787 Cheapest flights within K stops

→ recibe flights = [from, to, price]

→ source and destination

→ cities que van

→ cheapest flight from src to dst with at most K stops

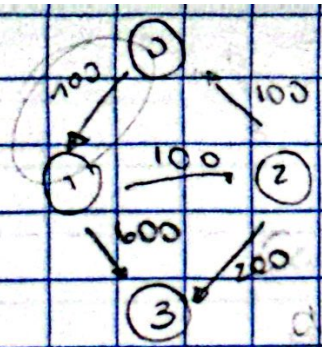
- problema de caminos mínimos (grafos)
  - ↳ como se quiere ver hasta K paradas se podría utilizar Floyd Warshall
    - ↳ en el algoritmo se guarda las distancias mínimas entre todas las ciudades (vértices)
    - ↳ los pesos de las aristas son el precio de los vuelos
  - * Problema: al actualizar la tabla, puede que no se actualice correctamente la distancia mínima por el límite K

OP1

- ↳ Bellman Ford modificado
  - → se recorren las aristas como mínimo
    - ↳ se guardan las distancias durante de las ciudades después de (K=0)
    - ↳ se itera K veces (max de imp) y se hace BF
      - → se hace una copia del costo por cada K iteración para saber los costos por cada max K.
      - → para cada f in flights

temp[to] = min(temp[to], dist[from] + price)

↳ o sea, si se puede llegar al from desde el src (distancia mínima), podemos llegar desde src → from → to, la nueva distancia del src al to sería el mínimo entre la distancia del source al to y la distancia del source al from más el precio del from al to.



src = 0 dst = 3 K = 1

K = 0

	0	1	2	3
	0	∞	∞	∞
	0	100	∞	∞

0 → 1 : min(∞, 100)

actualiza de 0 → 1 más distancia de 1 → 2 para dist mínima 0 → 2

K = 1

	0	100	∞	∞
	0	100	200	∞
	0	100	200	300

1 → 2 : min(∞, 100 + 100) (0 → 2) = 200

1 → 3 : min(∞, 100 + 600) (0 → 3) = 700

→ costo mínimo de 0 a 3 con 1 parada es 700.

op 2

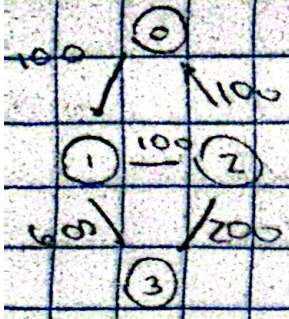
↳ DIJKSTRA con Min-Heap Modificado

↳ el min-heap o priority queue guía el camino mínimo del src al resto de ciudades

↳ cada nodo tendría destino, costo, número de paradas

↳ sería una idea parecida a la anterior, visito los nodos adyacentes al nodo top en el min-heap, agregarlos al min-heap; luego pop el nuevo min-top, visitar los nodos

src = 0 dst = 3 k = 2



Min-Heap

costo	to	stops
0	0	0

minStops

0	1	2	3
0	∞	∞	∞

- 0 → 1

costo = 100

to = 1

stops = -1

minHeap

100	3	2

→ dst = 3

y Z <= K+1

Min Heap

costo	to	stops
100	1	1

min stops

0	1	2	3
0	1	∞	∞

- 1 → 2 (0 → 2)

costo = 100 + 100 = 200

to = 2

stops = 2

- 1 → 3 (0 → 3)

costo = 100 + 600 = 700

to = 3

stops = 2

Min Heap

200	2	2

700	3	2

min stops

0	1	2	3
0	1	2	2

- 2 → 0 ✗
- 2 → 3 → costo = 200 + 200 → to = 3
  → stops = 3 ✗

Scanned with CamScanner