**Problem** : <https://cses.fi/problemset/task/1196/>

**Approach** :

-> We apply the same Dijkstra algorithm explained in “Shortest Routes” question .

-> Change is that Destination is already given,and **we need to find first ‘k’ shortest paths** and print them in ascending order, also we can visit **one city several times.**

-> So we use the same **minHeap** used earlier in normal Dijkstra algo.

-> For the **minDist** vector, we store the ‘k’ shortest paths to that node from src until now.

=> Like Normal Implementation,keep adding distances into the into the minHeap and minDist vector . In previous implementation,after processing a node,we used to ignore all entries lying below in minHeap for that node.Also we used to store only 1 optimal distance in minDist vector.

=> Here it’s a bit different.

1) We store **the best ‘k’ distances** to a node from src recorded until now in minDist vector.

(Means if a new entry with some unique distance arrives and it’s better than kth(worst) distance until now, we replace kth distance with new distance and again sort the vector.)

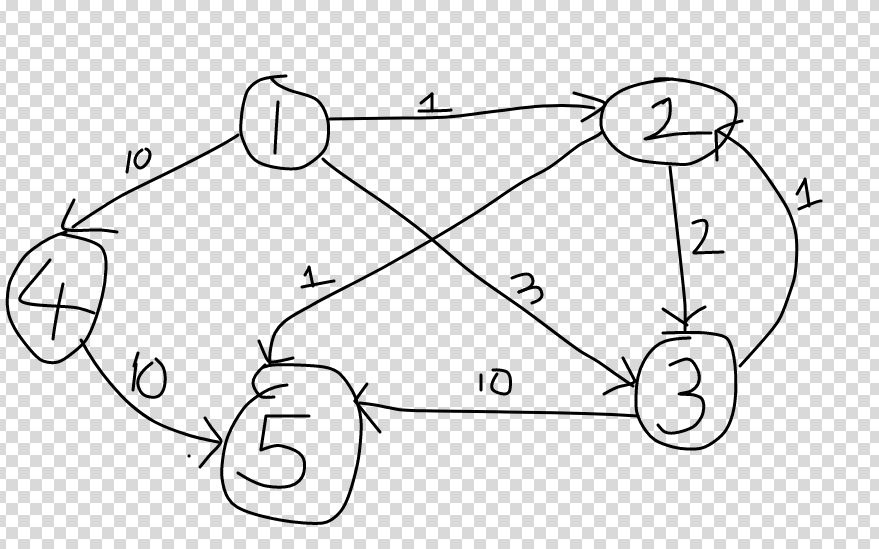
2) We might also need to **process a node multiple times** (given that same city can be visited several times) bcoz we need ‘k’ best distances to destination & visiting a city again might give another path lies in our ‘k’ best paths we are looking for.

(Imp point) As said can process a node every time an entry for it is popped off from Heap.But we process it **only if it’s distance from src is less than kth(worst) distance** for that node in minDist vector(as any distance bigger than kth distance can’t give any of ‘k’ best distances)

So a node might also be processed ‘k’ times! Why ?

Let’s assume ‘k’ distances to a node are added in minHeap.Now all of them are also processed when their turn came, but after processing them , in future,we encountered some better distances than kth path to that node , so all of them were again added and so need to be processed again.

-> When algo. Stops ‘k’ best paths cost to every node from src would be discovered , not just the destination(as we run the code until heap is not empty).



1st best path = 1->2->5

2nd best path = 1->2->3->2->5

3rd best path= 1->3->2->5

And so on…(so one city might need to be visited many times)

Run dijkstra on above algo and you will understand below code.

**Code :** <https://ideone.com/sBRmnU>