[s].append((e, c))
    adj[e].append((s, c))
```

```python
# 시작 정점과 연결된 간선을 pq에 삽입
for e, c in adj[1]:
    heapq.heappush(pq, (c, 1, e))
v[1] = 1  # 시작 정점 방문 처리

while pq:
    c, s, e = heapq.heappop(pq)
    # 다음 정점을 방문 했으면 -> 같은 집합이면
    # 건너 뜀
    if v[e] == 1:
        continue

    # 정답에 비용 추가 및 방문 처리
    result += c
    v[e] = 1

    for nxt, co in adj[e]:
        # 이미 방문한 정점이면 건너 뜀
        if v[nxt]:
            continue

        # pq에 간선 삽입
        heapq.heappush(pq, (co, e, nxt))

print(result)
```

최소 스패닝 트리 / 1197

백준 1197 / https://www.acmicpc.net/problem/ 1197

문제

그래프가 주어졌을 때, 그 그래프의 최소 스패닝 트리를 구하는 프로그램을 작성하시오.

최소 스패닝 트리는, 주어진 그래프의 모든 정점들을 연결하는 부분 그래프 중에서 그 가중치의 합이 최소인 트리를 말한다.

입력

첫째 줄에 정점의 개수 V(1 ≤ V ≤ 10,000)와 간선의 개수 E(1 ≤ E ≤ 100,000)가 주어진다. 다음 E개의 줄에는 각 간선에 대한 정보를 나타내는 세 정수 A, B, C가 주어진다. 이는 A번 정점과 B번 정점이 가중치 C인 간선으로 연결되어 있다는 의미이다. C는 음수일 수도 있으며, 절댓값이 1,000,000을 넘지 않는다.

그래프의 정점은 1번부터 V번까지 번호가 매겨져 있고, 임의의 두 정점 사이에 경로가 있다. 최소 스패닝 트리의 가중치가 -2,147,483,648보다 크거나 같고, 2,147,483,647보다 작거나 같은 데이터만 입력으로 주어진다.

출력

첫째 줄에 최소 스패닝 트리의 가중치를 출력한다.

질문?

전기가 부족해 / 10423

백준 10423 / https://www.acmicpc.net/problem/ 10423

있고, 따라서 추가적으로 드는 비용은 케이블을 설치할 때 드는 비용이 전부이다. 이 프로젝트의 문제는 케이블을 설치할 때 드는 비용이 굉장히 크므로 이를 최소화해서 설치하여 모든 도시에 전기를 공급하는 것이다. 여러분은 N개의 도시가 있고 M개의 두 도시를 연결하는 케이블의 정보와 K개의 YNY발전소가 설치된 도시가 주어지면 케이블 설치 비용을 최소로 사용하여 모든 도시에 전기가 공급할 수 있도록 해결해야 한다. 중요한 점은 어느 한 도시가 두 개의 발전소에서 전기를 공급받으면 낭비가 되므로 케이블이 연결되어있는 도시에는 발전소가 반드시 하나만 존재해야 한다. 아래 Figure 1를 보자. 9개의 도시와 3 개의 YNY발전소(A,B,I)가 있고, 각각의 도시들을 연결할 때 드는 비용이 주어진다.

입력

첫째 줄에는 도시의 개수 N(1 ≤ N ≤ 1,000)과 설치 가능한 케이블의 수 M(1 ≤ M ≤ 100,000)개, 발전소의 개수 K(1 ≤ K ≤ N)개가 주어진다. 둘째 줄에는 발전소가 설치된 도시의 번호가 주어진다. 셋째 줄부터 M개의 두 도시를 연결하는 케이블의 정보가 u, v, w로 주어진다. 이는 u도시와 v도시를 연결하는 케이블을 설치할 때 w의 비용이 드는 것을 의미한다. w는 10,000보다 작거나 같은 양의 정수이다.

출력

모든 도시에 전기를 공급할 수 있도록 케이블을 설치하는 데 드는 최소비용을 출력한다.

전기가 부족해 / 10423

예제 입력 1 복사

```
9 14 3
1 2 9
1 3 3
1 4 8
2 4 10
3 4 11
3 5 6
4 5 4
4 6 10
5 6 5
5 7 4
6 7 7
6 8 4
7 8 5
7 9 2
8 9 5
```

예제 출력 1 복사

```
22
```

전기가 부족해 / 10423

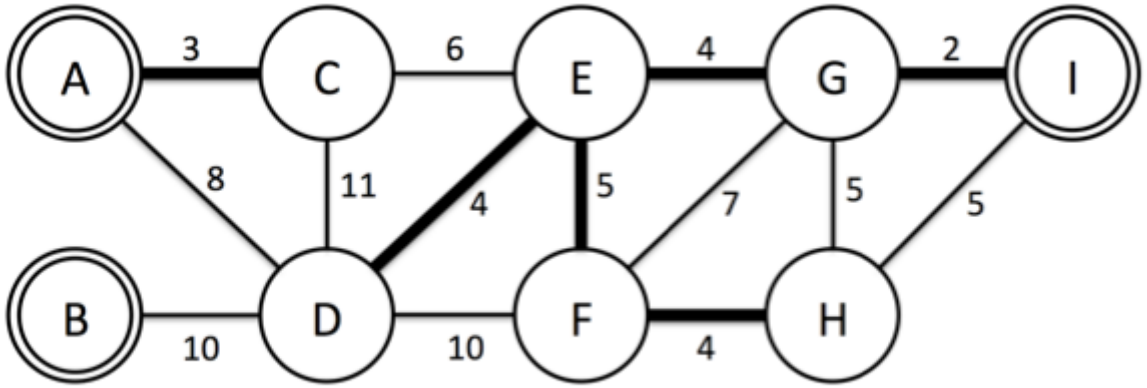


Figure 2

모든 정점이 K개의 정점 중 1개와 연결 되어 있게 할 때 사용하는 최소 간선의 비용을 구해야 함

전기가 부족해 / 10423

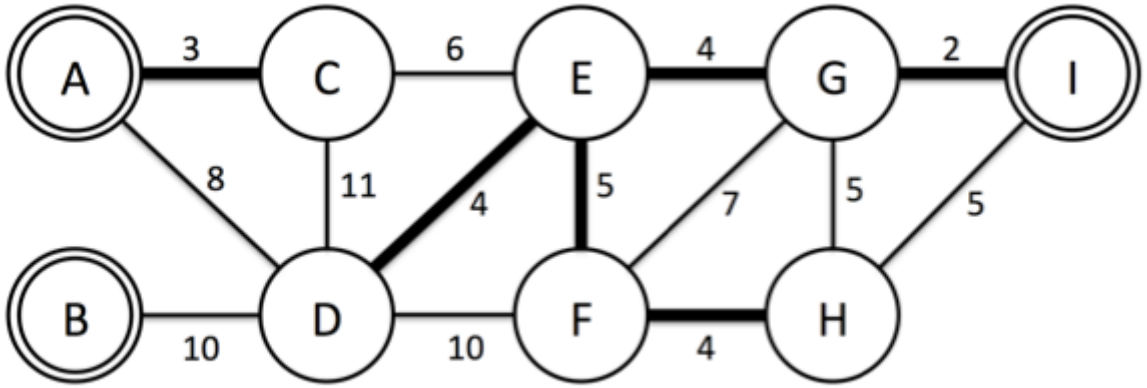


Figure 2

모든 정점이 K개의 정점 중 1개와 연결 되어 있게 할 때 사용하는 최소 간선의 비용을 구해야 함

-> K개의 정점을 하나로 보고 MST 구성

전기가 부족해 / 10423

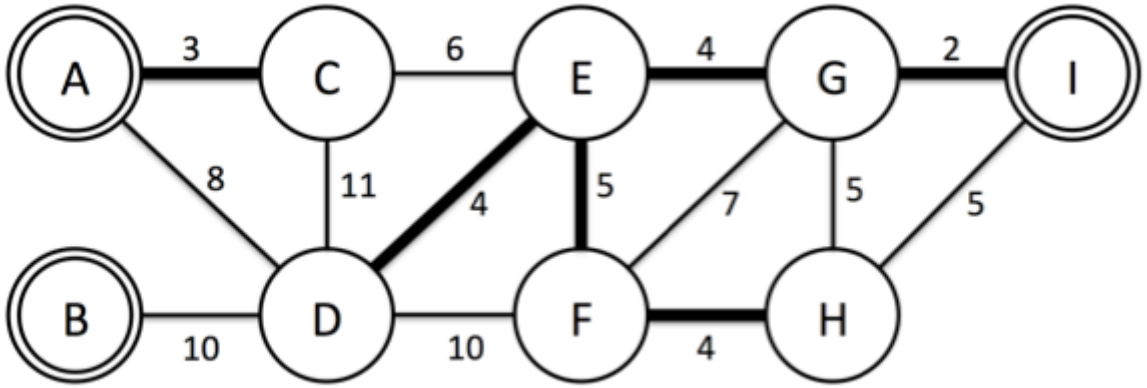


Figure 2

K개의 정점을 하나로 보고 MST 구성

정점 0을 새로 만들어 K개에 정점에 대해 비용 0인 간선으로 연결해서 MST 수행

전기가 부족해 / 10423

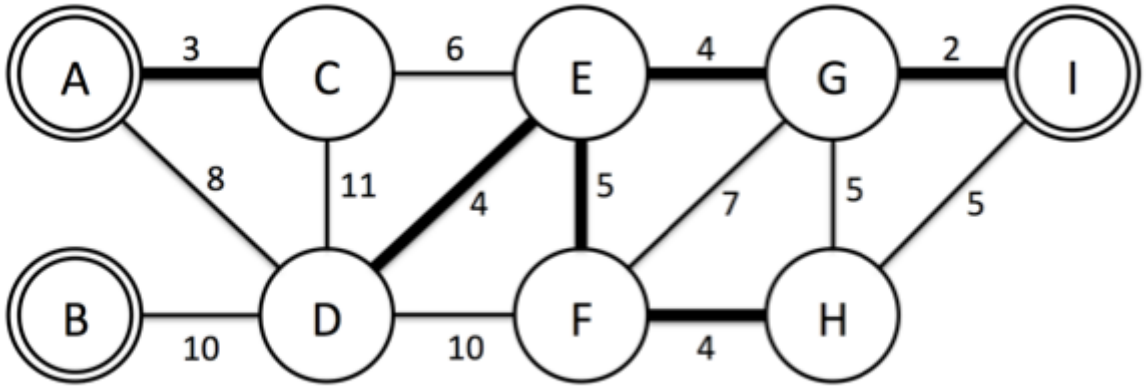


Figure 2

K개의 정점을 하나로 보고 MST 구성

시작 정점에 K개의 정점을 모두 넣고 MST 수행

전기가 부족해 / 10423

C++

```cpp
#include <iostream>
#include <queue>
using namespace std;
using ll = long long;
using pll = pair<ll, ll>;

const ll MAX = 1010;
ll n, m, k, result, a[MAX];
bool v[MAX];
vector <pll> adj[MAX];

class node{
public:
    ll s, e, c;
    bool operator > (const node& ot) const{
        return c > ot.c;
    }
};
priority_queue <node, vector<node>, greater<node>> pq;
```

```cpp
int main(){
    cin >> n >> m >> k;
    for(int i = 1;i <= k;i++) cin >> a[i];
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        adj[s].push_back({e, c});
        adj[e].push_back({s, c});
    }

    for(int i = 1;i <= k;i++){
        v[a[i]] = 1;
        for(auto& [e, c] : adj[a[i]]) pq.push({a[i], e, c});
    }

    while(!pq.empty()){
        auto[s, e, c] = pq.top(); pq.pop();
        if(v[e] == 1) continue;
        result += c; v[e] = 1;

        for(auto& [nxt, co] : adj[e]){
            if(v[nxt]) continue;
            pq.push({e, nxt, co});
        }
    }

    cout << result;
    return 0;
}
```

전기가 부족해 / 10423

Python

```python
import sys, heapq
input = sys.stdin.readline

MAX = 10101
n, m, k = map(int, input().split())
result = 0
v = [0] * (n + 1)
adj = [[] for _ in range(n + 1)]
pq = []

a = list(map(int, input().split()))
for i in range(m):
    s, e, c = map(int, input().split())
    adj[s].append((e, c))
    adj[e].append((s, c))
```

```python
for i in a:
    v[i] = 1
    for e, c in adj[i]:
        heapq.heappush(pq, (c, i, e))

while pq:
    c, s, e = heapq.heappop(pq)
    if v[e] == 1:
        continue
    result += c
    v[e] = 1

    for nxt, co in adj[e]:
        if v[nxt]:
            continue
        heapq.heappush(pq, (co, e, nxt))

print(result)
```

질문?

물대기 / 1368

백준 1368 / https://www.acmicpc.net/problem/1368

문제

선주는 자신이 운영하는 N개의 논에 물을 대려고 한다. 물을 대는 방법은 두 가지가 있는데 하나는 직접 논에 우물을 파는 것이고 다른 하나는 이미 물을 대고 있는 다른 논으로부터 물을 끌어오는 법이다.

각각의 논에 대해 우물을 파는 비용과 논들 사이에 물을 끌어오는 비용들이 주어졌을 때 최소의 비용으로 모든 논에 물을 대는 것이 문제이다.

입력

첫 줄에는 논의 수 N(1 ≤ N ≤ 300)이 주어진다. 다음 N개의 줄에는 i번째 논에 우물을 팔 때 드는 비용 Wi(1 ≤ Wi ≤ 100,000)가 순서대로 들어온다. 다음 N개의 줄에 대해서는 각 줄에 N개의 수가 들어오는데 이는 i번째 논과 j번째 논을 연결하는데 드는 비용 Pi,j(1 ≤ Pi,j ≤ 100,000, Pi,j = Pj,i, Pi,i = 0)를 의미한다.

출력

첫 줄에 모든 논에 물을 대는데 필요한 최소비용을 출력한다.

물대기 / 1368

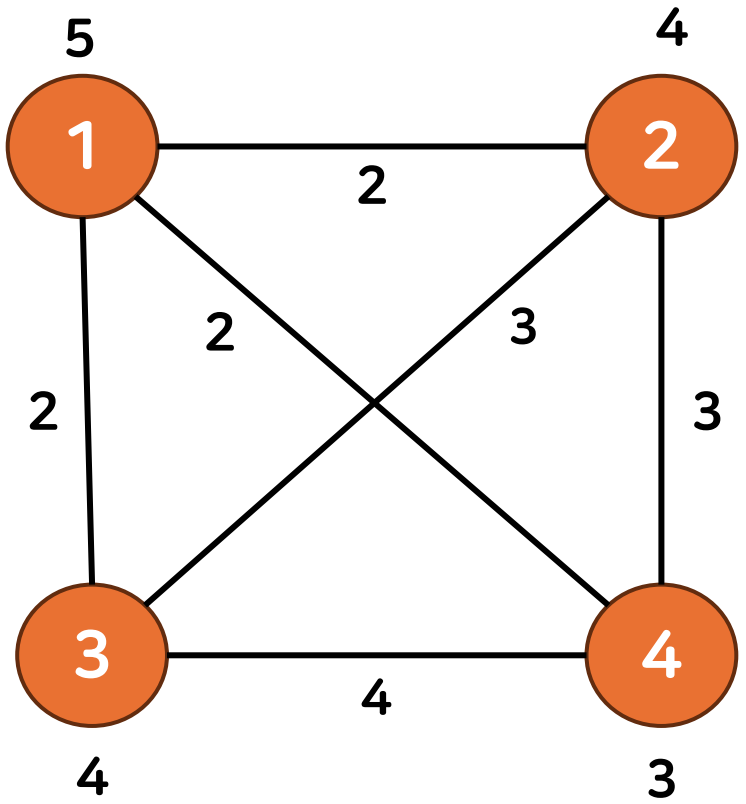
예제 입력 1 복사

```
4
5
4
4
3
0 2 2 2
2 0 3 3
2 3 0 4
2 3 4 0
```

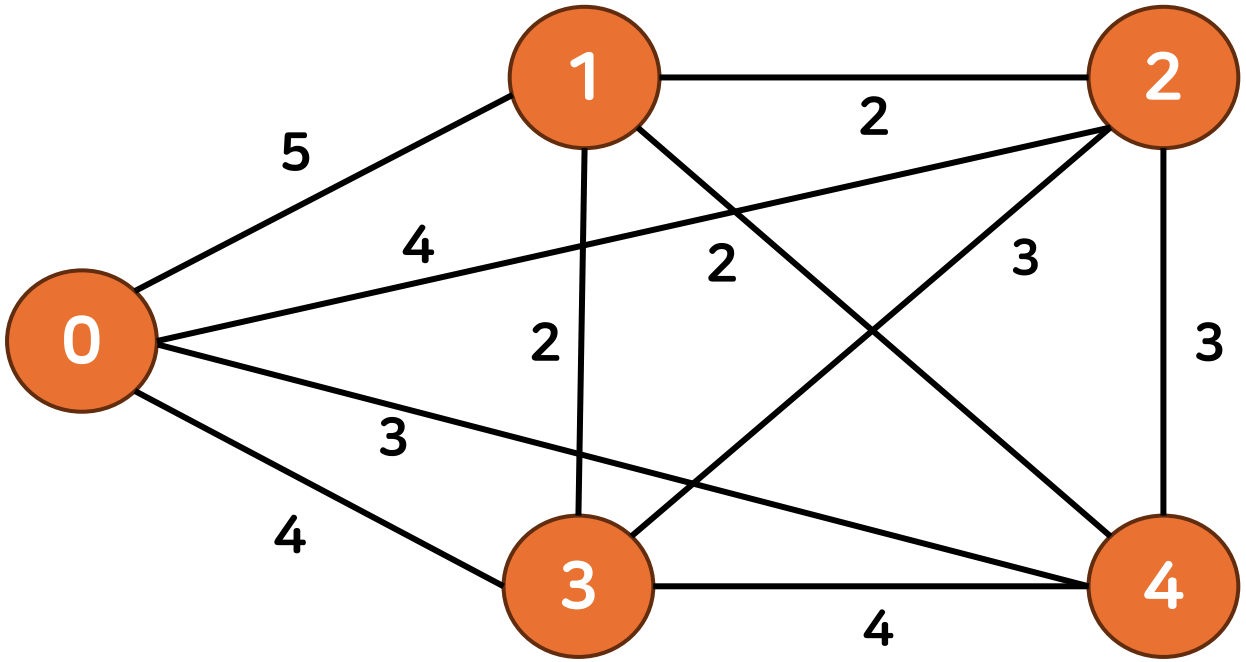
예제 출력 1 복사

```
9
```

물대기 / 1368



물대기 / 1368



물대기 / 1368

C++

```cpp
#include <iostream>
#include <queue>
using namespace std;
using ll = long long;
using pll = pair<ll, ll>;

const ll MAX = 303;
ll n, c, result;
bool v[MAX];
vector <pll> adj[MAX];

class node{
public:
    ll s, e, c;
    bool operator > (const node& ot) const{
        return c > ot.c;
    }
};
priority_queue <node, vector<node>, greater<node>> pq;
```

```cpp
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++) cin >> c, adj[0].push_back({i, c});
    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= n;j++){
            cin >> c; if(!c) continue;
            adj[i].push_back({j, c});
        }
    }

    v[0] = 1;
    for(auto& [e, c] : adj[0]) pq.push({0, e, c});

    while(!pq.empty()){
        auto[s, e, c] = pq.top(); pq.pop();
        if(v[e] == 1) continue;
        result += c; v[e] = 1;

        for(auto& [nxt, co] : adj[e]){
            if(v[nxt]) continue;
            pq.push({e, nxt, co});
        }
    }

    cout << result;
    return 0;
}
```

물대기 / 1368

Python

```python
import sys, heapq
input = sys.stdin.readline

MAX = 303
n = int(input())
result = 0
v = [0] * (n + 1)
adj = [[] for _ in range(n + 1)]
pq = []

for i in range(1, n + 1):
    c = int(input())
    adj[0].append((i, c))

for i in range(1, n + 1):
    c = list(map(int, input().split()))
    for j in range(n):
        if c == 0:
            continue
        adj[i].append((j + 1, c[j]))
```

```python
v[0] = 1
for e, c in adj[0]:
    heapq.heappush(pq, (c, 0, e))

while pq:
    c, s, e = heapq.heappop(pq)
    if v[e] == 1:
        continue
    result += c
    v[e] = 1

    for nxt, co in adj[e]:
        if v[nxt]:
            continue
        heapq.heappush(pq, (co, e, nxt))

print(result)
```

질문?

과제

최소 스패닝 트리 - https://www.acmicpc.net/problem/1197

전기가 부족해 - https://www.acmicpc.net/problem/10423

Traveling SCCC President -
https://www.acmicpc.net/problem/28119

물대기 - https://www.acmicpc.net/problem/1368

고생하셨습니다