* **Problem** : [Shortest Routes I](https://cses.fi/problemset/task/1671/)

Notice the 2 lines in the question :

Each flight is a one-way flight.(so directed graph)

You can assume that it is possible to travel from Syrjälä to all other cities.

* **Approach: Single src Shortest Path(Dijkstra)(Greedy)**

**(means from a src node to all other nodes).**

**Video :** [**09 Graph Theory:: Dijkstra's Algorithm with CSES 08 Shortest Routes I (1671)**](https://youtu.be/ditJWdFqoXk?t=1309)

**->(**We assume all nodes are reachable)

[Works for both directed and non directed graphs]

**=> Dijkstra says : If the smallest distance ‘x’ to a node ‘u’ from src amongst all distances to that node recorded until now in minHeap is SMALLER than the smallest distance to all other nodes from the src in minHeap ,then we GREEDILY LOCK ‘x’ as the smallest distance to ‘u’.**(this is why it might fail sometimes with neg edges and always fail detecting neg cycles)

* **Algo with implementation :**

-> We use a **Min Heap(priority queue)** , which stores (dist of a node from src,node id),

-> We also use a vector called **minDist** which records smallest dist to a node from src until a given instant.

-> We check distance of every node adjacent to current processed node and if for an adjacent node we encounter smaller distance than the one recorder in minDist , we update minDist and **add that dist. with node id to heap.**

-> **And on next iteration after processing previous node,minHeap gives the node with current smallest dist from src.**

-> **Continue while Heap is not empty:**

**-> Pop from min heap**, call that node ‘u’

So node with nearest dist from src node at this instance wud be selected.

(In 1st round off course src node is popped.And then the other ones according to their distances from src.)

//If min distance from src node to node 'u' recorded inside minDist

is less than what is popped from heap,means we already processed node 'u' earlier with less distance from src,so ignore this older entry which would have been pushed earlier.

E.g {4,10},{6,10},{8,10} ,are entries for a node 10 in minheap, first dist 8 wud have been added then 6,but we could not delete 8 from min heap ,same when 4 comes,so first {4,10} will be popped , at that time minDist[u] == dist\_u **and** node 10 will be processed ,not skipped and now all other instances of 10 lying down should be ignored by writing below condition.

**-> if(minDist[u] < dist\_u) continue;**

**-> For all nodes v adj to u :**

**Let dist\_u=dist of popped node from src**

**Let dist\_v=dist of adjacent node from src**

-> If we can get shorter dist to a node ‘v’ from ‘u’ than the best one recorded in midDist[v] then update it and add to minHeap

-> This line also prevents adding an already processed node(means a node taken out of heap once) back again in the heap

**-> If dist\_u+dist\_v < mindist[v]**

**Update mindist[v]=dist\_u+dist\_v**

**Add pair {mindist[v] , v} to Heap**

(Note that any pair e.g {5,2} (means current shortest dist to node 2 is 5)

is stored only once,means now if from some other node also(in future),the dist to 2 is 5,then that won’t be added again bcoz we dont want to PRINT ALL PATHS,we just want smallest distance.)

**-> Why we cant use BFS ?**

Ans) BFS works only when weight of each edge is 1.

a---5-----b

**\ /**

1 2

\ /

c

-> BFS returns answer as a->b, but right answer is a->c->b

**Note : If neg- edges present use DP(Bellman ford or Floyd Warshall).**

* **Code :** [**https://ideone.com/oDrk2F**](https://ideone.com/oDrk2F)