11/July/24

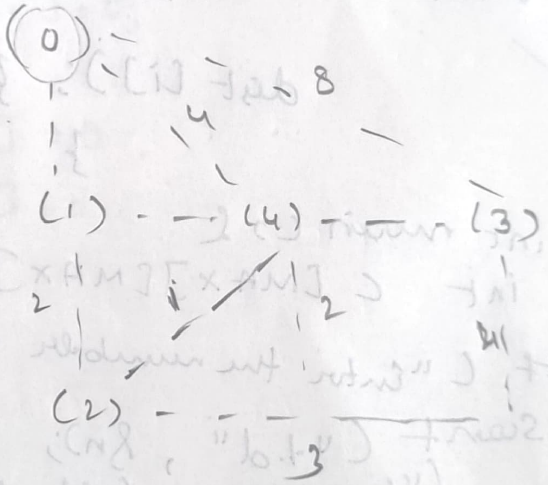
Dijkstra's algorithm.

```
#include <stdio.h>
# define INF 9999
# define MAX 100
void dijkstra (int c [MAX][MAX],
        int n, int src) {
    int dist [MAX], Vis[MAX], Count,
    min, u, i, j;
for (i=1; i <=n; i++) {
        dist [i] = INF;
          Vis [i]=0;
    }
  dist   [src] = 0;
    vis  [src] = 0;

        }
    dist [src] =0;
     Vis  [src]=1;
        Count=1;
    While (count != n) {
    min = INF;
    for (j=1; j<=n; j++){
            if (dist [j] < min && vis [j]!=1){
              min = dist[i];
                 u=j;
                }
              }
          vis [u] =1;
              Count++;
for (j=1; j<=n ;j++) {
  if ((min + c[u][j] < dist [j]) &&
     (vis [j] !=1))
   dist [j] = min + c[u][j];
        }
      }
 Printf ("Shortest distonces from source:%d:\n",
      src);
 for (i =1; i<=n; i++) {
   if (dist [i ]== INF)
   printf ("%d ->%d : Infinity \n", src , i);
   else
  Printf ("%d -> %d : %d\n", src, i,
         dist [i]); }
        }
 int main () {
  int  c [MAX][MAX], n, src;
Printf ("Enter the number of vertices:");
  Scanf ("%d" , &n);
Printf  ("Enter the Cost matrix (Enter INF as
          %d):\n", INF);
 for   (int  i=1 ;i<=n;i++) {
   for (int j=1; j<=n; j++) {
         Scanf  ("%d", &c[i][j]);
         if   (c[i] [j] ==INF)
           ~~Elif~~ c[i] [j] =INF;
             }
          }
        Printf  ("Enter the Source vertex :");
           Scanf  ("%d", &src);
        return 0;
```

Output: Enter number of vertices (max 100): 5

```
0      3      9999   9999   4

3      0      2      1      9999

9999   2      0      1      9999

9999   1      1      0      9999

4      9999   9999   9999   0 .
```

Graph:



Edge	weight
0 - 1	∞
1 - 2	∞
7 - 3	2
2 - 4	1

Kruskal's algorithm

```
# include      <stdio.h >
# define       INF= 9999
# define       MAX   10

Struct Edge {
    int  Src ,  dest  ,  weight ;
    };

    Struct    Subset {
            int Parent ;
               int rank ;

          };
      int  find     (Struct Subset   Subsets [ ], int i);
     void  Unionsets  (Struct  Subset  Subsets [ ], int x,
                            int y );
        void   kruskals   (int  c [MAX ] [MAX ], int n);
   int   Find (struct   Subset   Subsets [ ], int i ) {
          if  (subsets  [i ] . Parent   ! = i)
 Subsets       [i ] .  Parent  = find subsets , Subsets [i ].
                       Parent );
                                  [i ]. Parent ;
        return    Subsets
     void  union sets  (struct subset  Subsets [ ],
                            int x , int y ) {
               int    xroot = find (Subsets , x);
       int   yroot = find      (subsets , y));
  if (Subsets     [xroot ] . rank < subsets [yroot ].rank
          Subsets      [xroot   ] .Parent   = Yroot ;
  elseif (Subset s [ xroot ]. Rank  > subsets [yroot ].
                                              rank )
                     [yroot  ] . Parent   = xroot ;
    Subsets
        else {
```

```
Subsets [yroot].Parent = xroot;
      Subsets [xroot].rank++ ;

   }
 }

void kruskals (int C [MAX][MAX], int n)
      Struct Edge   result [n];
        int c=0;
        int i,j;
   for (i=0 ; i <n; i++)
   result [i]. Src = result [i].dest = result[i].
        weight =0;
      Struct Subset  Subsets [n];
      for (i=0; i<n; i++){
         Subsets [i]. Parent = i;
         Subsets [i]. Rank = 0;
        while (c<n-1}){
        int min = INF, u=-1, v=-1;
   for (i=0; i<n; i++){

        u=i =
        v=j ;

      }
    }

  int main(){
        int C [MAX][MAX], n;
        printf ("Enter the number of vertices'
          Scanf ("%d", &n);

        }
     }
```

```
Kruskals (c, n);
   return 0;
   }
```

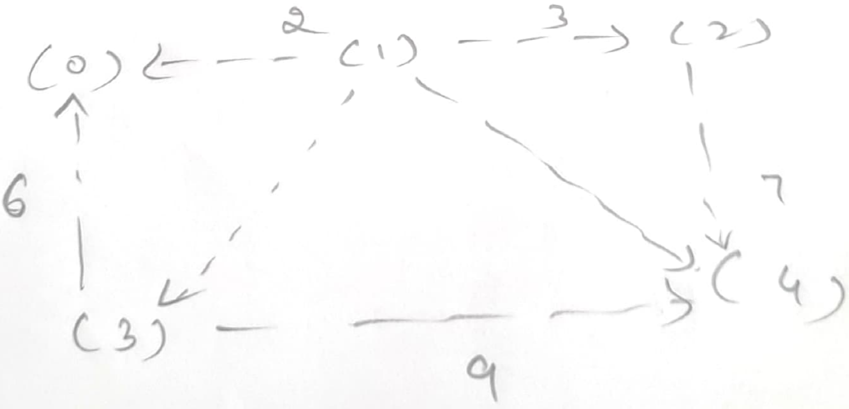
Output: Enter the number of vertices: 5

```
0   2   0   6   0
2   0   3   8   5
0   3   0   0   7
6   8   6   0   9
0   5   7   9   0
```

OP:

```
Edge          weight
0-1            2
4 - 2          3
0 - 3          6
1 - 4          5
```

11/7



2
3
(0)
(1)
(2)
6
7
(3)
(4)
9