**787. Cheapest Flights Within K Stops**

**Algorithm -Dijksharts shortest path**

class city

{

int source;

int destfromsource;

int costfromsource;

city(int source,int destfromsource,int costfromsource)

{

this.source=source;

this.destfromsource=destfromsource;

this.costfromsource=costfromsource;

}

}

class pair

{

int dest;

int cost;

pair(int dest,int cost)

{

this.dest=dest;

this.cost=cost;

}

}

class Solution {

public int findCheapestPrice(int n, int[][] flights, int src, int dst, int K)

{

ArrayList<ArrayList<pair>> graph=new ArrayList<>();

for(int i=0 ;i<n ;i++)

{

graph.add(new ArrayList<>());

}

for(int[] x :flights)

{

graph.get(x[0]).add(new pair(x[1],x[2]));

}

PriorityQueue<city> pq=new PriorityQueue<>((city c1,city c2)-> c2.costfromsource-c1.costfromsource);

pq.offer(new city(src,0,0));

while(!pq.isEmpty())

{

city temp=pq.poll();

if(temp.destfromsource==dst)

return temp.costfromsource;

if(temp.destfromsource>K)

continue;

List<pair> neighbour=graph.get(temp.source);

for(pair x : neighbour)

{

pq.offer(new city(x.dest,temp.destfromsource+1,temp.costfromsource+x.cost));

}

}

return -1;

}

}