



# 부산 국제 마라톤

**[문제]** 부산에서 개최될 2026년 국제 마라톤 대회 (Busan International Marathon, BIM)를 위하여 마라톤 코스를 설계하려고 한다. 마라톤 코스는 부산 도심 도로를 연결하여 구성한다. 단 선수들이 혼돈을 일으키지 않도록 한번 지나온 곳은 다시 지나지 않도록 해야 한다. 코스는 도심의 한 정점(vertex)를 출발하여 정확하게 42km의 거리를 지나 출발점으로 돌아온다. 원래 마라톤 42.195km의 거리에서 0.195km는 출발 장소 스타디움의 트랙을 달리도록 하여 처리하므로 여러분은 주어진 도로망 그래프에서 정확하게 42km 거리의 사이클(Cycle)을 찾아야 한다. 단 같은 지점을 한번 이상 지나도록 설계하지 않는다. 즉 경주루에 있는 모든 점은 출발점을 제외하고는 반드시 서로 달라야(unique) 한다.

아래 그림-1에는 도로망이 나타나 있다. edge에 표시된 red numbers는 해당 edge의 거리(km)을 의미한다. 모든 거리는 정수이다. 만일 출발 스타디움이 1번에 있다고 하면  $[1, 6, 11, 15, 9, 1]$ 의 사이클의 길이는  $11 + 14 + 5 + 7 + 5 = 42$ 가 되어 마라톤 코스로 사용될 수 있다. 또 다른 코스도 찾을 수 있는데,  $[1, 6, 4, 8, 15, 10, 3, 13, 1]$  코스도 42km이다. 만일 이 같이 사용가능한 코스가 1개 이상일 경우에는 더 많은 정점을 지나는 코스를 더 선호한다. 대부분의 관중들이 교차로(정점)에서 응원을 하기 때문에 정점을 많이 지나도록 설계하는 것이 좋기 때문이다.1) 만일 지나는 정점의 개수 마저도 같을 경우에는 edge 길이의 최대값이 낮은 쪽을 더 선호한다. 예를 들어 그 길이 순서가  $(12, 9, 8, 7, 3, 2, 1)$ ,  $(10, 10, 10, 5, 5, 1, 1)$  이라면 최대값이 더 작은 후자를 선택한다. 즉 길이 순서의 lexicographic 순서로 더 빠른 것을 선택한다.

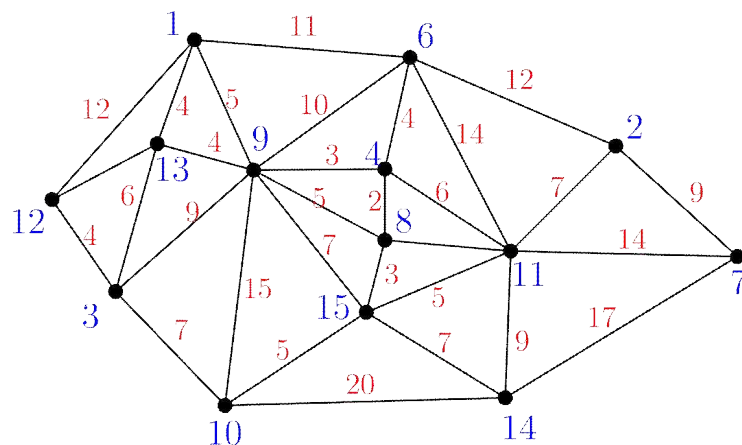


그림-1. 14개의 정점으로 구성된 도로망. 청색은 vertex  $v_i$ 의 index  $i$ 이며 붉은 색으로 표시된 숫자는 해당 edge  $(v_i, v_j)$ 의 길이(km)를 나타낸다.

1) 실제 마라톤 코스를 설계할 때 직선 주루와 곡선 주루를 적절히 섞어 선수가 달리는 동안 주변의 시각적 변화를 느끼도록 한다. 예를 들어 완전 직선으로만 된 42.195Km를 달리게 하면 선수들을 심리적으로 무척 힘들게 된다.

**[입출력]** 입력 파일 **stdin**의 첫 줄에 도로망 그래프의 정점의 수  $N$ 이 정수로 주어진다. 단  $10 \leq N \leq 50$  이다. 출발점 스타디움은 항상  $v_0$ 로 고정되어 있다. 즉 선수들은 항상  $v_0$ 에서 출발하여 여기로 되돌아 온다. 그 다음 이어지는  $N$ 개의 각 줄에  $v_i$ 의 index  $i$ 와 위치 좌표  $(x_i, y_i)$ 가 3개의 정수로 주어진다. 단  $1 \leq x_i, y_i \leq 100$ . 각 도로의 거리(km), 즉 edge  $(v_i, v_j)$ 의 두 끝점  $(x_i, y_i), (x_j, y_j)$  사이의 거리는 두 점의 2차원 공간거리의 floor로 계산한다. 따라서 그 값은 항상 정수 km 이다. 즉 다음과 같다.

$$distance(v_i, v_j) = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor$$

정점의 좌표가 모두 제시된 다음, 이어지는 줄에는 전체 edge의 개수  $M$ 이 정수로 주어지고, 다시  $M$ 개의 각 줄에 edge  $(v_i, v_j)$ 를 나타내는 두 정점의 index가 edge 일련 번호와 함께 'd i j'로 주어진다. 에지의 개수  $M$ 은  $M \leq 3N - 6$ 이다.<sup>2)</sup>

출력은 0번에서 출발하여 0번으로 되돌아오는 42km의 경로 중 가장 “좋은”것을 골라 그 정점 번호를 순서로 한 줄에 모두 출력해야 한다. 즉 이 순서의 처음과 끝은 반드시 0이 되어야 한다. 만일 그러한 cycle 경로가 존재할 경우  $v_0$ 에서 연결된 2개의 정점이 존재하는데,  $v_0$ 에 연결된 cycle  $v_i$  중에서 index  $i$ 가 작은 쪽으로 출발해야 한다. 예를 들어 [0, 5, 7, 2, 11, 15, 9, 0] 사이클이 42km라면 그 역순인 [0, 9, 15, 11, 2, 7, 5, 0]도 가능하지만 이들 중에서 스타디움 0에 연결된 두 이웃 정점 {5,9}중에서 더 작은 번호인 5쪽으로 출발해서, 9번 정점에서 스타디움 0으로 들어오도록 지정한다.

만일 입력 도로망에 따라서 어떤 경우에는 그렇게 42km로 딱 떨어지는 Cycle 경로가 존재하지 않는 경우도 있는데 이런 경우에는 음수인 -1을 첫 줄에 출력해서 이 사실을 알려야 한다.

**[제한조건]** 프로그램 이름은 **bmara**. {c, cpp, java, py}이며 제출 허용 횟수는 25회이다. 데이터 당 제한시간은 최대 3초, 그리고 token은 최대 700개이다. 단 이번 과제부터는 #define 문자열을 사용하여 코드의 길이를 줄이는 편법은 사용할 수 없다. #define은 반드시 1줄만 가능하며 multiline string은 더 이상 사용하지 않는다. 만일 이 방법을 사용하면 이전 점수에 관계없이 0점 처리된다.

**[도움말]** 전형적인 branch and bound를 사용하면 좀 더 빠르게 찾을 수 있다. 중요한 것은 가장 좋은 경로를 찾아야 하므로 어떤 42km인 경로를 찾았다고 해서 바로 “만세”를 부르면 안된다. 왜냐하면 그 보다 더 좋은 경로가 없음을 확인해야 하기 때문이다. 수업에서 여러번 말을 했지만 backtrack과 branch and bound는 optimal한 답을 찾아주는 exhaustive algorithm이기 때문이다.

상식으로 출발점에서 여러 이웃 노드가 있으면 짧은 edge를 선호한다. 그래야 정점의 수를 늘일 수 있기 때문이다. 그리고 출발점에서 32km까지 왔는데 이 끝 점에서 0번까지의 직선 거리가 10km가 넘는다면 이 방향으로 더 이상 가망성이 없기 때문에 bound를 시켜야 한다.

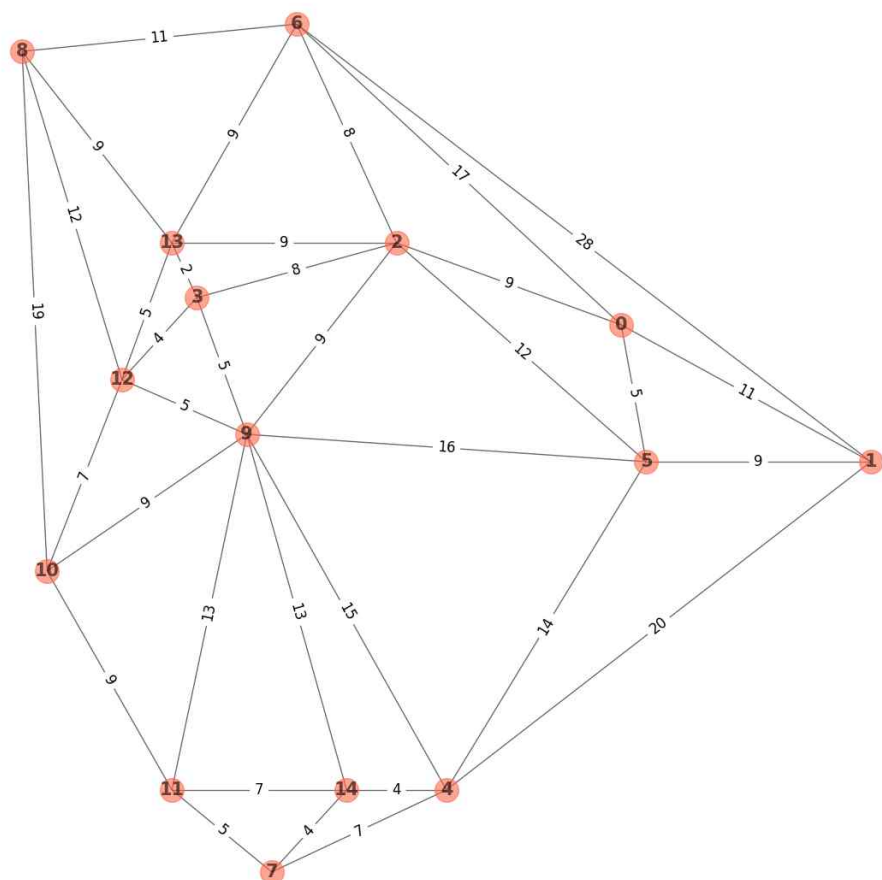
좀 더 빨리 bound를 시키기 위해서 greedy algorithm으로 어떤 한 마라톤 코스를 하나 구해놓고 이를 중심으로 활용해도 좋을 것이다.

꽤 손이 들어가는 문제이다. 3시간 이내 풀 수 있다면 A+ 코딩+알고리즘 실력의 소유자라고 할 수 있을 것이다. 그래프에 관련된 문제는 코테 등에서 고난이도 문제로 흔히 나오기 때문에 Graph를 읽는 것, BFS, DFS를 하는 코드는 완전히 손에 “불혀”두어야 한다.

2) 실제 해당 점으로 구성되는 그래프는 들로니 삼각분할(Delaunay Triangulation) 그래프이다. 이 도로망 데이터를 생성하는 python 코드도 같이 제공된다. 단 이코드를 사용하기 위해서는 과학계산 전용 모듈, 그래프 가시화 모듈인 scipy, networkx, matplotlib를 미리 설치해야 한다.

[예제]

stdin-1	stdin-2	stdin-3
15            //N	2   0   2	20   5   9
0       39   39	3   0   5	21   6   8
1       49   34	4   0   6	22   6   13
2       30   42	5   1   4	23   7   11
3       22   40	6   1   5	24   7   14
4       32   22	7   1   6	24   7   14
5       40   34	8   2   3	25   8   10
6       26   50	9   2   5	26   8   12
7       25   19	10   2   6	27   8   13
8       15   49	11   2   9	28   9   10
9       24   35	12   2   13	29   9   11
10       16   30	13   3   9	30   9   12
11       21   22	14   3   12	31   9   14
12       19   37	15   3   13	32   10   11
13       21   42	16   4   5	33   10   12
14       28   22	17   4   7	34   11   14
35       // M	18   4   9	35   12   13
1       0   1	19   4   14	
stdout		
0 .....0		



[도움말2] 아래 python 코드로 두 지점 P,Q의 정수거리를 구할 수 있다. 물론 numpy를 사용하면 더 간략하게, 다음과 같이  $\text{distance} = \text{numpy.linalg.norm}(P - Q)$  구할 수 있다. 물론  $\text{int}(\text{distance})$ 로 정수화해야 한다.

```
import math
import random

def pdist( P, Q) :
    x1, y1 = P[0], P[1]
    x2, y2 = Q[0], Q[1]
    intdist = int( math.sqrt( (x1-x2)**2 + (y1-y2)**2 ) )
    return( intdist )

M = 50
for w in range(10) :
    P = ( random.randint(1, M), random.randint(1, M) )
    Q = ( random.randint(1, M), random.randint(1, M) )
    idist = pdist( P, Q )
    print(f" {P=}, {Q=}, {idist=}")
```