GIST ALGORITHM MASTERS

2024 Contest

Official Solutions Editorial

GIST 알고리즘 마스터즈 준비위원회

	문제	의도한 난이도	출제자
A	GIST 찍기	Easy	송혜근 @지스트 invrtd_h
B	커플링 매치	Easy	박종현 @전남대 belline0124
C	FOCUS	Normal	송혜근 @지스트 invrtd_h
D	이진수 격자	Normal	송혜근 @지스트 invrtd_h
E	ChatGPT 만들기	Hard	송혜근 @지스트 invrtd_h
F	유리병 속 무한히 터지는 기포	Hard	송혜근 @지스트 invrtd_h, 이윤수 @전남대 lys9546
G	돈 복사	Hard	고민규 @전남대 jjkmk1013, 송혜근 @지스트 invrtd_h
H	기초마법학	Challenging	송혜근 @지스트 invrtd_h
I	DevNight 운영	Challenging	송혜근 @지스트 invrtd_h

참고

» 문제별 최초 해결 통계에서는 부정행위를 한 것이 확실한 참가자가 제외되었습니다.

A. GIST 찍기

`#implementation`, `#string`

출제진 의도 — **Easy**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 128회, 정답 51건
    정답률 39.844%
  - 최초 해결: 김근성 @전남대, 2분
- 오픈 콘테스트
  - 제출 115회, 정답 103건
    정답률 90.435%
  - 최초 해결: `riroan`, 1분

A. GIST 찍기

» 문제에서 주어진 공식을 그대로 구현하면 되는 것 같습니다. 하나의 난점이 있다면 대부분의 프로그래밍 언어는 0-based 인덱스를 쓰지만 행렬은 1-based 인덱스를 사용한다는 것입니다. 따라서 공식을 적절히 수정해야 합니다.

» 다음 공식을 그대로 구현하면 됩니다.

$$Y = (y_{ij}), y_{ij} = x_{\lfloor \frac{i}{K} \rfloor \lfloor \frac{j}{K} \rfloor}$$

» 또는 자료 구조 내에서 1칸의 padding을 두고 1-based 인덱스로 구현한 뒤 출력할 때만 padding을 자르는 방법으로도 구현 가능합니다.

B. 커플링 매치

`#implementation`, `#hash-map`

출제진 의도 — **Easy**

- 출제자: 박종현 @전남대 `belline0124`
- 온사이트 대회
  - 제출 165회, 정답 41건
    정답률 24.848%
  - 최초 해결: 김근성 @전남대, 7분
- 오픈 콘테스트
  - 제출 111회, 정답 77건
    정답률 70.27%
  - 최초 해결: `riroan`, 2분

» 입력으로 주어지는 각 반지 특징에 대해, 정확히 두 번 등장한 반지 특징을 찾는 것이 목표입니다.

» 반지 특징을 키로 갖고, 반지 특징의 등장 횟수와 해당 반지를 착용하고 있는 사람의 이름 집합을 값으로 갖는 키-값 쌍을 생성하고 활용하는 것이 정해입니다.

» 키-값 쌍의 개념과 이를 다루는 방법의 난이도가 낮으므로, 이 문제에서 키-값 쌍을 다루지 않는 풀이가 더 어렵습니다. 따라서 키-값 쌍을 사용하지 않고 단순히 반복문으로 처리하는 것을 별해로 두고 따로 막지 않았습니다.

C. FOCUS

`#implementation`, `#bitmasking`

출제진 의도 — **Normal**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 113회, 정답 36건
    정답률 31.858%
  - 최초 해결: 성정훈 @GIST, 16분
- 오픈 콘테스트
  - 제출 75회, 정답 51건
    정답률 69.333%
  - 최초 해결: `edwardblue`, 5분

» **서브태스크 1**: 3의 개수를 센 뒤 홀수 개면 0번 키가 P_1에, 1번 키가 P_0에 가 있고, 나머지 키는 모두 제자리에 있습니다. 짝수 개면 모든 키가 제자리에 있습니다.
» **서브태스크 2 & 3**: 두 가지 풀이가 있습니다.
» **비트마스킹 풀이**: 비트 연산 x & (1 << i)을 사용해 몇 번째 비트가 켜졌는지 정보를 모두 저장한 다음에, 켜진 비트 수가 2개일 때만 swap합니다.
» **전처리 풀이**: 길이 256의 배열 arr을 선언한 뒤, 모든 $0 \le i < j \le 7$에 대해서 $2^i + 2^j$의 값을 구해 $\mathrm{arr}\left[2^i + 2^j\right] = (i, j)$를 저장합니다.

D. 이진수 격자

#dp

출제진 의도 — **Normal**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 81회, 정답 20건
    정답률 24.691%
  - 최초 해결: 김근성 @전남대, 26분
- 오픈 콘테스트
  - 제출 84회, 정답 34건
    정답률 40.476%
  - 최초 해결: `hyperbolic`, 9분

» 원래 $N = 1000$ 정도로 설정한 다음 최댓값을 1000000007로 나눈 나머지를 구하는 문제를 출제하고 싶었으나 실버 문제가 부족해서 무산되었습니다.
» **서브태스크 1**: 완전 탐색을 하면 됩니다. 안타깝게도 **서브태스크 3** 풀이보다 그렇게 많이 쉽지는 않습니다.
» **서브태스크 2**: 대각선을 따라가면서 숫자를 수집하면 됩니다.
» **서브태스크 3**: 모든 칸마다 그 칸까지 도달했을 때 얻을 수 있는 이진수의 최댓값 $M = M[i][j]$를 저장하면 다음 점화식이 성립합니다. 따라서 DP로 문제를 해결할 수 있습니다.

$$M[i][j] = 2 \times \max(M[i-1][j], M[i][j-1]) + D[i][j]$$

(단, $D[i][j]$는 i행 j열 (0-based index) 칸에 적힌 수)

E. ChatGPT 만들기

`#implementation`, `#graph_theory`

출제진 의도 — **Hard**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 199회, 정답 7건
    정답률 3.518%
  - 최초 해결: 김근성 @전남대, 70분
- 오픈 콘테스트
  - 제출 128회, 정답 18건
    정답률 17.969%
  - 최초 해결: `iktk`, 25분

» **서브태스크 1**: 지문을 거의 그대로 구현하면 됩니다.
» 지문에는 문자 c가 주어질 때마다 훈련 코퍼스 내부에 있는 모든 문자 c를 검색한다고 적혀 있지만, 들어올 수 있는 문자의 개수는 29개밖에 되지 않으므로 29개의 문자에 대해 검색 결과를 모두 전처리해 두는 것이 좋습니다.
» 전처리 이후에는 문자가 하나씩 들어올 때마다 다음 문자를 $O(1)$에 예측할 수 있습니다. 이를 구현하면 서브태스크 1이 풀립니다. 시간복잡도는 $O(S+K)$입니다. (S는 훈련 코퍼스의 크기)

» **서브태스크 2**: 인덱스가 어느 값을 넘어가면 문자열에 주기성이 있음을 파악해야 합니다.
» 생성한 문장 s의 길이가 30 이상이라 가정해 봅시다. 등장 가능한 문자의 개수는 29개이므로, 30개의 문자 중 적어도 2개는 서로 같은 것이 있게 됩니다.
» 이를 찾아서 $s[i] = s[j]$라 합시다.
» 그러면 $s[i+1] = s[j+1]$, $s[i+2] = s[j+2]$, ... 등이 모두 성립합니다. 따라서 문자열이 주기성을 갖고, 문자열의 길이는 무한입니다.
» 문자열의 길이가 무한이라면, modulo 연산을 통해 K번째 문자를 $O(1)$에 구할 수 있습니다.
» 문자열의 길이가 유한이라면, 위의 논증에 의해 그 길이는 29 이하입니다. 따라서 입력받은 K가 30 이상이라면 점(.)을 출력하기만 하면 됩니다.

» 문자열의 앞부분에는 특별한 주기성이 없을 수도 있음에 유의해야 합니다.
» 예를 들어, 1개의 문장이 든 훈련 코퍼스 {[gisss]}가 주어졌을 때 생성되는 문자열은 [gisssssssss...입니다.
» 시간복잡도는 $O(S)$입니다.

F. 유리병 속 무한히 터지는 기포

`#simulation`, `#ad-hoc`, `#(hashmap)`

출제진 의도 — **Hard**

- 출제자: 송혜근 @지스트 `invrtd_h`, 이윤수 @전남대 `lys9546`
- 온사이트 대회
  - 제출 24회, 정답 0건
    정답률 0%
  - 최초 해결: -, -분
- 오픈 콘테스트
  - 제출 25회, 정답 11건
    정답률 48%
  - 최초 해결: `hyperbolic`, 14분

» **서브태스크 1**: 1초가 지날 때마다 배열 전체를 훑으면서 시뮬레이션을 그대로 진행하면 됩니다. 시간복잡도는 $O(NT)$입니다.

» **서브태스크 2**: 1번째 구역에서 생성되는 기포가 어디까지 올라갈 수 있는지 생각해보면 사실 그렇게 많이 올라가지 않습니다. 1번째 구역에서 10만 개의 기포가 생성되더라도 한 구역을 올라갈 때마다 기포의 개수가 $1/5$로 줄어들기 때문에, 이를 반복하면 기포의 개수가 금방 0이 되기 때문입니다.

따라서 1번째 구역에서 생성되는 기포가 도달할 수 없는 구역을 구한 뒤 그 곳을 배제하고 시뮬레이션을 돌립니다. 이때 배제되지 않은 구역의 개수가 $O(\log M)$ 정도이므로, 시간복잡도는 $O(T \log M)$입니다.

» **서브태스크 3**: 서브태스크 2의 풀이를 일반화하는 풀이입니다. 완전한 증명을 소개하지는 않고, 스케치 정도만 해 보겠습니다.

» 기포 발생이 $T = 0$ 시점에 $y = Y$ 위치에서 단 한 번만 일어난다고 가정합시다.

» 이 기포 발생으로 인해 영향을 받는 (시간, 위치) 쌍을 나열해봅시다. 그러면 다음과 같습니다.

» $(0, Y) \setminus (1, Y-1), (1, Y), (1, Y+1) \setminus (2, Y-2), (2, Y-1), (2, Y), (2, Y+1), (2, Y+2) \setminus \ldots$

» 마찬가지로, 기포의 개수가 지수적으로 줄어들기 때문에 영향받을 수 있는 (시간, 위치) 쌍의 개수에는 한계가 있습니다. 그 크기는 대략 $O(\log^2 M)$ 정도입니다.

» 기포 발생의 영향을 받지 않은 곳은 항상 기포의 개수가 0입니다.

» 따라서 T번의 기포 발생의 영향을 받은 $O(T \log^2 M)$개의 칸만 관리하면 됩니다.

» (정해) HashMap 등을 사용해 기포의 개수가 0인 칸은 제외하고 기포의 개수가 양수인 칸만 관리해주는 방법을 떠올릴 수 있습니다.

» HashMap을 사용하지 않는 풀이도 있는데, 정해보다 훨씬 빠르게 돕니다.

» 시간복잡도는 $O(T \log^2 M)$입니다.

G. 돈 복사

#bellman-ford, #parametric-search

출제진 의도 — **Hard**

- 출제자: 고민규 @전남대 jjkmk1013, 송혜근 @지스트 invrtd_h
- 온사이트 대회
  - 제출 0회, 정답 0건
    정답률 0%
  - 최초 해결: -, -분
- 오픈 콘테스트
  - 제출 27회, 정답 5건
    정답률 18.519%
  - 최초 해결: aeren, 58분

G. 돈 복사

» 거래를 그래프로 모델링 가능합니다.
» 고객이 가진 A번 물건을 B번 물건으로 바꾸고 그 과정에서 고객이 c원을 잃게 된다면, A에서 B로 향하는 가중치 c의 간선을 줍시다.
» 반대로 고객이 c원을 얻게 된다면, A에서 B로 향하는 가중치 $-c$의 간선을 줍니다.
» 거래 과정에서 물건이 하나만 개입하는 1번 정보 거래와 2번 정보 거래의 경우, 편의상 빈 쪽을 0번 물건으로 채웁시다.

» 이제 이 문제는 음수 사이클을 찾는 문제로 치환됩니다.
» 음수 사이클을 찾는 알고리즘인 벨만-포드 알고리즘을 사용할 수 있습니다.
» 다만 고객이 가진 돈이 부족하면 특정 간선을 사용하지 못하는 경우가 생기게 되므로, 정확히는 벨만-포드를 살짝 수정한 알고리즘을 사용하게 됩니다. 수정된 알고리즘에서는 고객이 가진 돈이 부족하면 간선 전파가 일어나지 않습니다.
» 수정된 알고리즘을 사용하여 고객이 초기 자금 c원을 갖고 음수 사이클에 도달할 수 있는지 없는지 판별할 수 있습니다. 알고리즘의 정당성은 표준 벨만-포드 알고리즘이 정당하게 음수 사이클 존재성을 판별할 수 있다는 증명을 변형하여 증명 가능합니다.

» 마지막으로, 음수 사이클에 도달하기까지 돈이 최소 얼마나 필요할지는 이분 탐색으로 구현 가능합니다. 초기 자금 c원을 갖고 음수 사이클에 도달할 수 있다면 $c+1$원을 갖고도 가능하고, 초기 자금 d원을 갖고 음수 사이클에 도달할 수 없다면 $d-1$원을 갖고도 불가능하기 때문입니다.

» 시간복잡도는 (벨만 포드의 시간 복잡도) $\times$ (이분 탐색의 시간 복잡도) = $O(MN\log C)$입니다.

부록: 정당성을 증명해봅시다.

» **주장**. 문제 상황을 DP로 모델링할 수 있습니다.

» $\mathrm{dp}[i][j]$를 j번 이하의 교환을 사용하여 i번 물건에 도달했을 때 가지고 있을 수 있는 가장 큰 돈이라고 합시다.

» Base case는 $\mathrm{dp}[0][0] = c, \mathrm{dp}[i][0] = -\infty (i \geq 1)$이라고 정의합니다. c는 초기 자금입니다.

» 이때 다음 공식이 성립합니다:
$\mathrm{dp}[i][j] = \max(\mathrm{dp}[i][j-1], \max_{v \in S}(\mathrm{dp}[v][j-1] + \mathrm{cost}[v][i]))$

» 이때 집합 S는 v에서 i로 향하는 간선이 존재하고 $\mathrm{dp}[v][j-1] + \mathrm{cost}[v][i]$의 값이 0 이상이 되게 하는 모든 v들의 집합입니다.

» **주장**. 알고리즘을 $n+1$라운드 진행했을 때 DP 테이블이 업데이트되면, 이 그래프에는 양수 사이클이 존재하고 그 사이클은 0번 정점으로부터 접근할 수 있습니다.
» 여기서 $n+1$을 쓰는 이유는 그래프의 정점의 개수가 $n+1$이기 때문입니다.
» **증명**. 정점 v의 값이 업데이트되었다는 것은 직전 라운드에 정점 v로 향하는 간선을 가진 정점 중 하나가 업데이트되었다는 것을 의미합니다. 이를 역추적하는 것을 반복하면 결국 0번째 라운드에서 $-\infty$이 아닌 값을 가진 유일한 정점인 0번 정점에서 시작해서, $n+1$번째 라운드에서 업데이트된 정점인 v로 향하는 path가 형성됩니다. 비둘기집의 원리에 의해 이 path는 반드시 사이클을 하나 갖고 있습니다. 업데이트는 언제나 값을 늘리므로, 이 사이클을 따라 한 바퀴 돌면 반드시 정점에 할당된 값이 늘어납니다. 따라서 이 사이클은 양수 사이클입니다.

» **주장**. 알고리즘을 k라운드 진행했을 때 DP 테이블이 업데이트되지 않으면, $k+1$라운드 진행해도 DP 테이블은 업데이트되지 않습니다.

» **증명**. k라운드에서 업데이트가 일어나지 않는다면 $\mathrm{dp}[i][k] = \mathrm{dp}[i][k-1] \geq \max_{v \in S} \mathrm{dp}[v][k-1] + \mathrm{cost}[v][i]$가 성립하고 그 역도 성립합니다. 이제

$$\mathrm{dp}[i][k+1] = \max(\mathrm{dp}[i][k], \max_{v \in S}(\mathrm{dp}[v][k] + \mathrm{cost}[v][i]))$$
$$= \max(\mathrm{dp}[i][k-1], \max_{v \in S}(\mathrm{dp}[v][k-1] + \mathrm{cost}[v][i]))$$
$$= \mathrm{dp}[i][k-1] = \mathrm{dp}[i][k]$$

을 얻습니다.

» **주장**. 알고리즘을 $n+1$ 라운드 진행했을 때 DP 테이블이 업데이트되지 않는 순간이 오면, 돈 복사는 불가능합니다. 위 주장에 의해 DP 테이블은 앞으로 영원히 업데이트되지 않을 것이므로, 각 정점마다 벌 수 있는 돈의 최댓값이 정해졌기 때문입니다.

H. 기초마법학

`#sorting`, `#greedy`, `#pq`

출제진 의도 — **Challenging**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 3회, 정답 0건
    정답률 0%
  - 최초 해결: -, -분
- 오픈 콘테스트
  - 제출 9회, 정답 6건
    정답률 66.667%
  - 최초 해결: `chmpro`, 70분

» **서브태스크 1**: 브루트포스를 사용합니다. 마법진의 오른쪽 위 점이 가질 수 있는 x좌표의 종류는 1억 가지이고 y좌표의 종류도 1억 가지인 것처럼 보이지만, 실제로는 x좌표가 어떤 마법원의 x좌표와 일치하고 y좌표가 어떤 마법원의 y좌표와 일치하는 N^2개의 경우만 살펴봐도 충분합니다. 따라서 $O(N^3)$의 시간복잡도로 문제를 해결할 수 있습니다.

» **서브태스크 2**: 점의 y좌표가 모두 같으니, 점을 x좌표 기준으로 정렬한 뒤 마법진이 모든 색깔의 마법원을 포함하기 직전까지 x좌표를 계속해서 늘려 나갑니다. 시간복잡도는 정렬의 시간복잡도 $O(N \log N)$입니다.

» **서브태스크 3**: **서브태스크 2**의 논리를 확장해봅시다. 마법진이 네 직선 $x = 0, x = X, y = 0, y = Y$로 둘러싸인 부분과 같다고 합시다. Y의 값이 고정되어 있을 때, 마법진의 힘을 최대화하는 것은 **서브태스크 2**의 논리를 그대로 사용 가능합니다. X의 값을 계속해서 늘려 나가다가 더 이상 늘릴 수 없을 때 Y의 값을 줄여 봅니다.

» 이 알고리즘이 가능한 이유는 Y의 값이 정해졌을 때 X의 최댓값 $X_{\max}(Y)$가 감소하지 않는 함수이기 때문입니다.

» Y의 값을 줄일 때마다 어떤 점이 마법진에서 빠지게 되는지 결정해야 합니다. 마법원의 y좌표가 큰 순으로 빠지게 되므로, 마법진 안의 모든 점을 관리하는 pq를 만들어 시간복잡도 $O(N \log N)$을 지킬 수 있습니다.

I. DevNight 운영

`#dijkstra`

출제진 의도 — **Challenging**

- 출제자: 송혜근 @지스트 `invrtd_h`
- 온사이트 대회
  - 제출 3회, 정답 0건
    정답률 0%
  - 최초 해결: -, -분
- 오픈 콘테스트
  - 제출 20회, 정답 5건
    정답률 25%
  - 최초 해결: `chmpro`, 97분

» **서브태스크 1**: 하나의 시작 정점에 대하여 데이크스트라 알고리즘을 한 번 돌리면 다른 모든 정점으로 향하는 최단 경로의 길이를 알 수 있습니다. 따라서 K개의 커뮤니케이션 룸 각각을 한 번씩 시작점으로 하여 데이크스트라 알고리즘을 돌리면 됩니다.

» 데이크스트라 알고리즘을 $O(M \log M)$에 구현한다고 했을 때, 시간복잡도는 $O(KM \log M)$입니다. 서브태스크 1은 K가 작아서 이 정도로도 충분히 돌아갑니다.

» **서브태스크 2**: Path를 만든 뒤 Path를 따라 sweeping하며 각 정점마다 양옆으로 커뮤니케이션 룸을 최대 2개까지 관리해 줍니다. 두 번째로 가까운 커뮤니케이션 룸은 각 컨퍼런스 룸의 왼쪽 2개, 오른쪽 2개 총 4개 중 하나에 있습니다.

» **서브태스크 3**: 에디토리얼에서 설명하는 풀이 말고도 검수자 풀이가 하나 더 있습니다. 아마 검수자 풀이가 더 쉬운 것 같습니다.
» Dijkstra 알고리즘에 대한 몇 가지 사전 지식이 필요합니다.
» **Multisource Dijkstra**: 시작 정점이 여러 개고 여러 개의 시작 정점 중 아무 곳에서나 출발할 수 있다고 가정합시다. 이런 상황에서도 도착 정점으로 향하는 최단 경로의 길이를 구할 수 있습니다.
» **증명**: Muitisource Dijkstra를 동치인 Standard Dijkstra 문제로 치환함으로써 증명할 수 있습니다. 임의로 정점을 하나 추가한 후 그 정점과 모든 시작 정점을 길이 0인 단방향 간선으로 연결합시다. 그리고 추가한 정점을 새로운 시작 정점으로 합니다.

» 문제는 각각의 컨퍼런스 룸으로부터 여러 개의 커뮤니케이션 룸 중 하나까지 가는 최단 경로를 요구하지만, Multisource Dijkstra 관점에서 생각해보면 여러 개의 커뮤니케이션 룸 중 하나에서 출발하여 각각의 컨퍼런스 룸으로 가는 최단 경로를 생각하는 것이 더 효과적입니다. 따라서 앞으로는 커뮤니케이션 룸을 시작 정점이라고 부르겠습니다.

» **Shortest Path Tree**: 루트 노드로부터 시작해 각 정점으로의 최단 경로로 구성된 tree입니다.

» Shortest Path Tree는 항상 존재하고, 데이크스트라 알고리즘을 돌리는 과정에서 얻을 수 있습니다. 알고리즘의 매 라운드마다 최단 경로의 길이가 아직 확정되지 않은 정점 중 가장 할당된 값이 작은 정점을 결정하게 됩니다.

» 이때 새롭게 결정된 정점은 이전에 이미 최단 경로의 길이가 결정된 정점 중 하나에서 영향을 받았을 것입니다. 그렇다면 영향을 받은 정점과 영향을 준 정점을 잇습니다.

» Shortest Path Tree는 유일하지 않습니다. 하지만 이 문제에서는 사람들이 최단 경로의 길이가 같으면 정점의 번호가 더 작은 커뮤니케이션 룸을 선호하기 때문에, Shortest Path Tree가 유일합니다.

» Multisource Dijkstra에도 Shortest Path Tree와 비슷한 것이 존재합니다. 단 이때는 Tree가 아닌 Forest(Tree가 여러 개 있는 것)로 존재합니다. 앞서 살펴본 증명에서 추가한 정점을 제거하면 Forest를 얻을 수 있습니다.

» 이때 **각각의 Forest는 여러 개의 시작 정점(커뮤니케이션 룸) 중 하나만을 루트 노드로 갖습니다.**

» 각각의 정점들이 어느 Forest에 속하느냐에 따라 정점들을 그룹 지어 줄 수 있습니다.

» 이제 k번 커뮤니케이션 룸을 루트 노드로 하는 정점들의 집합 V_k에 대해 생각해 봅시다.

» 우리는 V_k의 모든 정점들에 대해서 k번 커뮤니케이션 룸이 **아닌** 커뮤니케이션 룸 중 가장 가까운 커뮤니케이션 룸을 찾아야 합니다.

» V_k와 인접한 정점들을 출발점으로 해서 다시 dijkstra를 돌리면 됩니다. 인접한 정점들을 pq에 push할 때 최단 경로의 길이를 같이 넣어 줍니다.

» 예를 들어 V_k와 인접한 정점이 10번, 11번 정점이고 10번 정점과 가장 가까운 커뮤니케이션 룸까지의 거리가 100, 11번 정점과 가장 가까운 커뮤니케이션 룸까지의 거리가 110이라 합시다. 그러면 처음에 pq에 (10, 100)과 (11, 110)을 넣습니다.

» 이 알고리즘이 표준 dijkstra와 같은 방식으로 동작한다는 것은, multisource dijkstra 증명과 마찬가지로 새로운 정점을 하나 만들고 그 정점과 시작 정점들을 이어준 그래프를 만들면 증명할 수 있습니다. 새로 만든 간선들의 거리는 가장 가까운 커뮤니케이션 룸까지의 거리로 하면 됩니다.

» 모든 V_k에 대해서 dijkstra를 돌려 주면 됩니다.

» 이렇게 dijkstra를 K번 돌리면 시간복잡도가 $O(KM \log M)$이 될 것 같습니다. 하지만 잘 생각해보면 dijkstra를 한 번 돌릴 때 그래프 전체의 정보가 필요하지는 않습니다. 오직 V_k와 그와 인접한 정점 그리고 정점들 사이의 간선 정보만 필요합니다.

» 이와 관련해서 그래프의 크기를 줄이는 최적화를 잘 해 주면 시간복잡도를 $O(M \log M)$으로 맞출 수 있습니다.