state s: <L-O-S-K>

1.4

Neighbours:

<O-L-S-K> 1.7

<S-O-L-K> 1.9

<K-O-S-L> 1.3

<L-S-O-K> 1.3

<L-K-S-O> 1.2

<L-O-K-S> 1.5

(i) As we can see above, <L-K-S-O>, which is one of initial state's neighbour, has lower cost of path 1.2 < 1.4. Therefore next step should choose the state <L-K-S-O>.

(ii) We stop until no neighbour of current state has a lower value cost. If there exist a neighbour that has lower cost path, we change the state into the neighbour one.

(iii) We cannot ensure our solution is global optimal. Given the example:

A 20

B 10 C 15

D 8 E 5

Suppose we are to find minimum value of the tree. When we at state A we compare the value B and C. And choose to continue search through B as B < C. We ignore the C state, but actually its child E has lowest value of all. We can only get D = 8 as local minimal.