다익스트라 알고리즘
데이크스트라
Dijkstra

Algo 알고😆
진홍엽

목차

- 데이크스트라 알고리즘이란?
- 눈으로 살펴보기
- 데이크스트라 알고리즘 [원조]
- 개선된 데이크스트라 알고리즘 (HEAP)
- 데이크스트라 알고리즘!!

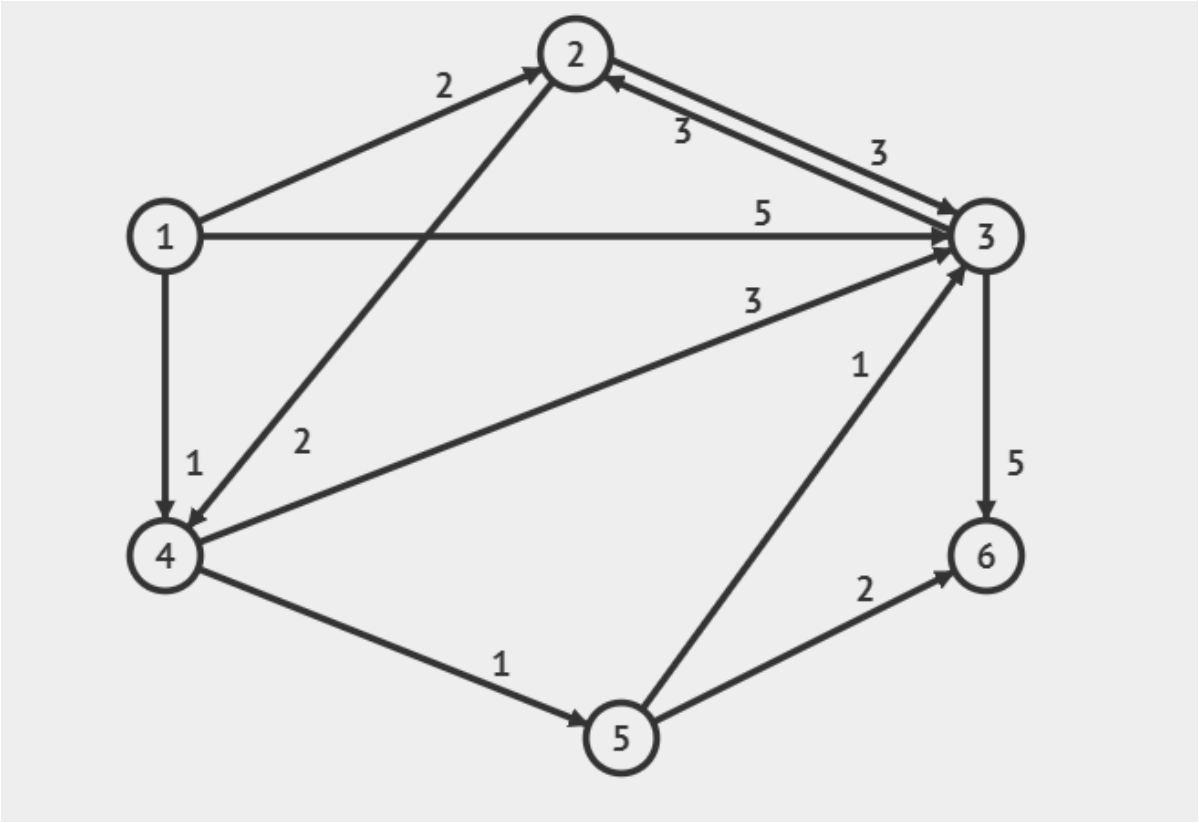
데이크스트라 알고리즘이란?

- 최단 경로 알고리즘
  - 특정 노드에서 모든 다른 노드로 가는 최단경로 계산
  - 음의 간선이 없을 경우만 사용 가능
  - 매 상황에서 가장 비용이 적은 노드를 선택해 임의의 과정 반복

눈으로 살펴보기

https://visualgo.net/en/sssp

```
7 11
1 2 2
1 3 5
1 4 1
2 3 3
2 4 2
3 2 3
3 6 5
4 3 3
4 5 1
5 3 1
5 6 2
```



데이크스트라 알고리즘 [원조]

다익스트라 알고리즘: 간단한 구현 방법 (Python)

```python
import sys
input = sys.stdin.readline
INF = int(1e9) # 무한을 의미하는 값으로 10억을 설정

# 노드의 개수, 간선의 개수를 입력받기
n, m = map(int, input().split())
# 시작 노드 번호를 입력받기
start = int(input())
# 각 노드에 연결되어 있는 노드에 대한 정보를 담는 리스트를 만들기
graph = [[] for i in range(n + 1)]
# 방문한 적이 있는지 체크하는 목적의 리스트를 만들기
visited = [False] * (n + 1)
# 최단 거리 테이블을 모두 무한으로 초기화
distance = [INF] * (n + 1)

# 모든 간선 정보를 입력받기
for _ in range(m):
    a, b, c = map(int, input().split())
    # a번 노드에서 b번 노드로 가는 비용이 c라는 의미
    graph[a].append((b, c))

# 방문하지 않은 노드 중에서, 가장 최단 거리가 짧은 노드의 번호를 반환
def get_smallest_node():
    min_value = INF
    index = 0 # 가장 최단 거리가 짧은 노드(인덱스)
    for i in range(1, n + 1):
        if distance[i] < min_value and not visited[i]:
            min_value = distance[i]
            index = i
    return index
```

```python
def dijkstra(start):
    # 시작 노드에 대해서 초기화
    distance[start] = 0
    visited[start] = True
    for j in graph[start]:
        distance[j[0]] = j[1]
    # 시작 노드를 제외한 전체 n - 1개의 노드에 대해 반복
    for i in range(n - 1):
        # 현재 최단 거리가 가장 짧은 노드를 꺼내서, 방문 처리
        now = get_smallest_node()
        visited[now] = True
        # 현재 노드와 연결된 다른 노드를 확인
        for j in graph[now]:
            cost = distance[now] + j[1]
            # 현재 노드를 거쳐서 다른 노드로 이동하는 거리가 더 짧은 경우
            if cost < distance[j[0]]:
                distance[j[0]] = cost

# 다익스트라 알고리즘을 수행
dijkstra(start)

# 모든 노드로 가기 위한 최단 거리를 출력
for i in range(1, n + 1):
    # 도달할 수 없는 경우, 무한(INFINITY)이라고 출력
    if distance[i] == INF:
        print("INFINITY")
    # 도달할 수 있는 경우 거리를 출력
    else:
        print(distance[i])
```

원형의 시간 복잡도

다익스트라 알고리즘: 간단한 구현 방법 성능 분석

- 총 $O(V)$번에 걸쳐서 최단 거리가 가장 짧은 노드를 매번 선형 탐색해야 합니다.
- 따라서 전체 시간 복잡도는 $O(V^2)$입니다.
- 일반적으로 코딩 테스트의 최단 경로 문제에서 전체 노드의 개수가 5,000개 이하라면 이 코드로 문제를 해결할 수 있습니다.
  - 하지만 노드의 개수가 10,000개를 넘어가는 문제라면 어떻게 해야 할까요?

개선된 데이크스트라 알고리즘 (HEAP)

다익스트라 알고리즘: 개선된 구현 방법 (Python)

```python
import heapq
import sys
input = sys.stdin.readline
INF = int(1e9) # 무한을 의미하는 값으로 10억을 설정

# 노드의 개수, 간선의 개수를 입력받기
n, m = map(int, input().split())
# 시작 노드 번호를 입력받기
start = int(input())
# 각 노드에 연결되어 있는 노드에 대한 정보를 담는 리스트를 만들기
graph = [[] for i in range(n + 1)]
# 최단 거리 테이블을 모두 무한으로 초기화
distance = [INF] * (n + 1)

# 모든 간선 정보를 입력받기
for _ in range(m):
    a, b, c = map(int, input().split())
    # a번 노드에서 b번 노드로 가는 비용이 c라는 의미
    graph[a].append((b, c))
```

```python
def dijkstra(start):
    q = []
    # 시작 노드로 가기 위한 최단 거리는 0으로 설정하여, 큐에 삽입
    heapq.heappush(q, (0, start))
    distance[start] = 0
    while q: # 큐가 비어있지 않다면
        # 가장 최단 거리가 짧은 노드에 대한 정보 꺼내기
        dist, now = heapq.heappop(q)
        # 현재 노드가 이미 처리된 적이 있는 노드라면 무시
        if distance[now] < dist:
            continue
        # 현재 노드와 연결된 다른 인접한 노드들을 확인
        for i in graph[now]:
            cost = dist + i[1]
            # 현재 노드를 거쳐서, 다른 노드로 이동하는 거리가 더 짧은 경우
            if cost < distance[i[0]]:
                distance[i[0]] = cost
                heapq.heappush(q, (cost, i[0]))

# 다익스트라 알고리즘을 수행
dijkstra(start)

# 모든 노드로 가기 위한 최단 거리를 출력
for i in range(1, n + 1):
    # 도달할 수 없는 경우, 무한(INFINITY)이라고 출력
    if distance[i] == INF:
        print("INFINITY")
    # 도달할 수 있는 경우 거리를 출력
    else:
        print(distance[i])
```

개선된 시간 복잡도

- 힙 자료구조를 이용하는 다익스트라 알고리즘의 시간 복잡도는 $\boldsymbol{O(ElogV)}$입니다.
- 노드를 하나씩 꺼내 검사하는 반복문(while문)은 노드의 개수 V 이상의 횟수로는 처리되지 않습니다.
  - 결과적으로 현재 우선순위 큐에서 꺼낸 노드와 연결된 다른 노드들을 확인하는 총횟수는 최대 간선의 개수(E)만큼 연산이 수행될 수 있습니다.
- 직관적으로 전체 과정은 E개의 원소를 우선순위 큐에 넣었다가 모두 빼내는 연산과 매우 유사합니다.
  - 시간 복잡도를 $O(ElogE)$로 판단할 수 있습니다.
  - 중복 간선을 포함하지 않는 경우에 이를 $O(ElogV)$로 정리할 수 있습니다.
    - $O(ElogE) \rightarrow O(ElogV^2) \rightarrow O(2ElogV) \rightarrow O(ElogV)$

$O(V^2)$ $O(ElogV)$

데이크스트라 알고리즘!!

- 최단 경로 알고리즘
  - 특정 노드에서 모든 다른 노드로 가는 최단경로 계산
  - 음의 간선이 없을 경우만 사용 가능
    - 음의 간선이 있을 경우, 방문 처리를 한 노드는 최단 거리라는 전제가 무너짐
  - 매 상황에서 가장 비용이 적은 노드를 선택해 임의의 과정 반복
    - 그리디 알고리즘
    - 매 상황에서 가장 적은? -> HEAP