Problem 787: Cheapest Flights Within K Stops

Problem Information

Difficulty: Medium
Acceptance Rate: 41.01%
Paid Only: No
Tags: Dynamic Programming, Depth-First Search, Breadth-First Search, Graph, Heap (Priority Queue), Shortest Path

Problem Description

There are `n` cities connected by some number of flights. You are given an array `flights` where `flights[i] = [fromi, toi, pricei]` indicates that there is a flight from city `fromi` to city `toi` with cost `pricei`.

You are also given three integers `src`, `dst`, and `k`, return _**the cheapest price** from _`src` _to_`dst` _with at most_`k` _stops._ If there is no such route, return __`-1`.

Example 1:

![](https://assets.leetcode.com/uploads/2022/03/18/cheapest-flights-within-k-stops-3drawio.png)

Input: n = 4, flights = [[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]], src = 0, dst = 3, k = 1 **Output:** 700 **Explanation:** The graph is shown above. The optimal path with at most 1 stop from city 0 to 3 is marked in red and has cost 100 + 600 = 700. Note that the path through cities [0,1,2,3] is cheaper but is invalid because it uses 2 stops.

Example 2:

![](https://assets.leetcode.com/uploads/2022/03/18/cheapest-flights-within-k-stops-1drawio.png)

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 1 **Output:** 200 **Explanation:** The graph is shown above. The optimal path with at most 1 stop from city 0 to 2 is marked in red and has cost 100 + 100 = 200.

Example 3:

![](https://assets.leetcode.com/uploads/2022/03/18/cheapest-flights-within-k-stops-2drawio.png)

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 0 **Output:** 500 **Explanation:** The graph is shown above. The optimal path with no stops from city 0 to 2 is marked in red and has cost 500.

Constraints:

* `2 <= n <= 100` * `0 <= flights.length <= (n * (n - 1) / 2)` * `flights[i].length == 3` * `0 <= fromi, toi < n` * `fromi != toi` * `1 <= pricei <= 104` * There will not be any multiple flights between two cities. * `0 <= src, dst, k < n` * `src != dst`

Code Snippets

C++:

```
class Solution {
public:
int findCheapestPrice(int n, vector<vector<int>>& flights, int src, int dst,
int k) {

}
};
```

Java:

```
class Solution {
public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k)
{

}
}
```

Python3:

```
class Solution:
def findCheapestPrice(self, n: int, flights: List[List[int]], src: int, dst:
```

```
int, k: int) -> int:
```