포탈

> ALPS 2020년 7월 내부대회 L번
> 출제자: 이승준

풀이

아이디어 : BFS + bit DP

시작 지점, 탈출 지점, 포탈 입구, 포탈 출구를 각각 s, e, p, q라 하고, 스위치들의 개수를 k라고 할 때,
각 스위치들을 a_1, ... , a_k라고 부르겠습니다.

시작 지점 s에서 탈출 지점 e까지 가는 경로의 종류는 2가지로 나눌 수 있습니다. 하나는 포탈을 사용하지 않는 경로이고, 다른 하나는 포탈을 사용하는 경로입니다. 두 경우의 최단거리 중 더 작은 것이 정답이 될 것입니다. (두 경우 모두 경로가 없을 경우 -1)

1. 포탈을 사용하지 않는 경로의 최단거리(시간)은 s에서 e까지 BFS를 통해 구할 수 있습니다.
2. 문제의 핵심은 포탈을 사용하는 경로의 최단거리를 구하는 것입니다. 문제의 조건에서 포탈을 사용하기 위해서는 그 전에 먼저 모든 스위치들을 적어도 한 번씩 방문해야 합니다. 결국 이 문제는 "k개의 스위치들을 어떠한 순서로 방문하는 것이 가장 빠를까?"로 생각할 수 있습니다.

이 문제를 해결하기 위해 우선 각 스위치들 간의 최단거리, 시작 지점과 스위치들간의 최단거리, 스위치들과 포탈 입구 간의 최단거리, 그리고 포탈 출구와 탈출 지점까지의 최단거리를 1번 경우처럼 BFS로 미리 구해 놓습니다. (지점 a와 b 사이의 최단거리를 $w(a, b)$로 표기하겠습니다) 여기서 스위치들의 쌍은 $_kC_2$개가 있고, 시작 지점과 스위치들, 스위치들과 포탈 입구의 쌍은 각각 k개, 포탈 출구와 탈출 지점의 쌍 1개가 있습니다. 그리고 각 쌍에 대해 최단거리를 구하는 데 BFS를 사용하므로 이 과정의 시간복잡도는 $O((_kC_2 + 2k + 1) \times n^2)$ = $O(k^2 \times n^2)$ 입니다.

본격적으로 문제를 해결하기 위해 간단한 접근부터 시도합니다. 단순하게 생각하면 k개의 스위치들을 방문하는$k!$ 가지의 경우의 수를 전부 확인할 수 있습니다. 각 순열에 대해 시작 지점부터 순열의 첫 번째 스위치를 방문한 뒤, 순열의 스위치들을 차례로 방문하고, 마지막으로 포탈 입구를 방문한 뒤 포탈의 출구에서 탈출 지점을 방문하면 됩니다. 그러나 문제의 제한에서 $k \le 15$ 이고, 따라서 확인해야 하는 경우의 수가 최대 $15!$ 개이기 때문에 이러한 방법은 시간 제한에 걸리게 됩니다.

이 때 $k!$ 개의 경우의 수를 모두 확인하는 것은 중복되는 과정이 필요하다는 것을 알 수 있습니다. 예를 들어, 순열의 i번째 원소를 b_i라고 할 때, $b_1 = 1$, $b_2 = 2$를 선택한 후 {3, 4, ... , k}의 숫자들을 순열로 나열하는 것과, $b_1 = 2$, $b_2 = 1$ 을 선택한 후 {3, 4, ... , k}의 숫자들을 순열로 나열하는 것에서 "{3, 4, ... , k}의 숫자들을 순열로 나열하는" 과정이 중복됩니다. 여기서 이 문제를 DP로 풀어야겠다는 착상을 얻을 수 있습니다. 우리는 각 순열에 대응하는 스위치 방문 경로의 최단거리를 구해야 하기 때문에, DP테이블을 "마지막에 방문한 스위치"와 "앞으로 방문해야 할 스위치들의 집합" 두 가지 요소로 정의해야겠다는 생각을 할 수 있습니다.

즉, DP 테이블과 재귀식은 다음과 같습니다.

$dp[A][\text{last}]$ = 방문한 스위치들의 집합이 A이고 마지막에 방문한 스위치가 last 경로들 중 최단 경로의 거리

$$dp[A][last] = \begin{cases} \min_{\forall\, a_i\ in\ A-\{\text{last}\}} (dp[A-\{last\}][a_i] + w(a_i, last)) & 방문한\ 스위치가\ 1\ 개\ 이상 \\ w(s, last) & 방문한\ 스위치가\ 0\ 개\ (dp의\ base\ case) \end{cases}$$

다시 말해, 방문한 스위치들이 A이고, 마지막으로 방문한 스위치가 last인 경로들 중 최단경로의 거리 $dp[A][\text{last}]$는 "(방문한 스위치들이 $A-\{\text{last}\}$고 마지막에 방문한 스위치가 $A-\{\text{last}\}$에 들어있는 어떤 스위치 a_i인 경로들 중 최단거리) + (그 a_i와 last간의 최단거리)" 중 가장 작은 것입니다. 그리고 언제나 시작 지점 s에서 출발하므로 방문한 스위치가 0개인 base case에서 $w(s, \text{last})$를 리턴합니다.

이렇게 dp테이블을 정의할 경우 처음 재귀 함수를 $\text{dp}[\text{존재하는 모든 스위치들}][\text{포탈의 입구}]$에 대해 호출하면 됩니다. 재귀 호출의 방향과 스위치를 방문하는 방향을 같게 하고 싶으면 dp 테이블 정의를 약간 바꿔 주면 됩니다. 이 풀이에서는 재귀 호출의 방향이 포탈의 입구로부터 s로 나아갑니다.

이 dp의 시간복잡도를 계산해 보면, 크기가 c인 A각각에 대해 last는 c가지가 가능하고, 각 (A, last) 쌍에 대해 가능한 모든 $a_i \in A - \{\text{last}\}$를 확인해야 하기 때문에 $c-1$가지를 확인해야 합니다. 크기가 c인 A의 개수는 ${}_kC_c$이므로 크기 c에 대해 $c \times (c-1) \times {}_kC_c$만큼의 재귀호출이 일어나고, 이에 전체 dp 수행시간은 아래와 같습니다.

$$O\left(\sum_{c=0}^{k} c*(c-1)*\binom{k}{c}\right) = O(\sum_{c=0}^{k} c*c*\binom{k}{c}) = O\left(k(k+1)2^{k-2} + 2k^2 - k\right) = O(k^2 * 2^k)$$

코드를 구현할 때 c개만 확인하는 것이 아니라 k개 전부를 순회하며 가능한 스위치를 찾는 방식으로 구현해도 시간복잡도에는 영향이 없습니다.

따라서, 이 풀이의 전체 수행시간은 $O(k^2n^2 + k^2 2^k)$ = $O(k^2(n^2 + 2^k))$입니다.

세줄요약

---

1. s와 e간, s와 모든 스위치간, 모든 스위치 사이, 모든 스위치와 포탈 입구간, 포탈 출구와 e간 최단거리를 BFS로 구함
2. "방문한 스위치 집합"과 "그 집합에서 마지막으로 방문한 스위치" 두 가지 요소로 dp테이블을 정의, 위의 재귀식을 구현
3. w(s, e)와 $\text{dp}[\text{존재하는 모든 스위치들}][\text{포탈의 입구}]$중 작은 것이 정답
4. 둘 다 경로가 없을 경우에는 -1

구현

자료구조는 인접 리스트와 인접 행렬 어느 쪽이든 상관 없습니다. dp 테이블의 경우 각 집합A를 표현하기 위해 비트 표현을 사용합니다. 예를 들어, 방문한 스위치의 집합이 $\{a_1, a_6, a_{12}\}$일 경우 $0000\ 1000\ 0010\ 0001_{(2)}$로 표현합니다.

가능한 모든 A의 개수가 $2^{16} - 1$ = 65535이므로 dp 테이블의 크기는 $65535 \times 15 + \alpha \approx 1,000,000$로 전역변수로 선언할 경우 c++기준에서 문제없이 실행됩니다. 비트로 집합 A에서 특정 원소가 있는지 확인하거나 특정 원소를 제거하는 과정 등은 비트마스크 연산을 통해 처리하면 됩니다.