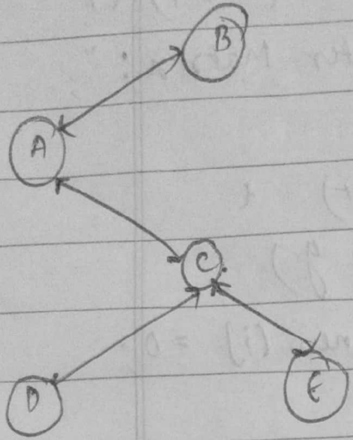
Raghav Gupta
1BM18CS075

Distance vector algorithm

adaptive non - adaptive



Bellman - ford equation

⇓

$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$

for each node y in N

Distance vector routing is an asynchronous algorithm in which node x sends the copy of its distance vector to all its neighbours.
When node x receives new distance vector from one of its neighbouring vectors, v, it saves the distance vector of v and uses the Bellmann - ford equation to update its own distance vector

* Algo

```
At each node x,
  for all destination y in N :
    $D_x(y) = c(x,y)$
  for each neighbour w
    $D_w(y) = ?$
  send distance vector $D_x = [D_x(y) : y \text{ in } N]$ to w

  wait (until it receive any distance vector
                from neigbour w)
  for each y in N :
    $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$
```

If $D_x(y)$ is changed for any destination y
send distance vector $D_x = [D_x(y) : y \in N]$
to all neighbours

Code :

header file

```
# define MAX 10
        int n
~~char router~~ .

    router () {
        for (int i=0; i< max ; i++)

        table old [i] = table new [i] = 99;
                                        }.
void copy ()
  for (i=0; i<n ; i++) {
        adj_odd [i] = adj - new [i];
        table_old [i] = table - new [i]

      }
    }
        int equal () {
      for (int i =0 ; i<n; i++) {
if  ( table-old [i] != table-new [i]
        return;

        }
void    input ()
cout << " Enter 1 if corresponding router
                      is adjacent to router "
     << char ('A' +j) <<  else enter 99 <<
                                       end
for (i=0; i<n; i++)
        if (i!=y)
            cout << char ('A'+i) << " ";
    cout << "\n Enter Matrix: ";

    for (i=0; i<n; i++) {
            if (i==y)
                  table_new (i] =0;
         else

         cin >> table_new [i]
    adj_new [i] = char ('A' + i);
         }
         cout << endl;
              }
 void  ~~main~~ ~~E~~  build  ()
    for ( int i=0; i<n ; i++)
    for ( k=0;  i!= J) && ( k<n); k++.

         if  ( table_old [i] != 99)
  if (   table_new [i] + r[i]. table_new [k] )
                               < table_new [k]

     table_new (k) = table_new [i] + r[i].
                                 table_new []


         }

      }
 void   build table ()
          int  i=0, j=0
       while  (i!=n)
          {
```

```
for (i=j, i<n; i++) {
        r[i].copy ();
        r[i].build (i) ; }
 for (i=0 ;i<n; i++)
        {
        if ( ! r[i].equal ()) {
                J = i
              break;
                }}
void  display ()
        Cout << destination
        for (i=0; i<n ; i++)
      Cout << char ('A'+i) << "  ";
        Cout << outgoing line.
        for (i=0 ;i<n ; i++)
          cout << adj_new [i] < "  "
           cout << \n Hop Count;
  }
void  main () {
cout << " Enter  no of  routers
       for (i=0; i<n; i++)
          r(i). input [i].
         build . table (),
          r[i]. display ()
          Cout << endl << endl;
                }
```