시작점에서 도착점까지의 가장 저렴한 가격을 계산하되, K개의 경유지 이내에 도착하는 가격을 리턴하라. 경로가 존재하지 않을 경우 -1 을 리턴한다.

Input:
n = 3, edges = [[0,1,100],[1,2,100],[0,2,500]]
src = 0, dst = 2, k = 1
Output: 200

1.dijkstra

```
import collections
import heapq

class Solution:
    def findCheapestPrice(self, n: int, flights: List[List[int]], src: int, dst: int, K: int) -> int:
        graph = collections.defaultdict(list)
        for u, v, w in flights:
            graph[u].append((v,w))
        q = [(0, src, K)]
        while len(q) > 0:
            time, node, count = heapq.heappop(q)
            if node == dst:
                return time
            if count >= 0:
                for v, w in graph[node]:
                    alt = time + w
                    heapq.heappush(q, (alt, v, count - 1))
        return -1
```