```
Wick-8
           algorithm
                     which will find maximum cost required
Ponned cities (Prim's Algorithm)
prithm:
             D START
            2) Input n
             3) if i>=n golo step 8
            F gut a solo n=< 1 fi (1)
             () input acincl
             6) if I < n gots stepy
            3) if icu dopo voto 3
             8) prims (qin)
             9) STOP
prints (int * * arr, int n) ?
  victor < bool > visited (n. false);
  victor (Int> weight (n.INF).
 priority-queue <pair < int, int >>, vector < pair < int, int >>;
     greatin <poir < int, int>> min-heap;
 int ore = 0;
   weight [Nrc] = 0;
  min_heap.push (moke pair (weight (sic), sic));
   while (! min-heap impty ()) {
    u = min - heap top(), second;
     min-heap.pop();
    of (tus battaio!) of
       visited [u] = Tru;
   for (int v=0; v< 1 ; ++v) {
    >[v][v] rno ++ 0= ![v][v]] + Cv] + Cv] ballaiv!) fi
                                          weight [v])
          weight [v] = aricu](v];
         min-heap. puh (make-pair (weight [v],v)).
```

```
for lauto i: weight)

for lauto i: weight)
   return sum;
```

```
net dus (Krukal Agonthm)
                                   printing cost required
        D START
        2) Input n
         3) if i>=n goto stop 8
         of it is n goto stip 7
         2) inbay dobp [1][]
         6) if j<n goto stip 4
         3 gits abo n>i p (F
         8) Knukali (graph, n)
         q) stop
knukab (graph, n) }
 victor < pair < int, poir < int, int >>> 6;
 for (1=0; i<n; i++)
 (j:0; j<n; j++)
    if (graph [i][j] !=0)
     G. put-back (make-pair (graph [i][j), mate pair (1,j))).
       bort (6. begin(), 6. und());
      vector <int> parent (ni NIC);
        5=0%
        for (auto 1:6) }
         u = i · broand. First :
         v = i . swand. second.
         w= i itist;
        if (Union Byweight (parent, u,v))
            return x'
```

```
maximum budget required for a project.
                ) START
Algorithm :-
                 2) input n
                 3) il'i>= n goto step 8
                 f gots atop = < 1 ji (4)
                 5) input groph (13(1)
                  6) if jon goto step 4
                 7) if ion goto step 3
                 8) Kruskals (graphin)
                  9) STOP
        Kruskals (graph, n) {
           for (i=0; i<n; i++)
              for (j=0;j<n;j++)
                if (graph (i)(j)!=0)
                   6. push-back (make-pair (graph [i)[j], mote-
                sort (G.begin (), G. end (), greater < pair < int, pair
                 victor. <int> parent (n. NIL).
                      8 = 0;
                  for (auto i: 6) }
                     u=i, swand first;
                      V = i. second. second;
                      w= i.fint;
                 if (Union Byweight (parent, u,v))
                         8t=w;
                    return &;
```