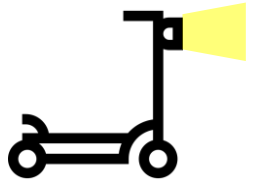
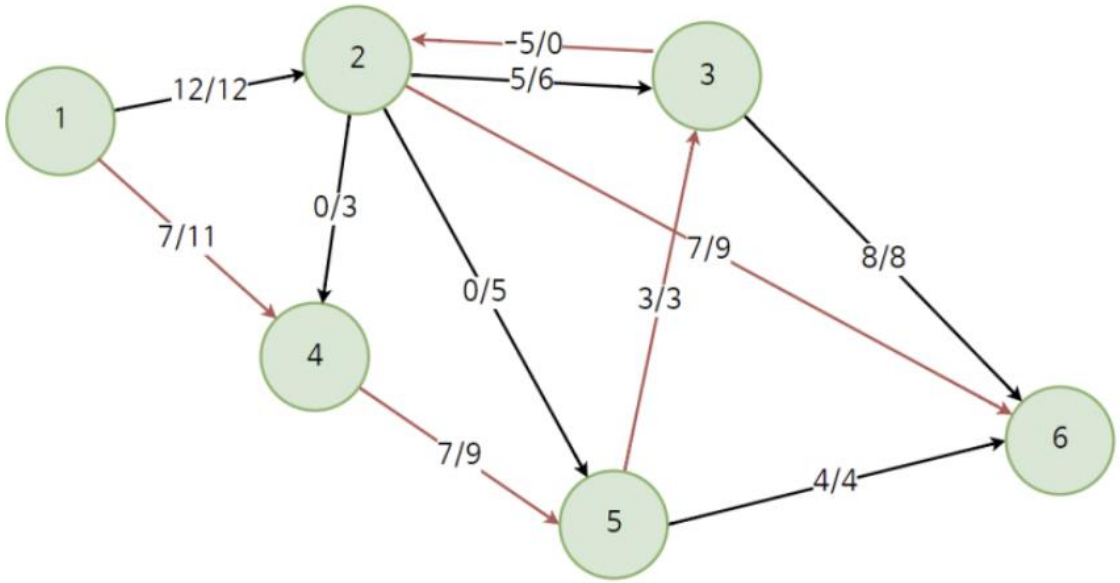


스마트 모빌리티를 위한 이동기기 재배치 메카니즘

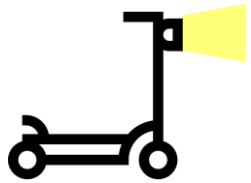
Grazioso 팀
박지현, 장승아, 윤소원

네트워크 플로우



- 네트워크 플로우 -> 얼마나 흘러갈 수 있는가?
  가중치 있음. 여러 개의 연결 가능.
- 시작노드와 끝노드가 있음.
- BFS 탐색시 유량이 결정되면
  순방향 링크의 값 감소, 역방향 링크의 값 증가

네트워크 플로우 문제



https://www.acmicpc.net/problem/6086

6086번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 검색

최대 유량 성공 다국어

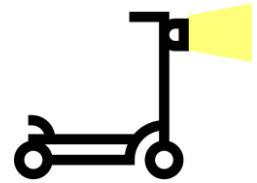
☆ 한국어

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	10766	3077	1570	25.899%

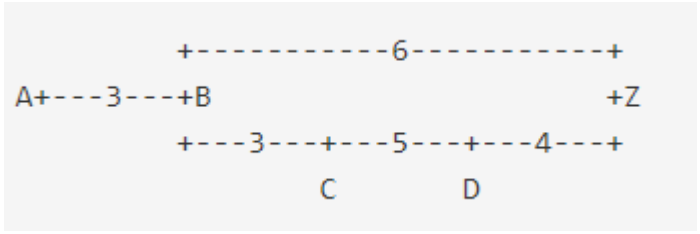
문제

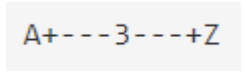
농사꾼 존은 소들이 충분한 물을 마시길 원했다. 그래서 농장에서 우물에서 외양간을 잇는 N개의 배수관의 지도를 만들기로 했다. 존은 아주 다양한 크기의 배수관들이 완전히 우연한 방법으로 연결돼있음을 알았다. 존은 파이프를 통과하는 유량을 계산하고 싶다.

네트워크 플로우 문제



https://www.acmicpc.net/problem/6086





각 파이프의 용량이 있고 A와 Z간 흐를 수 있는 용량의 최대치를 구하라.

예제 입력 1 복사

```
5
A B 3
B C 3
C D 5
D Z 4
B Z 6
```

예제 출력 1 복사

```
3
```

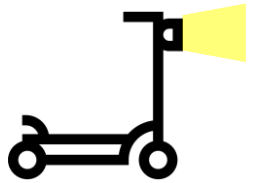
- 입력

파이프의 개수
파이프에 연결된 노드들(2개), 파이프의 용량

- 출력

파이프의 최대 유량

네트워크 플로우 코드



https://www.acmicpc.net/problem/6086

```
public class Aaa {
        static ArrayList<HashMap<Integer, Integer>> G;
        static int T, S=0, D;
        static int M, N;
        static int[] from;
```

S:시작점 D:끝점
T:노드 개수

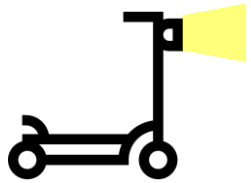
```
static void AddEdge(int f,int t, int w) {
        int val = G.get(f).getOrDefault(t, 0);
        G.get(f).put(t, val+w);
        val=G.get(t).getOrDefault(f, 0);
        G.get(t).put(f, val+w);
}
```

노드 간 연결.
이때, 같은 링크가 반복해서 나올 수 있다.
양방향으로 추가.

```
static int GetNode(String s) {
        if(s.charAt(0)<'a') return ((int)(s.charAt(0)-'A'));
        else return ((int)(s.charAt(0)-'a'))+26;
}
```

노드가 영문자로 들어오기 때문에 숫자로 바꿔줌.

네트워크 플로우 코드



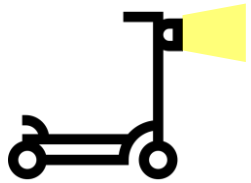
https://www.acmicpc.net/problem/6086

```
static boolean bfs() {
        boolean[] V = new boolean[T];
        for (int i=0; i<T; i++)
                V[i] = false;
        int[] que = new int[T];
        int front =0, rear = 0; from[S] = -1;
        que[rear++] = S; V[S]=true;

        while (front != rear) {
        int u = que[front++];
        for(Integer k : G.get(u).keySet()){
                if (V[k]||G.get(u).get(k) == 0) continue;
                que[rear++] = k;
                V[k]= true; from[k] = u;
                if (k == D) return (true);
                }
        }
        return false;
}
```

연결하는 과정 bfs
D(Z,끝점)에 연결되면 성공적인 탐색
그 외 실패했으므로 false

네트워크 플로우 코드

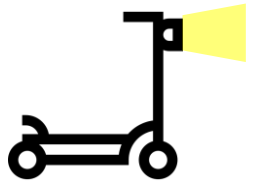


https://www.acmicpc.net/problem/6086

```
public static void main(String[] args)throws IOException {
        Scanner sc = new Scanner(System.in);
        N=sc.nextInt();
        T=52;S=0;D=25;
        G=new ArrayList<HashMap<Integer,Integer>>();
        for(int i=0;i<T;i++) G.add(new HashMap<Integer, Integer>());
        from  = new int[T];
        for(int i=0;i<N;i++) {
                int d1=GetNode(sc.next());
                int d2=GetNode(sc.next());
                int d3=sc.nextInt();
                AddEdge(d1, d2, d3);
        }sc.close();
```

입력 받아 그래프 생성

네트워크 플로우 코드



https://www.acmicpc.net/problem/6086

```
int n, f, min=10000000,sum=0;
while(bfs()) {
        n=D;f=from[D];
        while(n!=S) {
                min = Math.min(min, G.get(f).get(n));
                n=f;f=from[n];
        }
        n=D;f=from[D];
        while(n!=S) {
                G.get(f).put(n, G.get(f).get(n)-min);
                G.get(n).put(f, G.get(n).get(f)+min);
                n=f;f=from[n];
        }sum+=min;
}
System.out.println(sum);
```

되돌아오며 가장 적은 링크값 찾음.

다시 되돌아오며 링크 비용 변경.
순방향 링크의 값 감소, 역방향 링크의 값 증가.