

# 2016 FunctionCup 풀이

출제 : functionx

검수 : zigui

# A. Aurora Princess

먼저, 모든 사람들을 특정 배열에 체크한 후, 사망하거나 미국에 간 사람을 모두 배열에서 체크를 해제한 다음, 모든 사람들에 대하여 자기 자신, 어머니, 아버지가 모두 체크되어 있는지 구하면 된다.

## 분류

단순 구현

## B. Baseball Watching

각 학생이 9회 동안 야구를 보는 위치의 패턴을 9자리의 3진법 수로 나타낼 수 있다. 이때 가능한 수의 개수는  $3^9 = 19683$  개이다. 문학 선생님이 야구를 보는 위치의 패턴으로 가능한 19683개의 경우에 대해서 9회까지 버티는 학생의 수를 세면 된다. 문학 선생님이 특정 위치에서 야구를 볼 때 9회까지 버티는 3진법 수의 수는  $2^9 = 1024$ 가지이다. 따라서 1024가지 패턴 중 한 패턴을 가지는 학생의 수의 합을 구해주면 9회까지 버티는 학생의 수를 셀 수 있다. 이런 식으로 답을 구하면 시간복잡도는  $O(6^N)$ 이다.

### 분류

비트마스크, 백트래킹

## C. Corrupt Election

표들 중에서 유효한 표는  $\max(A, B) \leq X-1$  이하이고  $A+B \leq Y-1$  이하인 표이다. 체키가 당선되려면 유효한 표들의  $A-B$ 들의 합이 양수가 되어야 한다.

입력에서 주어진  $(\max(A, B), A+B)$  좌표들을 가지고 출현하는  $x$ -좌표들(0과 100,000 추가)과  $y$ -좌표들(0과 100,000 추가)을 나누면 최대  $N+2$ 개의  $x, y$  좌표가 나온다.  $x$ -좌표들의 정렬된 배열  $xc[i]$ 와  $y$ -좌표들의 정렬된 배열  $yc[j]$ 를 구하자. 그러면  $O(N^2)$ 개의  $(i, j)$  쌍들에 대해서  $([0 \sim xc[i]], [0 \sim yc[j]])$  직사각형 범위에 있는 점들의 가중치( $A-B$ )들의 합  $D[i][j]$ 를 구해주어야 한다.

그러면  $i, j > 0$ 이면  $D[i][j] = D[i-1][j] + D[i][j-1] - D[i-1][j-1] + ((xc[i], yc[j]))$ 에 있는 점들의 가중치의 합)이고 그 외의 경우에는  $D[i][j] = 0$ 이다.

$xc[i] \leq X < xc[i+1]$ 이고  $yc[j] \leq Y < yc[j+1]$ 인 모든  $(X, Y)$ 에 대해서 유효한 표들의  $A-B$ 들의 합은  $D[i][j]$ 이므로, 모든  $i, j$ 에 대해  $D[i][j] > 0$ 이면 답에  $(xc[i+1] - xc[i]) \times (yc[j+1] - yc[j])$ 를 더해주면 된다.

총 시간복잡도는  $O(N^2)$ 이다.

## 분류

동적계획법

# D. Do Not Touch Anything

$\text{ceil}(R/N) * \text{ceil}(C/N)$ 의 값을 구하면 된다. 이때 답이 32비트 범위를 넘어갈 수 있다는 것은 주의해야 한다.

## 분류

단순 구현

## E. Exchange Problem

1원 ~  $P_{N-1}+P_N$ 원 범위의 가격에 대해서만 고려하면 된다. 이는 동적계획법을 이용하여 구현할 수 있다.

그리디 알고리즘(병찬이의 방법)에서 C원을 낼 때 필요한 동전의 수를  $Gr[C]$ 라 하면  $Gr[C] = Gr[C-P_i] + 1$  ( $i$ 는  $P_i \leq C$ 를 만족하는 최대  $i$ )이다.

동적계획법 알고리즘에서 C원을 낼 때 필요한 동전의 수를  $Dp[C]$ 라 하면  $Dp[C] = \min(Dp[C-P_i] + 1)$ 이다.

앞서 언급한 범위에서 그리디 알고리즘이 모두 최적이면 모든 범위에서 그리디 알고리즘이 최적이라는 것은 귀류법으로 보일 수 있다. 만약 그리디 알고리즘이 최적이 아닌 가격  $P > P_{N-1}+P_N$ 이 존재한다면, 그 가격들 중 최솟값을  $K$ 라 하자.

만약  $K$ 원을 내는 최적해에서  $P_N$ 원짜리 동전을 낸다면, 그 방법에서  $P_N$ 원짜리 동전 하나를 제거하여  $K-P_N$ 원을 만드는 방법 역시 최적해이다.  $K > P_N$ 이므로, 그리디 알고리즘에 의하면  $P_N$ 원짜리 동전을 먼저 낼 것이다. 이는 그리디 알고리즘이 최적이 아니라는 가정과 모순이다.

한편,  $K$ 원을 내는 최적해에서  $P_N$ 원짜리 동전을 내지 않는다면,  $i$ 가 존재하여 최적해에서 동전 하나를 제거하여  $K-P_i$ 원을 만드는 방법도 최적해이다. 한편,  $K-P_i < K$ 이므로 그리디 알고리즘을 이용하여  $K-P_i$ 원을 내는 방법은 최적해이다.  $P_N \leq K-P_i$ 이므로  $K-P_i$ 원을 만드는 최적해에는  $P_N$ 원짜리 동전이 1개 이상 필요하다. 따라서  $K$ 원을 만드는 최적해에는  $P_N$ 원짜리 동전이 1개 이상 필요하다. 이는 가정과 모순이다.

따라서, 1 ~  $P_{N-1}+P_N$ 원 범위의 가격에서 그리디 알고리즘이 최적이라면 모든 범위에서 그리디 알고리즘이 최적이라는 것을 알 수 있다.

## 분류

그리디, 동적계획법

## F. Fairies' Sorcery

우선 요정들이 마법을 쓴 후의 상태를 어떻게 구할 지 생각해보자. 각 요정의 위치와 요정의 순서를 배열로 갖고 있다.

두 마리의 요정이 마법을 쓰는 경우는 두 요정의 위치를 구한 후 바꿔주면 된다.

한편, 한 마리의 요정이 마법을 쓰는 경우는 작은 아이디어를 이용하면 두 요정이 마법을 쓸 때와 똑같은 방법으로 수행할 수 있다. 0을 추가하여, 처음 요정의 순서를  $\{0, 1, 2, \dots, N\}$ 라고 했을 때 요정 P가 마법을 쓴 경우 0과 P를 바꿔주면 된다. 그러면 요정의 순서는  $\{P, 1, 2, \dots, P-1, 0, P+1, \dots, N\}$ 이 되는데, 실제 요정들의 순서는 0부터 시작하여 수들을 읽는 방식으로 구하면  $\{P+1, \dots, N, P, 1, 2, \dots, P-1\}$ 이 되어 문제에서 요구한 대로 요정들이 움직여졌다.

이 아이디어를 사용하면 모든 마법을 O(1)에 처리할 수 있다.

같은 종류의 마법을 두 번 사용하면 요정들의 순서가 바뀌지 않는다. 따라서 마지막 상태에서 M-1개의 마법들을 역순으로 사용하면 M-1개의 마법들을 쓰기 전의 상태를 알 수 있다.

처음 상태  $A=\{0, 1, 2, \dots, N\}$ 과 마지막 상태에서 M-1개의 마법들을 역순으로 사용한 후의 상태 B가 주어졌을 때, 빠진 마법의 위치가 W라 하자. 그러면 A에서 주어진 1 ~ W-1번째 마법들을 사용한 상태  $A'$ 와, B에서 주어진 1 ~ W-1번째 마법들을 사용한 상태  $B'$ 에 대해서  $A'$ 에서 한 번의 마법을 사용한 후  $B'$ 와 동치의 상태가 되어야 한다.  $A'$ 와  $B'$ 에서 숫자들이 앞으로 몇 칸 당겨졌을 수도 있기 때문에  $A'$ 와  $B'$ 를 직접 비교하기보단  $A'$ 와  $B'$ 의 차이를 구하여 무슨 마법을 썼는지 찾아야 한다.

$A'$ 와  $B'$ 의 차이  $C'[i]$ 를 다음과 같이 정의하자.

$$- C[i] = (A' \text{에서 } i \text{의 위치} - B' \text{에서 } i \text{의 위치} + N+1) \bmod N+1$$

그러면  $A' = \{0, 3, 4, 2, 1\}$ ,  $B' = \{4, 2, 3, 0, 1\}$  인 경우,  $C'[0]=2$ ,  $C'[1]=0$ ,  $C'[2]=4$ ,  $C'[3]=2$ ,  $C'[4]=20$ 이다.

보다시피,  $C'[1]$ 과  $C'[2]$ 를 제외한 모든 값이 같다. 따라서 우리는 1과 2를 바꾸는 마법을 생각해볼 수 있으며, 실제로  $A'$ 에서 1과

2를 바꾸면 C'의 값이 모두 같아지므로 B'와 동치가 된다.

즉, A'에서 한 번의 마법으로 B'로 만들 수 있다면 N+1개의 C'[i]들 중에서 N-1개의 값들이 같아야 한다. 즉, 임의의 A', B'에 대해 C'에서 최빈값을 구한 후 빈도가 N-1인지 확인한 후, 다른 C'[i]의 값을 가진 두 개의 i를 찾아서 답이 되는지 확인해보면 된다. 다만,  $N \leq 3$ 인 경우 N-1이 과반수가 아니므로 답이 여러 개 나올 수 있는데, 이때는 N이 작으므로 brute-force로 답을 찾으면 된다.

맨 처음 상태의 A와 B를 비교하고, 1, 2, …, M-1개의 마법을 쓴 후 A', B'를 비교하자. 한 번 마법을 쓴 후 A'에서 2개의 값이 바뀌고 B'에서도 2개의 값이 바뀌므로 C'에서는 최대 4개의 값이 바뀐다. C'에서 한 개의 값이 바뀔 때 최빈값을 경신하는 것은 STL-set 등의 자료구조를 이용하여  $O(\log N)$ 으로 가능하다.

답을 구하는 총 시간복잡도는  $O((N+M) \log N)$ 이다.

## 분류

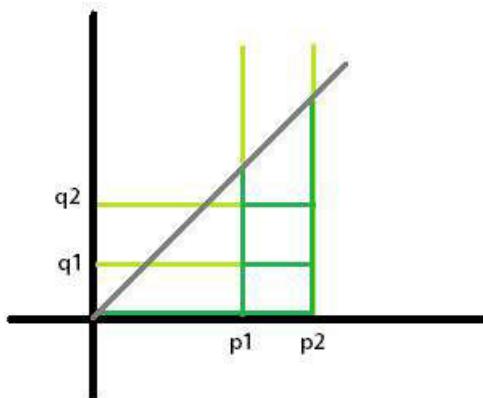
자료구조, Ad Hoc

## G. Grid Forest

이 문제는 prime gap (인접한 두 소수의 차이)이 매우 작다는 것을 이용하여 매우 작은 공간 안에서 BFS를 이용하는 문제이다.  $(x, y)$ 에서  $(0, 0)$ 이 보일 필요충분조건은  $x, y$ 가 서로소라는 의미이다.

우선  $x$ 좌표 또는  $y$ 좌표가 0 또는 1인 경우는 따로 처리해주자. 또,  $(x, y)$ 에서의 최단경로와  $(y, x)$ 에서의 최단경로는 직선  $y=x$ 를 기준으로 대칭이기 때문에  $x > y$ 인 경우만 생각하자.

100,000,000 이하 자연수 범위에서 prime gap이 커봐야 2200이므로  $p_1 \leq x \leq p_2$ 인 소수  $p_1, p_2$ 를 구하는 데  $O(220 * \sqrt{10^8})$ 의 시간복잡도가 필요하다. 한편, 비슷한 방법으로  $q_1 \leq y \leq q_2$ 인 소수  $q_1$  ( $q_1$ 이 없으면 1로 간주),  $q_2$  ( $q_2$ 가 없으면  $p_2-1$ 로 간주)를 구해야 하는데  $q_1, q_2$ 에 대해서는 추가적으로 ‘ $p_1$  이상  $p_2$  이하의 정수 중  $q_1, q_2$ 의 배수가 존재하지 않는다.’라는 조건을 만족해야 한다. 이 조건이 있어도  $q_2-q_1$ 은 커봐야 2,000을 넘지 않는다.



그러면 위 그림에서 진한 초록색으로 칠해진 지역은 무조건 갈 수 있다. 한편 회색으로 칠해진 지역 중  $(1,1)$ 을 제외한 모든 지역은 무조건 갈 수 없다.  $(x, y)$ 는  $(p_1 \sim p_2, q_1 \sim q_2)$  직사각형 영역 안에 있기 때문에  $(x, y)$ 에서의 최단경로는 무조건 직사각형 영

역 안에 있다.

한편, 초록색으로 칠해진 지역에 대해서 최단경로는 남쪽 또는 서쪽으로만 가는 경로이므로  $(x, y)$ 에서의 최단경로들 중  $(p_1, q_1)$ 을 지나는 경로가 항상 존재한다.

즉, 정답을 구하기 위해서는  $(x, y)$ 에서 시작하여  $(p_1 \sim p_2, q_1 \sim q_2)$  직사각형 영역만 탐색하여  $(p_1, q_1)$ 까지 가는 최단거리를 구하면 된다. 이는 BFS 알고리즘으로 간단하게 구현할 수 있다. 매 격자마다  $\text{gcd}$ 를 구해야 하므로 BFS 알고리즘의 시간복잡도는  $O(220 * 2000 * \log 10^8)$ 이다.

## 분류

수학, 정수론, BFS

## H. Hello World!

h, e, l, o, w, r, d의 값에 0~9의 값을 모두 대입해보고 복면산을 성립하는 경우가 나왔을 때 바로 답을 출력하고 프로그램을 종료하면 된다.

h, w는 0이 될 수 없다는 점과 h, e, l, o, w, r, d가 전부 다르다는 것을 꼭 체크해야 한다.

### 분류

백트래킹

# I. Ignition

우선, 태우는 노드를 고정시켰을 때 답을 어떻게 구하는지 생각해보자.

태우는 노드를 A라고 했을 때 노드 B가 타는 데 걸리는 시간은 노드 A에서 노드 B까지의 최단경로의 길이와 같다.

한편, 노드 B, C를 연결하는 길이  $|$ 의 간선에 대해서, 노드 B가 타는 데 걸리는 시간이  $b$ 이고 노드 C가 타는 데 걸리는 시간이  $c$ 이면  $b$ 와  $c$ 의 차이는  $|$  이하이다. 간선이 타는 데 걸리는 시간이  $x$ 라면, 노드 B에서 불은 불은 총  $x-b$ 만큼 태우고 노드 C에서 불은 불은 총  $x-c$ 만큼 태우므로 방정식  $(x-b)+(x-c) = |$ 을 세울 수 있고, 이 방정식의 해는  $x = (|+b+c)/2$ 이다.

따라서 태우는 노드를 하나 정하고 그 노드를 시작점으로 하는 최단경로의 길이를 모두 구하면, 그 이후로는  $O(M)$ 의 시간복잡도로 그래프가 타는 데 걸리는 시간을 구할 수 있다.

플로이드-워셜 알고리즘을 이용해서 미리 모든 쌍 간의 최단경로를 구한 다음, 각 노드를 기점으로 하여 그래프가 타는 데 걸리는 시간을 계산하면  $O(N^3 + MN)$ 의 시간복잡도로 문제를 해결할 수 있다.

## 분류

플로이드-워셜 알고리즘, 최단경로

# J. Jolly Jelly Jiffy

우선 찬식이가 흥빈이를 만난 후의 상태를 고려하지 않았을 때 찬식이의 취향을 생각해보자.

찬식이가 흥빈이를 만나기 전에 젤리를 샀을 때 최종 상태가 [3, 4, 2, 1]이라면, X개의 젤리를 산 후의 상태는 [3, 4, 2, 1]에서 X보다 작은 수만 취한 (순서는 바뀌지 않는다) 수열이 된다.

2, 3, 4번 젤리에 대해서 각 젤리를 넣은 후의 상태는 각각 [2, 1], [3, 2, 1], [3, 4, 2, 1]이다. 해당 젤리 위쪽에 있는 다른 모든 젤리들은 모두 그 젤리보다 맛있으나, 해당 젤리 바로 아래쪽에 있는 젤리는 그 젤리보다 맛없다. 따라서 우리는 4개의 정보 (2>1, 3>2, 3>4, 4>2)를 얻을 수 있다.

한편 찬식이의 취향별로, 찬식이가 흥빈이를 만난 후에 젤리를 샀을 때 가장 위에 있는 젤리의 번호를 길이 N의 수열 A[i]로 나타내면 A[1]=N이고, 2 이상의 i에 대해서 A[i]=A[i-1] or N+1-i 이다. 또, A에서 나타나는 수들을 내림차순으로 정렬한 수열 B[i]를 만들면, 수열 A와 수열 B는 일대일 대응이 된다.

우리는 “찬식이가 흥빈이를 만난 후에 젤리를 살 때 젤리 보관함의 가장 위쪽에 있는 젤리의 번호” S를 마지막으로 하는 수열 B를 잡은 다음 그 수열 B를 만들 수 있는 취향의 수를 구한 후, 그 경우의 수들을 전부 더하는 방식으로 답을 구할 것이다.

알고 있는 정보가 K개일 경우 취향의 수는  $2^{\frac{N(N-1)}{2} - k}$  이며, 새로 정보를 K'개 더 알았다면 취향의 수는 항상  $2^{K'} 배 줄어든다.$

우선 S-1, S-2, …, 1번 젤리가 S번 젤리보다 맛없다는 정보는 확실하게 알 수 있으므로 이는 이미 알고 있는 정보 목록에 추가 한다. 그 이외의 정보들은 모두 모르는 정보이다.

모르는 정보의 수를  $U$ 라고 하자. 이제, 우리는 동적계획법을 이용하는데, 1차원 DP 테이블  $D[i]$ 를 잡아서 아래와 같이 정의하자.

- $D[i]$  :  $i$ 로 끝나는 모든 수열  $B$ 를 잡아서 각  $B$ 에 대해서 얻을 수 있는 새로운 정보의 수  $C[1], \dots, C[x]$ 를 구했을 때,  $2^{U-C[x]}$ 들의 합 (단,  $i$ 가  $i-1, i-2, \dots, 1$ 보다 맛있다는 정보는 무시)

$D[i]$ 를 구하는 점화식은  $D[N] = 2^U, D[i] = \sum_{j>i} D[j] \times 2^{-\delta[j][i]}$  ( $i < N$ ) 이다. 여기서  $\delta[j][i]$ 는  $j > j-1, j > j-2, \dots, j > i+1, j < i$

중에서 새로운 정보의 수를 의미하며, 새로운 정보가 이미 알고 있는 정보와 모순되면  $\delta[j][i] = \infty$  으로 간주하여  $D[i]$ 에 더해주지 않는다.  $a > b \geq c > d$ 일 때  $\delta[a][b]$ 에서 세는 새로운 정보와  $\delta[c][d]$ 에서 세는 새로운 정보는 중복되거나 모순되지 않기 때문에 위 식과 같은 방식으로  $D[i]$ 를 구하면 정확하게 구할 수 있다. 문제의 정답은  $D[S]$ 이다.

$\delta[j][i]$ 를 구해놓은 상태라면  $D[i]$ 를 구하는 것은  $O(N^2)$  시간복잡도에 가능하다.

$\delta[j][i]$ 를 구할 때, 일반적인 방법으로 구하면 시간복잡도가  $O(N^3)$ 이지만,  $\delta[j][j-1], \dots, \delta[j][1]$ 을 순서대로 구하면  $O(N^2)$  시간복잡도에 가능하다. 이는  $j > j-1, j > j-2, \dots, j > i+2$  중에서 새로운 정보의 수를 구했다면  $j > i+1$ 와  $j < i$ 가 새로운 정보인지, 그리고 모순이 되는지만 보면 되기 때문이다.

경우의 수를 구했다면, 역추적은 쉽게 할 수 있다. 한편, 앞서 말한 것과는 달리 실제로는 나머지를 구하기 때문에  $2^{-x}$  ( $\equiv (5 \times 10^8 + 4)^x \pmod{10^9 + 7}$ )을 따로 전처리로 구해놓아야 한다. 이때  $\delta[j][i] < N$ 이므로 값을 배열로 저장해도 무방하다.

## 분류

동적계획법

## K. King of Chairs

우선, 궁녀가 다른 궁녀의 머리를 보는 경우의 수(inversion)의 기댓값을 구하자.

각 궁녀의 키에 해당하는 확률변수  $A_1, A_2, A_3, \dots, A_N$ 에 대해서 inversion 수의 기댓값은  $\sum_{i,j} E(A_i > A_j)$ 이다.

$E(A_i > A_j)$  =  $1/4 [(S_i > S_j) + (S_i > H_j) + (H_i > S_j) + (H_i > H_j)]$ 이다.

모든  $i, j$ 에 대해서  $E(A_i > A_j)$ 의 값을  $1/2$  이상으로 만들 수 있다.  $S_i < S_j$ 일 때  $i$ 를  $j$  앞에 놓으면  $i$ 가 앉으면  $j$ 가 서있든 앉아있든  $j$ 가  $i$ 를 볼 수 있으므로  $E(A_i < A_j)$ 의 값이  $1/2$  이상이다.

만약  $S_i = S_j$ 일 때는  $E_i < E_j$ 일 때  $i$ 를  $j$  앞에 놓으면  $E(A_i < A_j)$ 의 값이  $1/2$  이상이다.

따라서,  $S_i$ 가 작은 순으로,  $S_i$ 가 같으면  $E_i$ 가 작은 순으로 정렬하면 기댓값이 최대가 된다.  $N$ 명의 궁녀들을 정렬한 후 기댓값을 구하는 것은  $O(N^2)$  또는  $O(N \log N)$ 으로 할 수 있다.

## 분류

수학, 확률, 그리디

## L. List of Unique Numbers

부분수열  $\{A_l, A_{l+1}, \dots, A_r\}$ 에서 같은 수가 여러 번 등장한다면  $\{A_l, A_{l+1}, \dots, A_{r+x}\}$ 에서도 같은 수가 여러 번 등장한다. 한편, 부분수열  $\{A_l, A_{l+1}, \dots, A_r\}$ 에서 같은 수가 여러 번 등장하지 않는다면  $\{A_l, A_{l+1}, \dots, A_{r-x}\}$ 에서도 같은 수가 여러 번 등장하지 않는다. 이 성질을 이용하여, 모든  $l$ 의 값에 대해서 같은 수가 여러 번 등장하지 않는  $r$ 의 최댓값을 구하여  $r-l+1$ 의 합을 구하면 답을 구할 수 있다.

부분수열  $\{A_l, A_{l+1}, \dots, A_r\}$ 에서 같은 수가 여러 번 등장하지 않는다면  $\{A_{l+1}, \dots, A_r\}$ 에서도 같은 수가 여러 번 등장하지 않는다. 따라서  $l$ 이 증가하면  $r$ 의 최댓값은 변하지 않거나 증가한다.  $l=1$ 일 때 같은 수가 2번 나올 때까지 수를 계속 추가하면,  $r$ 은 ( $l=1$ 일 때의  $r$ 의 최댓값) + 1이다. 여기서  $A_1$ 을 없애준 후 부분수열에서 같은 수가 여러 번 등장하는지 확인하고,  $r$ 의 최댓값을 구하고,  $A_2$ 를 없애고,  $r$ 의 최댓값을 구하고, ….

숫자별 개수를 배열로 따로 넣어주어서 중복을 확인한다면  $O(N)$ 의 복잡도로 답을 구할 수 있다.

## 분류

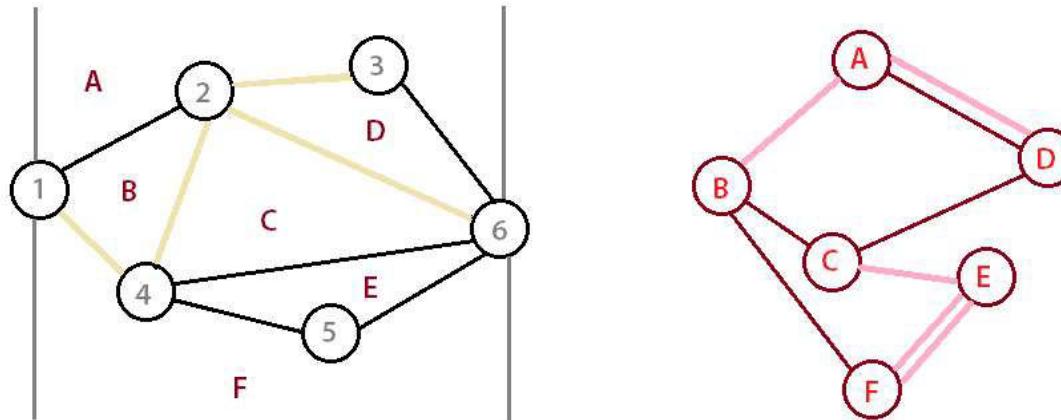
단순 구현

# M. Masonry Bridge

우선, 1번 섬과 N번 섬이 연결되는 최소 시간은 1번 섬과 N번 섬 사이의 최단거리와 같다. 이는 다익스트라 알고리즘을 이용하여 구할 수 있다.

1번 섬과 N번 섬이 연결되는 최대 시간을 구하려면 상당히 복잡한 과정이 필요하다.

1번 섬과 N번 섬이 각각 왼쪽 끝과 오른쪽 끝에 있고, 간선들이 교차하지 않으므로, 아래 그림과 같이 영역들로 이루어진 새로운 그래프(듀얼 그래프, dual graph)를 만들 수 있다.



기존 그래프에서 간선은 듀얼 그래프에서는 “기존 그래프의 간선이 맞대고 있는 두 면의 번호”를 연결하는 간선이 된다. 따라서 듀얼 그래프에서는 중복 간선과 셀프 간선이 나올 수 있다. 모든 면들은 선분들을 통해 맞닿아 있으므로 듀얼 그래프는 항상 연결 그래프이다.

1번 섬과 N번 섬이 연결되기 바로 직전에는 1번 섬과 N번 섬이 연결되어 있지 않다. 기존 그래프에서 1번 섬과 N번 섬이 연결되지 않을 필요충분조건은 듀얼 그래프에서 A번 정점과 F번 정점 사이를 “기존 그래프에서 제거된 간선에 대응되는 간선”들을 이

용하여 드나들 수 있는 것과 같다.

따라서, 1번 섬과 N번 섬이 연결되기 바로 직전의 상태는 둘째 그래프에서 “간선 하나를 더 추가하여 A번 정점과 F번 정점을 연결할 수 있는 상태”와 같다.

즉, 1번 섬과 N번 섬이 연결되는 최대 시간은 (전체 다리의 건설시간의 합) - (둘째 그래프에서 ‘간선 하나를 더 추가하여 A번 정점과 F번 정점을 연결할 수 있는 상태’를 만들 때, 연결해야 할 간선들의 건설시간의 합의 최솟값)이다.

전자는 쉽게 구할 수 있다. 후자는 둘째 그래프에서 A번 정점에서 시작하는 최단경로와 F번 정점에서 시작하는 최단경로를 다익스트라 알고리즘으로 구한 다음, 추가할 간선 P-Q (Q-P도 계산해주어야 한다.)를 선택하여 A-P간의 최단경로와 F-Q간의 최단경로의 합의 최솟값을 구하면 된다.

둘째 그래프를 구성하는 것은 각 간선 P-Q에 대해서 간선 P→Q와 Q→P를 만든 후, 2M개의 간선들을 노드로 하는 그래프를 만드는데, 기존 그래프에서 노드 Q를 시작점으로 각도 정렬을 했을 때 노드 P의 다음 점이 노드 R인 경우 (간선 P→Q)에서 (간선 Q→R)로 가는 방향성 간선을 추가하는 식으로 구성한다. 이렇게 하면 몇 개의 사이클이 생기는데, 각각의 사이클이 하나의 면을 의미하고 사이클 하나를 이루는 노드(기존 그래프에서 간선)들은 각 면을 구성하는 간선이 된다. 다만, 이런 방식대로 하면 A번 면과 F번 면이 한 사이클에 있으므로 이는 따로 처리해야 한다.

평면 그래프에서 간선의 수의 상한은  $3 \times (\text{노드의 수})$ 이므로  $M \leq 150,000$ 으로 보아도 무방하다. 둘째 그래프를 구성하는 시간 복잡도는  $O(M \log M)$ 이며 오일러의 정리에 따라 둘째 그래프의 노드의 수는  $3+M-N$ 이므로 둘째 그래프에서의 다익스트라 알고리즘은 시간복잡도가  $O(M \log M)$ 이므로 총 시간복잡도는  $O(M \log M)$ 이다.

## 분류

기하, 그래프