코드포스 10주차 문제 풀이

문제 난이도

	문제 번호	문제 이름	난이도
A	28131	K-지폐	G1
B	1884	고속도로	G1
C	25619	자취방 정하기	G2
D	7140	데이터 만들기 1	G4
E	31566	힘세고 강한 아침	G3
F	13168	내일로 여행	G3
G	1162	도로포장	P5
H	1602	도망자 원숭이	P3

A. K-지폐

다이나믹 프로그래밍, 그래프 이론, 다익스트라, 최단 경로

A. K-지폐

- **다익스트라 알고리즘 변형:**
  - 우선순위 큐를 사용하여 최단 경로를 찾습니다.
  - 초기 상태로 시작 도시 S에서의 거리를 0으로 설정하고 큐에 추가합니다.
  - 큐에서 도시와 현재 거리를 추출하여 인접한 도시로 이동합니다. 이동할 때마다 거리 합을 K로 나눈 나머지로 처리하여 그 때의 최소거리를 갱신합니다.
  - Dist[remainder][city] : K로 나눈 나머지 remainder, 도시 city 일 때의 최단 거리
  - Dist[0][N]이 INF이면 IMPOSSIBLE, 아니면 출력합니다.

- **시간 복잡도:** $O((NK + M)\log(NK))$

B. 고속도로

다이나믹 프로그래밍, 그래프 이론, 다익스트라, 최단 경로

B. 고속도로

- **다익스트라 알고리즘의 변형:**
  - 현재 도시에서 연결된 모든 도로를 검사합니다.
  - 도로를 통해 다음 도시에 도달했을 때, 예산을 초과하지 않으면 해당 경로를 우선순위 큐에 추가합니다.
  - 예산 내에 도달한 도시의 최소 경로 길이를 갱신합니다.
  - Dist[cost][city] : city에 cost비용으로 도달할 때의 최소 경로
  - 목적지에 도달할 때(city가 N일 때)마다 경로 길이를 확인하여 ans 값을 갱신합니다.
  - Ans값이 갱신되었으면 답을 출력하고, 아니면 -1출력합니다.

- **시간 복잡도:** $O(Elog(NK))$

C. 자취방 정하기

수학, 그래프 이론, 다익스트라, 최단 경로, 확률론, 기댓값의 선형성

C. 자취방 정하기

- ✓ **데이터 입력 및 그래프 구성**:
  - ✓ 정점 N, 간선 M의 개수를 입력 받고, 각 간선 정보를 입력 받아 그래프를 구성합니다.
  - ✓ 각 간선은 두 가지 교통량 a와 b중 하나를 가집니다. 이를 합하여 간선의 가중치 w로 설정합니다.

- ✓ **음수 간선 여부 확인 (BFS) (O(N + M))**:
  - ✓ BFS를 통해 그래프에 음수 가중치 간선이 있는지 확인합니다. 음수 간선이 있으면 경로 탐색 방식이 달라지기 때문에 이 단계가 필요합니다.

- ✓ **다익스트라 알고리즘 (O((N + M)logN))**:
  - ✓ 음수 간선이 없는 경우, 다익스트라 알고리즘을 사용하여 각 정점까지의 최단 거리를 계산합니다.
  - ✓ 최단 거리 DIST [i] 가 $2 \times T$이하인 정점을 결과로 저장합니다.
  - ✓ 왜 $2 \times T$인가 하면, 간선의 가중치가 $a + b$로 주어졌기 때문에 최악의 경우 T의 두 배로 고려합니다.

- ✓ **결과 출력**:
  - ✓ 조건을 만족하는 자취방의 개수를 출력하고, 그 자취방의 정점 번호를 오름차순으로 출력합니다.

D. 데이터 만들기 1

그래프 이론, 애드 혹, 해 구성하기, 다익스트라, 최단 경로, 플로이드-워셜

D. 데이터 만들기 1

- ✓ **조건들:**
  - ✓ ModifiedDijkstra: AC, FloydWarshall: TLE, 데이터는 최대 107개의 정수로 구성
- ✓ **ModifiedDijkstra**:
  - ✓ Dijkstra 알고리즘은 우선순위 큐를 사용하여 작동합니다. 실제로 처리해야 할 간선을 주지 않으면 정점이 서로 연결되어 있지 않기 때문에, 복잡도가 낮고 TLE를 발생시키지 않습니다.
- ✓ **FloydWarshall**:
  - ✓ Floyd-Warshall 알고리즘은 모든 정점 쌍 사이의 최단 경로를 구하기 위해$O(V^3)$의 시간 복잡도를 가지며, 100개 보다 정점이 많은 그래프에서 이 알고리즘은 1,000,000번의 반복을 넘기 때문에 TLE를 발생시킵니다.
- ✓ 따라서 정점의 개수를 100보다 많이, 간선의 개수는 주지 않고 쿼리를 주면 정답입니다.

E. 힘세고 강한 아침

그래프 이론, 최단 경로, 플로이드-워셜

E. 힘세고 강한 아침

- **Floyd-Warshall 알고리즘의 변형:**
  - 플로이드 워셜 알고리즘은 3중 반복문입니다. (경유지, 출발지, 목적지)
  - 이 문제를 풀기 위해 4중 반복문을 사용합니다. (제외할 곳, 경유지, 출발지, 목적지)
  - Dist[except][start][end]: except언어를 제외하고, start언어에서 시작하여 end언어로 가는 최단거리
  - $O(N^4)$이지만 시간안에 풀 수 있습니다.

- **쿼리 처리:**
  - 각 질문에 대해 s에서 e로 k를 거치지 않고 번역이 가능한지 확인하고, 가능하면 최소 비용을 출력합니다.

F. 내일로 여행

구현, 자료 구조, 그래프 이론, 해시를 사용한 집합과 맵,
최단 경로, 플로이드-워셜

F. 내일로 여행

- **초기 비용 설정**:
  - DIST 배열과 discounted 배열을 초기화(INF)합니다. 이 배열들은 각각 내일로 티켓을 사용하지 않았을 때와 사용했을 때의 최소 비용을 저장합니다.
  - 교통수단 정보를 바탕으로 비용을 DIST와 discounted 배열에 저장합니다. 내일로 티켓의 혜택이 적용되는 교통수단에 대해 할인된 비용을 설정합니다.

- **Floyd-Warshall 알고리즘**:
  - 모든 도시 간의 최단 경로를 계산하기 위해 Floyd-Warshall 알고리즘을 사용합니다. 이는 각 도시 간 최단 경로를 구하는 알고리즘으로, $O(N^3)$의 시간 복잡도를 가집니다.
  - DIST와 discounted 배열을 각각 업데이트하여 최단 경로를 계산합니다.

- **여행 비용 계산:**
  - 내일로 티켓을 사용하지 않았을 때의 총 비용과 내일로 티켓을 사용했을 때의 총 비용을 계산합니다.
  - 각 여행할 도시 간의 이동 비용을 더하여 총 비용을 구합니다.
  - 내일로 티켓을 사는 것이 더 저렴한 경우 "Yes", 그렇지 않은 경우 "No"를 출력합니다.

G. 도로포장

다이나믹 프로그래밍, 그래프 이론, 다익스트라, 최단 경로

G. 도로 포장

- ✓ **거리 배열 초기화**:
  - ✓ DIST[i][j]는 i개의 도로를 포장했을 때 도시 j까지의 최소 거리입니다.
  - ✓ 모든 거리 값을 무한대로 초기화하고, 서울(도시 1)의 거리 값은 0으로 초기화합니다.

- ✓ **Dijkstra 알고리즘**:
  - ✓ 우선순위 큐를 사용하여 현재 도시까지의 거리와 포장된 도로의 수를 고려하여 최단 경로를 구합니다.
  - ✓ 큐의 원소는 (현재 거리, 현재 도시, 포장한 도로의 수)로 구성됩니다.
  - ✓ 도로 포장 없이 해당 도시까지의 최단 거리와 포장한 도로의 수를 이용하여 큐에 추가합니다.
  - ✓ 현재의 도로를 포장할 수 있는 경우(포장한 도로의 수 + 1 <= K), 포장 후의 상태를 고려하여 거리를 갱신합니다(현재의 도로 거리 무시).

- ✓ 시간 복잡도: $O((N+M)logN)$

H. 도망자 원숭이

그래프 이론, 정렬, 최단 경로, 플로이드-워셜

H. 도망자 원숭이

- ✓ **도로 정보 처리**:
  - ✓ 각 도로에 대해 양방향으로 저장하고, 두 도시 간의 도로의 통행 시간(w)을 DIST 배열에 저장합니다.
  - ✓ 도로가 연결된 두 도시에서 멍멍이가 괴롭힐 수 있는 최대 시간을 monkey 배열에 업데이트합니다.

- ✓ **Floyd-Warshall 알고리즘**:
  - ✓ DIST[i][j]는 도시 i 에서 도시 j 까지의 최단 경로를 나타내며, monkey[i][j]는 해당 경로를 통과할 때 멍멍이가 괴롭힐 수 있는 최대 시간을 저장합니다.
  - ✓ 멍멍이가 괴롭힐 수 있는 시간을 기준으로 도시를 **정렬**한 후, **괴롭히는 시간이 작은 도시 순서대로 경유(첫 번째 루프)**하여 알고리즘을 수행합니다.

- ✓ **쿼리 처리**:
  - ✓ 각 쿼리에 대해 출발 도시와 도착 도시를 입력 받고, 최단 경로의 길이(DIST[i][j])와 경로를 통과할 때 멍멍이가 괴롭힐 수 있는 최대 시간(monkey[i][j])을 합한 값을 출력합니다.
  - ✓ 경로가 없으면 -1을 출력합니다.

- ✓ 시간 복잡도: $O(N^3)$