

## MthSc 816 — Assignment #1 (Primal Journey)

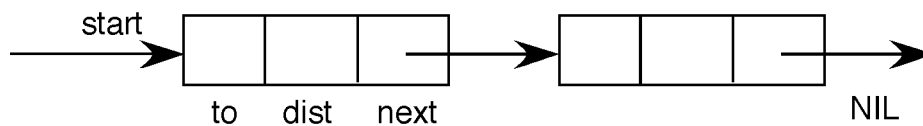
### Data Structures (for $G$ )

As read in each edge, create linked adjacency lists for every node  $v$ .

Adjacency list for  $v$  represented by a **start** pointer and a **linked list**:

**start**( $v$ ) = pointer to start of linked list for node  $v = 1, 2, \dots, n$

{**to**, **dist**, **next**} for each record of linked list



Space needed for these linked lists is  $O(n + 6m)$ , if  $G$  has  $m$  edges.

Initialize using **first**( $v$ ) = **NIL** for all  $v$ .

No need to have elements (nodes adjacent to  $v$ ) sorted in order.

### Input

Read in successive undirected edges  $(v,w)$ :

**add\_edge**( $v,w$ ), **add\_edge**( $w,v$ )

In **add\_edge**( $v,w$ ) function, insert  $w$  at the front of linked list for  $v$ . This saves extra space for keeping track of end of each linked list.

## Data Structures (for Search)

Keep three arrays, handled in a stack fashion

**top** = current position in the stacks

**path(k)** = the  $k$ -th node on the path

**current\_edge(k)** = pointer to next available edge out of **path(k)**

**cum\_length(k)** = cumulative length up to node **path(k)**

Space complexity is  $O(3p)$  if consider the first  $p$  primes up to prime  $P$ .

Checking for **IsPrime** may require additional space; if use a Boolean array then  $O(P)$  space.

## DFS

Modularize using routine **FindEdge** to find next admissible arc out of a node, given the current path length and starting at **current\_edge**.

Start at each node in turn as the root and do the DFS.

If there is an admissible arc out of the node, update the stack and advance to next node.

If not, retreat to previous node on path.

In either case, call **FindEdge**.

## Other Items

Proper initializations can avoid special cases.

If necessary, dimension node arrays as  $[0 .. n]$  to avoid subtracting one each time.

## Solutions

Problem #1: Total Mileage = 523

2 – 10 – 17 – 10 – 9 – 13 – 15 – 12 – 9 – 13 – 14 – 20 – 21 – 14 – 20 – 21 –  
22 – 23 – 19 – 23 – 18 – 5 – 18 – 19 – 23 – 18 – 19 – 23 – 19 – 7 – 6 – 1 – 9 –  
23 – 19 – 6 – 5 – 18 – 23 – 22 – 21 – 20 – 14 – 21 – 20 – 14 – 13 – 9 – 12 –  
15 – 13 – 9 – 12 – 11 – 16 – 15 – 13 – 14 – 20 – 8 – 3 – 4 – 3 – 1 – 4 – 2

Problem #2: No primal journey exists

```

[new_edge, new_length] = FindEdge(node, first, length)

// start exploring arcs emanating from node, beginning at first

new_edge = 0
new_length = -1

edge = first
while (edge.ne.null)
    if length + dist(edge) is prime
        new_edge = edge
        new_length = length + dist(edge)
        break
    else
        edge = next(edge)
    endif
endwhile

[found_tour] = Search(root)

top = 1
path(top) = root
cum_length(top) = 0
node = root
first = start(root)
length = 0

while (not_done)
    [edge, length] = FindEdge(node, first, length)
    if edge > 0
        current_edge(top) = next(edge)
        node = to(edge)
        top = top + 1
        path(top) = node
        cum_length(top) = length
        first = start(node)
    else
        top = top - 1
        node = path(top)
        first = current_edge(top)
        length = cum_length(top)
    endif
endwhile

```