In [1]:
```
class node:
    def __init__(self, name):
        self.name = int (name)
        self.adj_list_edges = []
```

In [2]:
```
class edge:
    def __init__(self, src, dist, wt):
        self.src = src
        self.dist = dist
        self.wt = wt
```

In [3]:
```
import random
import sys

def create_rand_graph(num_nodes, probability):
    g = []
    num_edges = 0

    for i in range(num_nodes):
        g.append(node(i))


    for i in range(num_nodes-1):
        wt = random.randint(1,1000)
        g[i].adj_list_edges.append(edge(g[i], g[i+1], wt))
        g[i+1].adj_list_edges.append(edge(g[i+1], g[i], wt))
        num_edges += 1

    wt = random.randint(1,10000)
    g[num_nodes-1].adj_list_edges.append(edge(g[num_nodes-1], g[0], wt))
    g[0].adj_list_edges.append(edge(g[0], g[num_nodes-1], wt))
    num_edges += 1

    #randomly adding other nodes in the graph
    for i in range(num_nodes):
        for j in range(i+1,num_nodes):
            if(random.randint(0,num_nodes) < (probability-1/num_nodes) * num_nodes):
                edge_weight = random.randint(1,10000)
                g[i].adj_list_edges.append(edge(g[i], g[j], edge_weight))
                g[j].adj_list_edges.append(edge(g[j], g[i], edge_weight))
                num_edges+=1

    return g, num_edges
```

In [4]:
```
    def max_BW_fringe(graph, status_array, bandwidth_array, N):
        index_of_max_bw_fringe = -1
        max_fringe_bw = -sys.maxsize - 1
        for i in range(0, N):
            if((status_array[i] == 1) and (bandwidth_array[i] >= max_fringe_bw)):
                index_of_max_bw_fringe = i
                max_fringe_bw = bandwidth_array[i]
        return index_of_max_bw_fringe
```

In [5]:

```
def apply_dijkstra(graph, s, t ):
    BW = [-1]* len(graph)
    status = [0]* len(graph)
    dad = [-1]*len(graph)
    fringes_count = 0
    #print(s.name)
    #print(t.name)

    status[s.name]= 2 #in tree
    BW[s.name]  =  float('inf')

    for edges in s.adj_list_edges:
        dad[edges.dist.name] = s.name
        status[edges.dist.name] = 1 #fringes
        BW[edges.dist.name] = edges.wt
        fringes_count += 1
        #print(fringes_count)

    maximum_bw = sys.maxsize
    while(fringes_count > 0):
        current_max = max_BW_fringe(graph,status, BW, len(graph))
        status[current_max] = 2
        if(current_max == t.name):
            maximum_bw = BW[current_max]
            path = str(t.name)
            k= t.name
            while(k != s.name):
                path = str(dad[k]) + "->" + path
                k = dad[k]
            print(path)
            return maximum_bw

        fringes_count -= 1

        for edge in graph[current_max].adj_list_edges:
            w = edge.dist.name
            if(status[w] == 0):
                status[w] = 1
                fringes_count += 1
                #print(fringes_count)
                dad[w] = current_max
                BW[w] =  min(BW[current_max], edge.wt)
            elif(status[w] == 1 and BW[w] < min(BW[current_max], edge.wt)):
                dad[w] = current_max
                BW[w] = min(BW[current_max], edge.wt)
```

In [6]:

```
import math

class Heap:
    def __init__(self, maxsize):
        self.size = 0
        self.maxsize = maxsize
        self.Arr = [-1]* self.maxsize
        self.Arr_dist = [-1] * self.maxsize
        self.Arr_pos = [-1] * self.maxsize

    def insert(self, element):
        self.Arr[self.size] = element.wt
        self.Arr_dist[self.size] = element.dist.name
        self.Arr_pos[element.dist.name]=self.size
        self.fix_heap_up(self.size)
        self.size += 1


    def delete(self,pos):
        self.size = self.size - 1
        self.swap(self.size,pos)
        self.Arr[self.size] = -1
        self.Arr_dist[self.size] = -1
        self.Arr_pos[self.Arr_dist[self.size]] = -1
        self.fix_heap_down(pos)


    def popmax(self):
        val = self.Arr[0]
        val_dist = self.Arr_dist[0]
        self.size = self.size - 1
        self.swap(0,self.size)
        self.Arr[self.size] = -1
        self.Arr_dist[self.size] = -1
        self.Arr_pos[self.Arr_dist[self.size]] = -1
        self.fix_heap_down(0)

        return val,val_dist


    def fix_heap_up(self, pos):
        if  self.Arr[pos] > self.Arr[self.parent(pos)]:
            self.swap(pos, self.parent(pos))
            pos = self.parent(pos)
            self.fix_heap_up(pos)
        else:
            return


    def fix_heap_down(self, pos):
        l = self.left_child(pos)
        r = self.right_child(pos)
        maximum = pos

        if l != -1:
            if l< self.size and self.Arr[l] > self.Arr[pos]:
                maximum = l
            else:
                maximum = pos

        if r!= -1:
            if r<= self.size and self.Arr[r] > self.Arr[maximum]:
                maximum = r
            else:
                maximum = maximum

        if maximum != pos:
            self.swap(pos, maximum)
            self.fix_heap_down(maximum)


    def parent(self, i):
        if i == 0:
            return 0
        if (i%2 != 0):
            return math.floor((i-1)/2)
        else :
            return math.floor((i-2)/2)



    def left_child(self, i):
        if 2*i+1 < self.size :
            return (2 * i + 1)
        else:
            return -1

    def right_child(self, i):
        if 2*i+2 < self.size :
            return (2 * i + 2)
        else:
            return  -1


    def swap(self, a_pos, b_pos):
        self.Arr[a_pos], self.Arr[b_pos] = self.Arr[b_pos], self.Arr[a_pos]
        self.Arr_dist[a_pos], self.Arr_dist[b_pos] = self.Arr_dist[b_pos], self.Arr_dis
        self.Arr_pos[self.Arr_dist[a_pos]], self.Arr_pos[self.Arr_dist[b_pos]] = self.A

    #def max(self):
      #  return self.Arr[0]
```

In [7]:

```
def apply_dijkstra_with_heap(graph, s , t):
    BW = [-1]* len(graph)
    status = [0]* len(graph)
    dad = [-1]*len(graph)

    my_heap = Heap(len(graph))

    status[s.name]= 2 #in tree
    BW[s.name]  =  float('inf')

    for edges in s.adj_list_edges:
        status[edges.dist.name] = 1 #fringes
        BW[edges.dist.name] = edges.wt
        dad[edges.dist.name] = s.name
        my_heap.insert(edges)

    while(my_heap.size != 0):
        current_max = my_heap.popmax()
        status[current_max[1]] = 2
        if(current_max[1] == t.name):
            maximum_bw = BW[current_max[1]]
            path = str(t.name)
            k= t.name
            while(k != s.name):
                path = str(dad[k]) + "->" + path
                k = dad[k]
            print(path)
            return maximum_bw
        for edge in graph[current_max[1]].adj_list_edges:
            w = edge.dist.name
            if(status[w] == 0):
                status[w] = 1
                BW[w] =  min(BW[current_max[1]], edge.wt)
                my_heap.insert(edge)
                dad[w] = current_max[1]
            elif(status[w] == 1 and BW[w] < min(BW[current_max[1]], edge.wt)):
                pt = my_heap.Arr_pos[w]
                my_heap.delete(pt) # position is not w
                BW[w] = min(BW[current_max[1]], edge.wt)
                my_heap.insert(edge)
                dad[w] = current_max[1]
```

In [8]:

```
class Heap_with_src:
    def __init__(self, maxsize):
        self.size = 0
        self.maxsize = maxsize
        self.Arr = [-1]* self.maxsize
        self.Arr_src = [-1] * self.maxsize
        self.Arr_dist = [-1] * self.maxsize


    def insert(self, element):
        self.Arr[self.size] = element.wt
        self.Arr_src[self.size] = element.src.name
        self.Arr_dist[self.size] = element.dist.name
        self.fix_heap_up(self.size)
        self.size += 1


    def popmax(self):
        val = self.Arr[0]
        val_src = self.Arr_src[0]
        val_dist = self.Arr_dist[0]
        self.size = self.size - 1
        self.swap(0,self.size)
        self.Arr[self.size] = -1
        self.Arr_dist[self.size] = -1
        self.Arr_src[self.size] = -1
        self.fix_heap_down(0)

        return val, val_src, val_dist


    def fix_heap_up(self, pos):
        if  self.Arr[pos] > self.Arr[self.parent(pos)]:
            self.swap(pos, self.parent(pos))
            pos = self.parent(pos)
            self.fix_heap_up(pos)
        else:
            return
    def fix_heap_down(self, pos):
        l = self.left_child(pos)
        r = self.right_child(pos)
        maximum = pos

        if l != -1:
            if l< self.size and self.Arr[l] > self.Arr[pos]:
                maximum = l
            else:
                maximum = pos

        if r!= -1:
            if r<= self.size and self.Arr[r] > self.Arr[maximum]:
                maximum = r
            else:
                maximum = maximum

        if maximum != pos:
            self.swap(pos, maximum)
            self.fix_heap_down(maximum)


    def parent(self, i):
        if i == 0:
            return 0
        if (i%2 != 0):
            return math.floor((i-1)/2)
        else :
            return math.floor((i-2)/2)



    def left_child(self, i):
        if 2*i+1 < self.size :
            return (2 * i + 1)
        else:
            return -1

    def right_child(self, i):
        if 2*i+2 < self.size :
            return (2 * i + 2)
        else:
            return  -1

    def swap(self, a_pos, b_pos):
        self.Arr[a_pos], self.Arr[b_pos] = self.Arr[b_pos], self.Arr[a_pos]
        self.Arr_src[a_pos], self.Arr_src[b_pos] = self.Arr_src[b_pos], self.Arr_src[a_
        self.Arr_dist[a_pos], self.Arr_dist[b_pos] = self.Arr_dist[b_pos], self.Arr_dis
```

In [9]:

```
from queue import Queue
import sys


def Find(v, p):
    w = v
    q = Queue()
    while(p[w] != -1):
        q.put(w)
        w = p[w]
    while not q.empty():
        p[q.get()] = w

    return w


def Union(r1, r2, h, p):
    if (h[r1] > h[r2]):
        p[r2] = r1
    elif (h[r2] > h[r1]):
        p[r1] = r2
    else:
        p[r2] = r1
        h[r1] = h[r1] + 1

def apply_dfs(graph, node_number, color_array, path_array, target):
    if (node_number == target):
        return True
    found = False
    color_array[node_number] = 2 # gray
    for edge in graph[node_number].adj_list_edges:
        if(color_array[edge.dist.name] == 1):
            path_array[edge.dist.name] = edge.src.name
            #print('bbfggr')
            found = apply_dfs(graph, edge.dist.name, color_array, path_array, target)
            if found:
                break
    color_array[node_number] = 3 #black
    return found


def apply_kruskal(gr, s, t):
    graph = gr[0]
    parent= [-1]*len(graph)
    hgt = [0]*len(graph)
    col = [1]*len(graph)
    path = [-1]*len(graph)

    my_heap_src = Heap_with_src(gr[1])

    for vertices in graph:
        for edges in vertices.adj_list_edges:
            if edges.src.name < edges.dist.name:
                my_heap_src.insert(edges)

    g2 = []
    for i in range(len(graph)):
        g2.append(node(i))

    for i in range(0,my_heap_src.maxsize):
        w, sr, d = my_heap_src.popmax()
        x1 = Find(sr, parent)
        x2 =  Find(d, parent)
        #print(uk,' ',vk,  ' ',wt)
        if (x1 != x2):
            g2[sr].adj_list_edges.append(edge(g2[sr], g2[d], w))
            g2[d].adj_list_edges.append(edge(g2[d], g2[sr], w))
            Union(x1,x2, hgt, parent)
    is_valid = apply_dfs(g2, s, col, path, t)

    poth = str(t)
    k= t
    Max_BW = sys.maxsize
    while(k != s):
        for ed in g2[k].adj_list_edges:
            if ed.dist.name == path[k]:
                Max_BW = min(Max_BW, ed.wt)
        poth = str(path[k]) + "->" + poth
        k = path[k]
    print(poth)


    return Max_BW
```

In [10]:

```
#amar_graph = create_rand_graph(10)
import time
```

In [11]:

```
#final testing
sparse_graph_1 = create_rand_graph(5000, float(6/5000))
dense_graph_1 = create_rand_graph(5000, 0.20)
```

In [12]:

```
source_1 = random.randint(1,4999)
target_1 = random.randint(1,4999)
```

In [13]:

```
strt = time.time()
result = apply_dijkstra(sparse_graph_1[0], sparse_graph_1[0][source_1], sparse_graph_1[
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)
```

```
1618->4708->4218->694->3228->590->2355->4874->3442->4476->53->2033->4156->873->1291->804
->1801->1169->4408->4192->3627->3672->962->1221->608->208->1163->1833->3656->1647->4216-
>4499->773->3842->2590->4898->468->90->3950->1048->2141->3764->3963->2900->858->3415->99
3->4988->739->2663
Maximum Bandwidth with dijkstra:  7907
Time taken:  0.0797872543334961
```

In [14]:

```
strt = time.time()
result = apply_dijkstra_with_heap(sparse_graph_1[0], sparse_graph_1[0][source_1], spars
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)
```

```
1618->4708->4218->694->3228->590->2355->4874->3442->4476->53->2033->4156->873->1291->804
->1801->1169->4408->4192->3627->3672->962->1221->608->208->1163->1833->3656->1647->4216-
>4499->773->3842->2590->4898->468->90->3950->1048->2141->3764->3963->2900->858->3415->99
3->4988->739->2663
Maximum Bandwidth with dijkstra with heap:  7907
Time taken:  0.01593017578125
```

In [15]:

```
strt = time.time()
result = apply_kruskal(sparse_graph_1, source_1, target_1)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
1618->4708->4218->694->3228->590->2355->4874->3442->4476->53->2033->4156->873->1291->804
->1801->1169->4408->4192->3627->3672->962->1221->608->208->1163->1833->3656->1647->4216-
>4499->773->3842->2590->4898->468->90->3950->1048->2141->3764->3963->2900->858->3415->99
3->4988->739->2663
Maximum Bandwidth with kruskal:  7907
Time taken:  1.4046571254730225
```

In [16]:

```
strt = time.time()
result = apply_dijkstra(dense_graph_1[0], dense_graph_1[0][source_1], dense_graph_1[0][
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)
```

```
1618->2148->1996->4838->3791->3764->3490->3522->2663
Maximum Bandwidth with dijkstra:  9956
Time taken:  6.700338125228882
```

In [17]:

```
strt = time.time()
result = apply_dijkstra_with_heap(dense_graph_1[0], dense_graph_1[0][source_1], dense_g
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)
```

```
1618->2148->1996->4838->3791->3764->3490->3522->2663
Maximum Bandwidth with dijkstra with heap:  9956
Time taken:  3.6363608837127686
```

In [18]:

```
strt = time.time()
result = apply_kruskal(dense_graph_1, source_1, target_1)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
1618->2148->1996->4838->3791->3764->3490->3522->2663
Maximum Bandwidth with kruskal:  9956
Time taken:  270.0397584438324
```

In [19]:

```
source_2 = random.randint(1,4999)
target_2 = random.randint(1,4999)
```

In [20]:

```
strt = time.time()
result = apply_dijkstra(sparse_graph_1[0], sparse_graph_1[0][source_2], sparse_graph_1[
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)
```

```
1400->75->4149->1150->2980->3167->3794->4919->3691->3576->3974->3554->3542->3957->4498->
3294->2965->4642->3300->4237->4656->3093->2896->3981->3296->3191->3881->3889->4976->4203
->3272->3248->4268->3432->4154->2319->4175->3290->1259
Maximum Bandwidth with dijkstra:  4360
Time taken:  2.0220634937286377
```

In [21]:
```
strt = time.time()
result = apply_dijkstra_with_heap(sparse_graph_1[0], sparse_graph_1[0][source_2], spars
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)
```

```
1400->75->4149->1150->2980->3167->3794->960->2029->2225->4123->1595->3229->4540->3071->2
455->4871->4793->1943->1114->1108->167->4391->1542->1564->3873->4175->3290->1259
Maximum Bandwidth with dijkstra with heap:  4360
Time taken:  0.2397623062133789
```

In [22]:
```
strt = time.time()
result = apply_kruskal(sparse_graph_1, source_2, target_2)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
1400->75->4149->1150->2980->3167->3794->960->2029->2225->4123->1595->3229->4540->3071->2
455->4871->4793->1943->1114->1108->167->2611->51->3342->2708->2430->3399->273->2196->459
4->1345->2419->617->1283->465->4512->4883->2883->3761->3000->93->4798->2588->287->3290->
1259
Maximum Bandwidth with kruskal:  4360
Time taken:  1.6451702117919922
```

In [23]:
```
strt = time.time()
result = apply_dijkstra(dense_graph_1[0], dense_graph_1[0][source_2], dense_graph_1[0][
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)
```

```
1400->4475->4471->3254->3284->4323->4200->4584->4300->4322->4981->3940->3698->4949->3783
->3557->4446->4024->4978->3554->4125->3393->3851->3551->3137->3240->4159->3690->3552->36
56->4178->4840->3389->3353->3771->3362->3460->4179->4244->3810->4857->3042->3127->1259
Maximum Bandwidth with dijkstra:  9971
Time taken:  4.597011566162109
```

In [24]:
```
strt = time.time()
result = apply_dijkstra_with_heap(dense_graph_1[0], dense_graph_1[0][source_2], dense_g
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)
```

```
1400->4475->2588->3826->566->775->1888->1676->4720->1007->1033->2488->4932->2167->1084->
2504->343->3822->2830->4359->3695->2984->3261->2675->2536->184->701->2149->4928->3134->3
364->11->2376->4322->4300->4584->307->4637->193->2033->4135->4572->2650->3542->4222->884
->177->3188->1732->1801->2464->1208->3708->1248->180->240->4803->1295->2283->3209->275->
3042->3127->1259
Maximum Bandwidth with dijkstra with heap:  9971
Time taken:  1.4890692234039307
```

In [25]:
```
strt = time.time()
result = apply_kruskal(dense_graph_1, source_2, target_2)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
1400->4475->1141->2169->347->4195->888->4694->3799->561->3245->4715->2201->2330->538->10
30->4220->3715->1107->2041->573->3241->2974->2934->4663->3240->3137->3551->3851->771->37
91->3764->3490->1570->2023->4888->1248->3708->1208->2464->1801->1732->3188->177->884->35
```

```
60->651->3836->1280->4231->1259
Maximum Bandwidth with kruskal:  9971
Time taken:  252.84461069107056
```

In [26]:

```python
source_3 = random.randint(1,4999)
target_3 = random.randint(1,4999)

strt = time.time()
result = apply_dijkstra(sparse_graph_1[0], sparse_graph_1[0][source_3], sparse_graph_1[
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(sparse_graph_1[0], sparse_graph_1[0][source_3], spars
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(sparse_graph_1, source_3, target_3)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra(dense_graph_1[0], dense_graph_1[0][source_3], dense_graph_1[0][
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(dense_graph_1[0], dense_graph_1[0][source_3], dense_g
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(dense_graph_1, source_3, target_3)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
4418->2024->4838->1643->2989->1778->4112->3153->3434->2316->3267->3248->4664->3433->3615
->2392->3929->3197->4362->3294->4498->3957->3542->3554->4620->4018->2802->2829->3877->35
55->3619->4701->4691->2080->3690->2430->3480->3059->4583->2280->3535->4413->4287->1841->
4067->2921->3850->3965->4123->2225->2029->4062->2254->2855->2489->4372->1655->3336
Maximum Bandwidth with dijkstra:  5424
Time taken:  1.2712302207946777
4418->2024->3385->283->926->2314->4607->3863->3537->2266->1137->710->4017->4560->3582->3
348->159->2074->1655->3336
Maximum Bandwidth with dijkstra with heap:  5424
Time taken:  0.15621304512023926
4418->2024->3385->283->926->2314->4607->3863->3537->2266->1646->2225->4123->1595->3229->
4540->3071->2455->4871->4793->1943->1114->1108->167->2611->51->3342->1401->92->1324->138
```

```
3->4548->116->4930->2424->1226->581->3119->758->2382->3303->284->2748->3955->245->1990->
3822->4073->1017->618->3633->108->905->4210->2766->4845->4166->1874->353->328->3336
Maximum Bandwidth with kruskal:  5424
Time taken:  1.3703923225402832
4418->949->1071->4353->1437->3592->3827->3014->3778->3609->2857->1275->283->4993->1829->
3852->3336
Maximum Bandwidth with dijkstra:  9987
Time taken:  3.7282705307006836
4418->949->1071->4353->1437->581->4448->2357->2081->4268->267->1212->4021->627->1282->25
67->3611->1473->1040->4715->2201->2330->538->1030->4220->3715->1107->2041->50->3030->361
8->4139->3551->3330->3641->1717->1695->222->2599->1363->3609->2857->1275->283->4993->182
9->3852->3336
Maximum Bandwidth with dijkstra with heap:  9987
Time taken:  2.4568209648132324
4418->949->1071->4353->1437->581->4448->2357->2081->4268->267->1212->4021->627->1282->25
67->3611->1473->1040->4715->2201->2330->538->1030->4220->3715->1107->2041->573->3241->29
74->2934->4663->3240->3137->3551->3330->3641->1717->1695->222->2599->1363->3609->2857->1
275->283->4993->1829->3852->3336
Maximum Bandwidth with kruskal:  9987
Time taken:  244.76609659194946
```

In [27]:

```
source_4 = random.randint(1,4999)
target_4 = random.randint(1,4999)

strt = time.time()
result = apply_dijkstra(sparse_graph_1[0], sparse_graph_1[0][source_4], sparse_graph_1[
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(sparse_graph_1[0], sparse_graph_1[0][source_4], spars
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(sparse_graph_1, source_4, target_4)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra(dense_graph_1[0], dense_graph_1[0][source_4], dense_graph_1[0][
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(dense_graph_1[0], dense_graph_1[0][source_4], dense_g
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(dense_graph_1, source_4, target_4)
```

```
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
845->2248->2299->2645->1460->763->4511->2949->2594->638->2305->1455->3702->3812->1640->3
889->3881->2513->1788->2283->1055->2879->2736->2707->2811->4793->4871->2455->3071->4540-
>3229->1595->4123->2225->1646->2266->3537->3863->2776->4667->4415->1347->2483->4427->368
7->3242->1850->2922->3714->3860->3621->1549->2102->3657->2708->3342->2093->1825->4409->2
098->3845->3476->3615->2392->3929->3197->4362->3294->2125->2365->3897->1851->2953->2183-
>3506->4358->2295->3738->3922->4597->2907->4548->2505->2511->2585->3502->3301->4412->453
7->3130->4204->939->2429->3603->4304->4308->4935
Maximum Bandwidth with dijkstra:  6874
Time taken:  0.9272215366363525
845->2248->2299->2645->1460->763->4511->2949->2594->1965->563->2458->2245->4330->4929->1
204->1137->2266->1646->2225->4123->1595->3229->4540->3071->2455->4871->4793->1943->1114-
>1108->167->2611->51->3342->2093->1825->4409->932->852->4881->344->2370->2060->4304->430
8->4935
Maximum Bandwidth with dijkstra with heap:  6874
Time taken:  0.14055418968200684
845->2248->2299->2645->1460->763->4511->2949->2594->1965->563->2458->2245->4330->4929->1
204->1137->2266->1646->2225->4123->1595->3229->4540->3071->2455->4871->4793->1943->1114-
>1108->167->2611->51->3342->2093->1825->4409->932->852->4881->344->2370->2060->4304->430
8->4935
Maximum Bandwidth with kruskal:  6874
Time taken:  1.355623722076416
845->3359->4781->4186->726->2664->2076->2859->3864->215->4516->2167->1084->2504->4227->2
477->4375->2066->3953->1226->4957->4943->558->1815->1528->325->2345->4224->1598->3441->3
544->2259->3545->792->4450->4584->4200->4323->3360->2407->1921->676->4353->1437->581->44
48->2357->1465->900->2039->2334->2457->4968->595->678->1180->4935
Maximum Bandwidth with dijkstra:  9987
Time taken:  3.8905069828033447
845->3359->4781->4186->726->2664->2076->2859->3864->215->4516->2167->1084->2504->343->38
22->2830->4359->3695->2984->3261->2675->2536->184->701->2149->4928->3134->3364->11->2376
->4322->4300->4584->4200->4323->3360->3824->1999->3188->1732->1801->2464->1208->3708->12
48->180->240->3555->3646->2627->1040->2117->120->488->4370->1681->678->1180->4935
Maximum Bandwidth with dijkstra with heap:  9987
Time taken:  1.5512614250183105
845->3359->4781->4186->726->2664->2076->2859->3864->215->4516->2167->4932->2488->1033->1
007->4720->1676->91->3646->2627->1040->2117->120->488->4370->1681->678->1180->4935
Maximum Bandwidth with kruskal:  9987
Time taken:  246.0799651145935
```

In [28]:

```
source_5 = random.randint(1,4999)
target_5 = random.randint(1,4999)

strt = time.time()
result = apply_dijkstra(sparse_graph_1[0], sparse_graph_1[0][source_5], sparse_graph_1[
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(sparse_graph_1[0], sparse_graph_1[0][source_5], spars
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(sparse_graph_1, source_5, target_5)
endt = time.time()
```

```
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra(dense_graph_1[0], dense_graph_1[0][source_5], dense_graph_1[0][
endt = time.time()
print("Maximum Bandwidth with dijkstra: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_dijkstra_with_heap(dense_graph_1[0], dense_graph_1[0][source_5], dense_g
endt = time.time()
print("Maximum Bandwidth with dijkstra with heap: ",result)
print("Time taken: ",endt-strt)


strt = time.time()
result = apply_kruskal(dense_graph_1, source_5, target_5)
endt = time.time()
print("Maximum Bandwidth with kruskal: ",result)
print("Time taken: ",endt-strt)
```

```
3566->4232->4827->1530->1440->2561->1183->1617->1016->2336->4663->3427->2630->4946->3443
->3632->2741->1321->2708->3657->2102->1549->3621->4094->2893->4230->1792->3156->1753->17
86->1778->2989->1215->1063->2183->4521->4514->2715->1700->3518->3930->2780->2790->2794->
4293->4606
Maximum Bandwidth with dijkstra:  4367
Time taken:  2.801439046859741
3566->4232->4827->1530->3387->142->4235->4931->4293->4606
Maximum Bandwidth with dijkstra with heap:  4367
Time taken:  0.33890390396118164
3566->4232->4827->1530->3387->142->4235->4931->4293->4606
Maximum Bandwidth with kruskal:  4367
Time taken:  1.333824872970581
3566->4674->714->2316->1753->4151->1861->982->3378->2887->3018->3887->4222->2855->4228->
3854->3554->4125->3393->3851->3551->3137->3240->4159->3690->3552->3656->3288->3635->3275
->4031->4938->4243->3350->3877->4120->2994->2093->2023->4888->1248->1120->3381->2879->98
8->896->3649->611->3583->4606
Maximum Bandwidth with dijkstra:  9978
Time taken:  6.371484756469727
3566->4674->714->2316->1753->4151->1861->982->4601->3441->3544->2259->3545->792->4450->4
584->307->4637->193->2033->4135->4572->2650->3542->4222->884->177->3188->1732->1801->246
4->1208->3708->1248->1120->3381->2879->988->896->3649->611->3583->4606
Maximum Bandwidth with dijkstra with heap:  9978
Time taken:  3.4535512924194336
3566->4674->714->2316->1753->4151->1861->982->4601->3441->1598->4224->2345->1251->2196->
4327->1008->1543->1841->3786->1120->3381->2879->988->896->3649->611->3583->4606
Maximum Bandwidth with kruskal:  9978
Time taken:  264.26337790489197
```

In [34]:

```
print("For Sparse Graph")
print(" ")
for k in range(1,5):
    sparse_graph = create_rand_graph(5000, float(6/5000))
    for i in range(0,5):
        source = random.randint(1,4999)
        target = random.randint(1,4999)

        strt = time.time()
        result = apply_dijkstra(sparse_graph[0], sparse_graph[0][source], sparse_graph[
        endt = time.time()
        print("Maximum Bandwidth with dijkstra: ",result)
        print("Time taken: ",endt-strt)


        strt = time.time()
        result = apply_dijkstra_with_heap(sparse_graph[0], sparse_graph[0][source], spa
        endt = time.time()
        print("Maximum Bandwidth with dijkstra with heap: ",result)
        print("Time taken: ",endt-strt)


        strt = time.time()
        result = apply_kruskal(sparse_graph, source, target)
        endt = time.time()
        print("Maximum Bandwidth with kruskal: ",result)
        print("Time taken: ",endt-strt)
```

```
For Sparse Graph

3016->2866->4118->2846->3074->4179->4798->4807->4539->4189->3993->4740->3865->4825->4126
->3856->4927->4611->4995->3905->4511->4674->3990->4497->782
Maximum Bandwidth with dijkstra:  1228
Time taken:  2.4245970249176025
3016->2866->320->4696->4291->1964->4167->54->548->1276->2434->350->22->322->4470->3435->
48->2353->272->3330->2564->4007->1080->3190->4403->2383->671->1043->125->264->3517->1238
->3090->4497->782
Maximum Bandwidth with dijkstra with heap:  1228
Time taken:  0.05502915382385254
3016->2866->320->4696->4291->1964->4167->54->548->1276->2434->350->22->322->4470->3435->
48->2353->272->3330->2564->4007->1080->3190->4403->2383->671->1043->636->1616->708->614-
>298->1168->2310->794->47->3177->105->3963->3365->3193->4802->1773->3517->1238->3090->44
97->782
Maximum Bandwidth with kruskal:  1228
Time taken:  1.2550795078277588
321->3689->2770->2509->1043->636->2368->536->4164->954->3172->2848->3251->1552->557->493
5->4227->353->618->394->2877->322->3812->1897->2367->4925->4644->2017->1067->2470->3418-
>4513->1271->932->1411->1873->1497->2765->518->3434->4672->2706->3531
Maximum Bandwidth with dijkstra:  7422
Time taken:  1.4973580837249756
321->3689->2770->2509->1043->636->1616->708->614->298->1168->2310->2327->664->2284->1896
->1396->1410->3436->3006->2018->1108->1405->3824->3577->3921->2216->932->1411->1873->149
7->2765->518->3434->4672->2706->3531
Maximum Bandwidth with dijkstra with heap:  7422
Time taken:  0.13935279846191406
321->3689->2770->2509->1043->636->1616->708->614->298->1168->2310->794->47->3177->105->3
963->3365->3193->4802->1773->3517->1238->688->1291->4075->2002->3277->752->1437->962->37
79->1410->3436->3006->2018->1108->1405->3824->3577->3921->2216->932->1411->1873->1497->2
765->518->3434->4672->2706->3531
Maximum Bandwidth with kruskal:  7422
Time taken:  1.2851991653442383
4869->3754->3590->1065->4630->1863->4776->3325->588->2255->471->4733->2572->1250->4384->
2348->4353->3636->1969->2186->2635->4465->915->3629->3055->2124->4254->2401->3298->2127-
>802->3787->3995->1291->4075->2002->3277->752->1437->962->3779->1410->1396->4271->3252->
3500->955->803->1485->2919->990->3918->2879->4638->2838->1297->3449->1590->3527->738->73
2->595->1921->4296->2598->3334->3174->665->2295->4420->3404->2148->3689->2770->2509->104
3->636->1616->708->1387->462->2544->2949->4886->4192->777->4506->1358->922->3522
Maximum Bandwidth with dijkstra:  7404
Time taken:  0.7145898342132568
4869->3754->3590->1065->4630->1863->4776->3325->588->2255->471->4733->2572->1250->4384->
2348->2270->2087->4583->4321->3557->1684->3387->2677->4067->727->2328->4764->3763->4941-
>3748->3779->1410->1396->1896->2284->984->3817->969->1444->4417->3392->381->4853->777->4
```

```
506->1358->922->3522
Maximum Bandwidth with dijkstra with heap:  7404
Time taken:  0.13280677795410156
4869->3754->3590->1065->4630->1863->4776->3325->588->2255->471->4733->2572->1250->4384->
2348->2270->2087->4583->4321->3557->1684->3387->2677->4067->727->2328->4764->3763->4941-
>3748->3779->962->1437->752->3277->2002->4075->1291->688->1238->3517->1773->4802->3193->
3365->3963->105->3177->47->794->2310->2327->664->2284->984->3817->969->1444->4417->3392-
>118->1286->3036->1567->450->2769->68->117->3295->3098->2533->4604->757->4853->777->4506
->1358->922->3522
Maximum Bandwidth with kruskal:  7404
Time taken:  1.3013403415679932
2874->2333->2731->875->4742->4521->2406->2337->1655->3626->2368->636->1616->708->614->29
8->1168->2310->2327->664->2284->1896->1396->1410->3436->3006->2018->1968->1292->3733->16
76->4211->4741->2842->2110->3285->2882->3297->3418->4513->1271->932->1411->1873->1497->2
765->518->2365->1608->3664->3362->181->4378->466->1610->184
Maximum Bandwidth with dijkstra:  7297
Time taken:  1.29679536819458
2874->2333->2731->875->4742->4521->2406->2337->1655->3626->2368->636->1616->708->614->29
8->1168->2310->2327->664->2284->1896->1396->1410->3436->3006->2018->1108->1405->3824->35
77->3921->2216->932->1411->1873->1497->2765->518->2365->1608->3664->3362->181->4378->466
->1610->184
Maximum Bandwidth with dijkstra with heap:  7297
Time taken:  0.23281478881835938
2874->2333->2731->875->4742->4521->2406->2337->1655->3626->2368->636->1616->708->614->29
8->1168->2310->794->47->3177->105->3963->3365->3193->4802->1773->3517->1238->688->1291->
4075->2002->3277->752->1437->962->3779->1410->3436->3006->2018->1108->1405->3824->3577->
3921->2216->932->1411->1873->1497->2765->518->2365->1608->3664->3362->181->4378->466->16
10->184
Maximum Bandwidth with kruskal:  7297
Time taken:  1.253995656967163
2695->2558->3416->4490->3112->647->1442->3153->106->1088->1290->4533->1324->4123->3964->
90->1685->281->488->4158->2238->2718->3070->2616->4453->59->4290->171
Maximum Bandwidth with dijkstra:  7650
Time taken:  0.8050944805145264
2695->2558->3416->4490->3112->647->1442->3153->106->1088->1290->4533->1324->4123->3964->
90->1685->281->488->4158->2238->2718->3070->2616->4453->59->4290->171
Maximum Bandwidth with dijkstra with heap:  7650
Time taken:  0.12497711181640625
2695->2558->3416->4490->3112->647->1442->3153->106->1088->1290->4533->1324->4123->3964->
90->1685->281->488->4158->2238->2718->3070->2616->4453->59->4290->171
Maximum Bandwidth with kruskal:  7650
Time taken:  1.3501689434051514
1211->4640->3579->4814->2266->4563->2368->2396->3052->3656->4886->3734->2440->2478->4131
->3229->3547->3658->3475->2738->3028->3627->1217->4039->1155
Maximum Bandwidth with dijkstra:  4752
Time taken:  2.282524585723877
1211->4640->3579->526->4818->4590->1676->4022->4804->4106->3938->4236->234->3808->1087->
3050->3033->4104->3176->4259->3919->213->2891->3943->4216->278->47->4620->2861->2074->74
4->298->1155
Maximum Bandwidth with dijkstra with heap:  4752
Time taken:  0.12500715255737305
1211->4640->3579->526->4818->4590->1676->4022->4804->4106->3938->4236->234->3808->1087->
3050->3033->4104->3176->4259->4979->3725->1907->1989->774->2684->3254->4155->3598->2384-
>2178->58->3893->710->4292->4620->2861->2074->744->298->1155
Maximum Bandwidth with kruskal:  4752
Time taken:  1.3227949142456055
3706->2325->4669->4388->4145->4671->4140->4405->3786->4646->4709->4208->4287->4221->4953
->3736->3743->4144->3708->3810->4667->4757->3600->4499->3952->4426->3822->4843->4172->48
72->4786->3805->4857->4252->4923->4040->4688->3715->3669->4141->3919->4259->3833->4455->
3447->2618->3323
Maximum Bandwidth with dijkstra:  1712
Time taken:  1.2018330097198486
3706->2325->1907->3725->4979->4259->3176->4104->3033->811->4009->118->3743->1391->4189->
3633->4443->2187->4582->3892->193->1174->1448->2621->2622->4227->3781->2703->3410->4899-
>4565->4973->4311->594->3323
```

```
Maximum Bandwidth with dijkstra with heap:  1712
Time taken:  0.15860199928283691
3706->2325->1907->3725->4979->4259->3176->4104->3033->3050->1087->3808->234->4236->3938-
>4106->4804->4022->1676->4590->4818->1400->1835->602->510->3538->13->1292->3946->4760->3
698->2618->3323
Maximum Bandwidth with kruskal:  1712
Time taken:  1.3585364818572998
1766->3590->1180->992->848->3478->143->3466->3115->4600->2435->4253->122->4532->2220->13
91->3743->118->4009->811->3033->4104->3176->4259->4979->3725->1907->1989->774->2684->325
4->4155->3598->2384->2178->58->3893->710->4292->4620->2861->1210->1386->3333
Maximum Bandwidth with dijkstra:  7777
Time taken:  0.4468379020690918
1766->3590->1180->992->848->3478->143->3466->3115->4600->2435->4253->122->4532->2220->13
91->3743->118->4009->811->3033->4104->3176->4259->4979->3725->1907->1989->774->2684->325
4->4155->3598->2384->2178->58->3893->710->4292->4620->2861->1210->1386->3333
Maximum Bandwidth with dijkstra with heap:  7777
Time taken:  0.09070324897766113
1766->3590->1180->992->848->3478->143->3466->3115->4600->2435->4253->122->4532->2220->13
91->3743->118->4009->811->3033->4104->3176->4259->4979->3725->1907->1989->774->2684->325
4->4155->3598->2384->2178->58->3893->710->4292->4620->2861->1210->1386->3333
Maximum Bandwidth with kruskal:  7777
Time taken:  1.2831099033355713
4880->2497->2629->2921->3670->4464->1504->2426->3650->4467->219->1528->867
Maximum Bandwidth with dijkstra:  7191
Time taken:  1.5506501197814941
4880->2497->2629->2921->3670->4464->1504->2426->3650->4467->219->1528->867
Maximum Bandwidth with dijkstra with heap:  7191
Time taken:  0.17183279991149902
4880->2497->2629->2921->3670->4464->1504->2426->3650->4467->219->1528->867
Maximum Bandwidth with kruskal:  7191
Time taken:  1.3714282512664795
479->1277->582->3979->4230->4831->4071->3375->3910->3644->3134->3006->3388->4220->4076->
4369->4891->4872->4172->4843->3822->4426->3952->3863->3796->3719->4942->3650->4827->4143
->4346->3898->3612->4314->3759->3869->4022->4804->4106->3938->4366->4961->3782->3788->44
85->3542->1873
Maximum Bandwidth with dijkstra:  2978
Time taken:  1.8399128913879395
479->1277->582->3501->2070->2372->15->4828->4976->249->4610->4642->2558->4167->115->1074
->3702->1873
Maximum Bandwidth with dijkstra with heap:  2978
Time taken:  0.0
479->1277->582->3501->2070->2372->15->4828->4976->249->4610->4642->2558->4167->115->1074
->3702->1873
Maximum Bandwidth with kruskal:  2978
Time taken:  1.432565450668335
1539->4693->1961->3120->2240->2603->2976->1948->1965->4249->2882->1621->2090->3360->3556
->2559->3540->2823->4683->2430->3535->3782->4620->2731->2841->3775->1265->3749->3324->37
41->2840->3342->4190->4333->4437->3555->2574->2129->2885->1856->1590->3965->1604->1147->
1331->3875->937->844->1849->3951->3125->1983->2180->2428->3500->3290->4324->1001->2419->
4302->2513->3794->2589->2788->4844->3063->1241->2802->3173->1353->4735->4545
Maximum Bandwidth with dijkstra:  7147
Time taken:  1.7097926139831543
1539->4693->3468->2273->4880->1273->3013->4744->3517->4076->2877->2828->2220->1801->1434
->4941->4854->4634->140->1081->4044->3406->1231->1444->1127->2296->1526->1406->1151->192
3->4870->4607->2524->79->360->3786->4768->3204->3027->4302->2513->3794->2589->2788->4844
->3063->1241->2802->3173->1353->4735->4545
Maximum Bandwidth with dijkstra with heap:  7147
Time taken:  0.20303583145141602
1539->4693->3468->2273->4880->1273->3013->4744->3517->4076->2877->2828->2220->1801->1434
->4941->4854->4634->140->1081->4044->3406->1231->1695->2554->4618->1147->1331->3875->937
->844->1849->4764->1393->511->4496->2304->1851->1253->815->3443->590->416->1447->1591->3
912->2218->1798->3899->4300->477->4440->836->3914->914->4188->580->4396->4550->3054->438
6->2653->4448->1099->2812->2742->954->4262->3787->1746->2984->4664->2702->102->2757->493
6->2513->3794->2589->2788->4844->3063->1241->2802->3173->1353->4735->4545
Maximum Bandwidth with kruskal:  7147
```

```
Time taken:  1.2693729400634766
3963->3407->4056->2074->2551->383->4120->3892->2622->2749->2072->4448->2653->3509->3367-
>2513->4302->3027->4327->3595->3887->3209->2180->2454->3266->4222->2819->1996->1818->306
6->2930->3229->4180->2448->1950->3012
Maximum Bandwidth with dijkstra:  6142
Time taken:  0.6591475009918213
3963->3407->4056->2074->2551->383->4120->3892->2622->539->4396->580->4188->914->3914->83
6->4440->477->3353->451->1012->35->1184->1491->545->654->4602->1428->1454->4996->4504->1
604->1147->4618->2303->4501->2601->623->1295->3902->1950->3012
Maximum Bandwidth with dijkstra with heap:  6142
Time taken:  0.1874547004699707
3963->3407->4056->2074->2551->383->4120->3892->2622->539->4396->4550->3054->4386->2653->
4448->1099->2812->2742->954->4262->3787->1746->3513->4320->1380->4665->3119->3128->3012
Maximum Bandwidth with kruskal:  6142
Time taken:  1.2537586688995361
3585->3080->1052->1788->4276->997->136->3452->4126->871->1240->1545->1479->1263->1931->2
388->3899->1798->2218->3912->1591->1447->416->590->3443->815->1253->1851->2304->4496->51
1->1393->4764->1849->844->937->3875->1331->1147->4618->2554->1695->1231->1444->1127->229
6->1526->4683->2430->3535->3782->4620->2731->96->4110->3157->470->1964->3869->838->3084-
>3824->3918->3720->3972->2637->3002->1802->75
Maximum Bandwidth with dijkstra:  5465
Time taken:  2.3053908348083496
3585->3080->1052->1788->4276->997->136->3452->4126->871->1240->1545->1479->1263->1931->2
388->3899->1798->2218->3912->1591->1447->416->590->3443->815->1253->1851->2304->4496->51
1->1393->4764->1849->844->937->3875->1331->1147->4618->2554->1695->1231->1444->1127->229
6->1526->1406->71->1304->2991->3002->1802->75
Maximum Bandwidth with dijkstra with heap:  5465
Time taken:  0.3124122619628906
3585->3080->1052->1788->4276->997->136->3452->4126->871->1240->1545->1479->1263->1931->2
388->3899->1798->2218->3912->1591->1447->416->590->3443->815->1253->1851->2304->4496->51
1->1393->4764->1849->844->937->3875->1331->1147->4618->2554->1695->1231->1444->1127->229
6->1526->4683->2430->3535->3782->4620->2731->96->4110->3157->470->1964->3869->838->3084-
>3824->3918->3720->3972->2637->3002->1802->75
Maximum Bandwidth with kruskal:  5465
Time taken:  1.2860147953033447
3258->4362->4332->3406->3831->4854->4193->3958->4006->3228->3133->4876->3285->4827->3359
->3622->3688->3861->3684->3682->4237->4826->3949->3375->4336->4337->3354->4291->4736->41
52->4839->4411->4176->4641->3623->4167->4454->4077->3349->4982->3452->3002->3653->3180->
3689->4071->4474->4988->1511
Maximum Bandwidth with dijkstra:  4004
Time taken:  1.7588253021240234
3258->4362->4332->3406->1231->1695->2554->4618->1511
Maximum Bandwidth with dijkstra with heap:  4004
Time taken:  0.22053074836730957
3258->4362->4332->3406->1231->1444->1127->2296->1526->4683->2430->3535->3782->4620->2731
->96->4110->3157->470->1964->3869->838->1124->1511
Maximum Bandwidth with kruskal:  4004
Time taken:  1.282038927078247
2421->4167->4454->1881->4972->4222->3266->2454->2581->2977->2810->3555->4437->4333->4190
->3342->2840->3741->3324->3749->4315->3876->3507->4113->2856->3706->3689->4071
Maximum Bandwidth with dijkstra:  5545
Time taken:  0.03124213218688965
2421->4167->4454->1881->841->333->4823->424->4927->2768->231->4433->2102->1219->3010->40
32->525->4451->2577->4038->3353->477->4300->3899->1798->2218->1440->269->1173->2617->189
3->1069->4071
Maximum Bandwidth with dijkstra with heap:  5545
Time taken:  0.20307016372680664
2421->4167->4454->1881->841->333->4823->424->4927->2768->231->4433->2102->1219->3010->40
32->525->4451->2577->4038->3353->1658->686->2198->2090->1621->2882->4249->1965->1948->18
16->1972->1373->4670->218->3959->2812->1099->4448->2653->4386->3054->4550->4396->580->41
88->914->3914->836->4440->477->912->3276->3056->4931->626->248->4071
Maximum Bandwidth with kruskal:  5545
Time taken:  1.2989253997802734
1588->1374->2154->1193->1896->2506->4800->1889->2502->1030->44->4154->3161->2441->3930->
3820->2519->2798->1740->3575->290->3850->2261->931->3416->3050->2622->3748->2292->4002->
```

```
3860->1921->3622->4309->2228->187->1185->3833->3697->398->3127->2207
Maximum Bandwidth with dijkstra:  7674
Time taken:  0.8805990219116211
1588->1374->2154->1193->1896->2506->4800->1889->2502->1030->44->4154->3161->2441->3930->
3820->3195->1805->3969->2111->4864->2382->3239->1048->3434->3158->3556->398->3127->2207
Maximum Bandwidth with dijkstra with heap:  7674
Time taken:  0.13329172134399414
1588->1374->2154->1193->1896->2506->4800->1889->2502->1030->44->4154->3161->3776->161->4
558->3748->2292->4002->3860->1921->3622->4309->2228->187->1185->3833->3697->398->3127->2
207
Maximum Bandwidth with kruskal:  7674
Time taken:  1.4600849151611328
1348->1548->3911->3339->2385->1380->4698->4301->1256->4317->2053->2203->1951->185->3966-
>1344->1715->4210->3515->3220->4700->2362->2127->3188->867->221->2247->1992->3727->2251-
>2548->797->3935->2699->1920->1489->2882->4435->4115->3189->3501->2964->2773->1918->2285
->4580->4653->2704->1235->932->344->3525->3776->3161->2441->3930->3820->2519->2798->1740
->3575->290->3850->2261->931->3416->3050->2622->3748->4558->3019->3534->3740->4851->2225
->4982->4915->1768->2852->3921->1675->4133->589->579->3069->3782->2281->1469->137->1550-
>3350->2783->1415->1723->3754->1624->4561->2313->2014->4363->4235->3950->889->363->2478-
>3815->250->1562->4911->144
Maximum Bandwidth with dijkstra:  3254
Time taken:  2.7496871948242188
1348->1548->3911->3339->2385->1380->4698->4301->1256->4317->2053->2203->1951->185->3966-
>1344->3889->1071->286->4422->409->2506->4800->1889->2502->1030->44->4154->3161->3776->1
61->4558->3019->3534->3740->4851->2225->4982->4915->1768->2852->3921->1675->4133->589->5
79->3069->3782->2281->1469->137->1550->3350->2783->1415->1723->3754->1624->4561->2313->2
014->4363->4235->3950->889->363->2478->3815->250->1562->4911->144
Maximum Bandwidth with dijkstra with heap:  3254
Time taken:  0.34070420265197754
1348->1548->3911->3339->2385->1380->4698->4301->1256->4317->2053->2203->1951->185->3966-
>1344->3889->1071->286->4422->409->2506->4800->1889->2502->1030->44->4154->3161->3776->1
61->4558->3019->3534->3740->4851->2225->4982->4915->1768->2852->3921->1675->4133->589->5
79->3069->3782->2281->1469->137->1550->3350->2783->1415->1723->3754->1624->4561->2313->2
014->4363->4235->3950->889->363->2478->3815->250->1562->4911->144
Maximum Bandwidth with kruskal:  3254
Time taken:  1.3406047821044922
3519->3857->4967->355->4987->3639->2953->1771->3247->169->393->3689->2964->2773->1918->2
285->4580->4653->2704->1235->932->344->229->4334->1965->816->862->3589->2064->1692->3708
->1943->4666->2921->2853->2956->1930->2419->2643->880->3672->19
Maximum Bandwidth with dijkstra:  6087
Time taken:  2.4697651863098145
3519->3857->4967->355->4987->3639->2953->1771->3247->169->393->3689->2964->2773->4010->3
997->2302->2110->164->2947->1892->1069->4673->2707->4935->1110->474->880->3672->19
Maximum Bandwidth with dijkstra with heap:  6087
Time taken:  0.3370952606201172
3519->3857->4967->355->4987->3639->2953->1771->3247->169->1557->560->3889->1344->1715->4
5->1060->1901->40->3953->649->366->393->3689->2964->2773->4010->3997->2302->2110->164->2
947->1892->1069->4673->2707->4935->1110->474->880->3672->19
Maximum Bandwidth with kruskal:  6087
Time taken:  1.365389108657837
2694->1463->1639->3590->4794->1212->3333->4465->2808->1398->2288->2774->4032->1777->3836
->3104->4183->4545->4735->4696->1469->4943->3687->2064->1692->3708->1943->4666->2921->28
53->2956->1930->2419->2643->880->3168->4505->4543->1970->4055->2055->1941->4230->4615->4
508->521->1954->1458->4330->523->4370->3645->1602->2163->1613->1612
Maximum Bandwidth with dijkstra:  931
Time taken:  2.5064520835876465
2694->1463->1639->3590->4794->1212->3333->4465->2808->1398->485->4055->1970->4543->4505-
>3168->880->474->1110->4935->2707->4673->1069->1892->2947->164->2110->2302->63->2742->10
21->4349->1012->4574->2656->3131->3757->1964->2238->3046->1464->4746->2376->4103->3296->
1602->2163->1613->1612
Maximum Bandwidth with dijkstra with heap:  931
Time taken:  0.3091249465942383
2694->1463->1639->3590->4794->1212->3333->4465->2808->1398->485->4055->1970->4543->4505-
>3168->880->474->1110->4935->2707->4673->1069->1892->2947->164->2110->2302->63->2742->10
21->4349->1012->4574->2656->3131->3757->1964->2238->3046->3414->4286->2376->25->1418->42
```

```
72->1063->2163->1613->1612
Maximum Bandwidth with kruskal:  931
Time taken:  1.2289345264434814
660->4838->2200->4269->3159->2427->759->886->848->1692->3708->1943->4666->2921->2853->29
56->1930->2419->2643->880->474->1110->4935->2707->404
Maximum Bandwidth with dijkstra:  7749
Time taken:  0.09372806549072266
660->4838->2200->4269->3159->2427->759->886->848->1692->3708->1943->4666->2921->2853->29
56->1930->2419->2643->880->474->1110->4935->2707->404
Maximum Bandwidth with dijkstra with heap:  7749
Time taken:  0.07799506187438965
660->4838->2200->4269->3159->2427->759->886->848->1692->2064->3589->862->816->1965->4334
->229->344->3525->3776->3161->4154->44->1030->2502->1889->4800->2506->409->4422->286->10
71->3889->1344->1715->45->1060->1901->40->3953->649->366->393->3689->2964->2773->4010->3
997->2302->2110->164->2947->1892->1069->4673->2707->404
Maximum Bandwidth with kruskal:  7749
Time taken:  1.300628662109375
```

In [35]:

```
print("For Dense Graph")
print(" ")
for k in range(1,5):
    dense_graph = create_rand_graph(5000, 0.20)
    for i in range(0,5):
        source = random.randint(1,4999)
        target = random.randint(1,4999)


        strt = time.time()
        result = apply_dijkstra(dense_graph[0], dense_graph[0][source], dense_graph[0][
        endt = time.time()
        print("Maximum Bandwidth with dijkstra: ",result)
        print("Time taken: ",endt-strt)


        strt = time.time()
        result = apply_dijkstra_with_heap(dense_graph[0], dense_graph[0][source], dense
        endt = time.time()
        print("Maximum Bandwidth with dijkstra with heap: ",result)
        print("Time taken: ",endt-strt)


        strt = time.time()
        result = apply_kruskal(dense_graph, source, target)
        endt = time.time()
        print("Maximum Bandwidth with kruskal: ",result)
        print("Time taken: ",endt-strt)
```

```
For Dense Graph

4660->1467->4990->4963->3265->3784->3598->1843->2156->3471->4003->657->4122->1185->671->
859->2604->3344->968->3012->1767
Maximum Bandwidth with dijkstra:  9983
Time taken:  3.9455325603485107
4660->1467->4990->196->2596->1609->3871->4354->4361->126->2184->502->3445->951->3086->11
92->3104->3778->1749->1523->1010->186->1157->2024->1763->3342->653->998->1773->1314->257
8->3828->4744->4110->22->658->11->3793->3845->3331->370->432->1046->2050->3215->613->487
4->3186->1905->3528->1769->3722->2181->561->791->968->3012->1767
Maximum Bandwidth with dijkstra with heap:  9983
Time taken:  2.726890802383423
4660->1467->4990->196->2596->1609->3871->4354->4361->126->2184->502->3445->951->3086->11
92->3104->3778->1749->1523->1010->186->1157->2024->1763->3342->653->998->1773->1314->257
8->3828->4744->4110->22->658->1961->4978->3599->1898->793->2421->1958->2264->1624->3134-
```

```
>3977->1261->846->3096->3603->220->2178->1767
Maximum Bandwidth with kruskal:  9983
Time taken:  250.5055913925171
4335->3017->4969->264->2363->1234->1525->475->1813->4674->867->1123->4966->2496->427->98
7
Maximum Bandwidth with dijkstra:  9985
Time taken:  3.933016300201416
4335->2146->3665->1994->3670->1505->4593->410->222->702->799->4886->1738->443->4978->196
1->658->22->4110->1495->3335->322->3061->3982->3694->1813->4674->867->1123->4966->2496->
427->987
Maximum Bandwidth with dijkstra with heap:  9985
Time taken:  3.161238431930542
4335->3017->4969->264->2363->1234->1525->475->1813->4674->867->1123->4966->2496->427->98
7
Maximum Bandwidth with kruskal:  9985
Time taken:  235.2939715385437
3230->3967->1129->1589->4759->4862->4312->1897->2314->3238->4282->4856->4638->1437->4255
->1817->4319->2118->4144->3492->3173->2948->2905->3680->3809->3052->4510->4938->1823->40
04->231->1270->2442->1522->2693
Maximum Bandwidth with dijkstra:  9980
Time taken:  4.647265195846558
3230->3967->1129->1589->4133->4548->315->4134->2067->3719->1618->506->105->922->3269->23
68->961->2941->1521->2586->295->187->989->2477->2282->4898->2828->4253->716->4476->1430-
>1243->4914->3081->3194->2663->1281->1987->3520->4041->3943->2803->138->1242->450->1394-
>2579->779->641->3997->1488->4996->3699->3855->3338->2664->279->4965->4828->3488->110->4
623->3715->566->1200->3446->1309->231->1270->2442->1522->2693
Maximum Bandwidth with dijkstra with heap:  9980
Time taken:  2.7126219272613525
3230->3967->1129->3430->4651->446->184->457->1626->2990->403->3190->2656->2837->872->311
2->4241->1555->961->2941->1521->2586->295->187->989->2477->2282->4898->2828->493->47->48
84->229->850->2918->77->527->2417->465->2204->1032->675->4616->1010->186->1157->2024->17
63->3342->653->998->1773->1314->2578->3828->4744->4110->22->658->1961->4978->3599->279->
4965->4828->3488->110->4623->3715->566->1200->3446->1309->231->1270->2442->1522->2693
Maximum Bandwidth with kruskal:  9980
Time taken:  230.34096121788025
710->4239->3975->1662->3907->2081->1914->2563->3636->3667->3933->599->2761->1785->4293->
1657->2739->4261->4233->2871->3896->4853->3825
Maximum Bandwidth with dijkstra:  9987
Time taken:  1.516162395477295
710->4239->3975->777->4446->3995->658->11->3793->3845->3331->370->2753->2435->4661->677-
>3883->3343->4135->120->731->416->2739->4261->4233->2871->3896->4853->3825
Maximum Bandwidth with dijkstra with heap:  9987
Time taken:  1.1571824550628662
710->4239->3975->777->47->4884->229->850->2918->77->527->2417->465->2204->1032->675->461
6->2347->254->599->2761->1785->4293->1657->2739->4261->4233->2871->3896->4853->3825
Maximum Bandwidth with kruskal:  9987
Time taken:  231.04006791114807
83->343->3328->4002->3014->4190->4174->4198->2270->3422->2798
Maximum Bandwidth with dijkstra:  9973
Time taken:  5.235710144042969
83->343->3328->3303->359->3612->883->3972->3170->4->1883->3314->4624->2066->3400->2849->
4796->3994->4226->4893->1122->286->4590->1396->3440->467->32->4126->1013->1944->2184->12
6->4361->1384->4026->1833->2715->3237->1220->2572->539->3190->2656->2837->872->3112->424
1->1555->961->2941->1521->2586->295->187->989->2477->2282->4898->2828->4253->716->4476->
1430->3174->4933->3422->2798
Maximum Bandwidth with dijkstra with heap:  9973
Time taken:  2.9458861351013184
83->343->1074->427->2496->4966->1123->867->4674->1813->3694->3982->3061->322->3335->1495
->4110->4744->3828->2578->1314->1773->998->653->3342->1763->2024->1157->186->1010->4616-
>675->1032->2204->465->2417->527->77->2918->850->229->4884->47->493->3301->2916->1719->1
089->979->1119->4652->4099->4675->2909->118->36->1730->4314->3796->4030->3422->2798
Maximum Bandwidth with kruskal:  9973
Time taken:  230.6040906906128
403->4665->2079->31->3540->4712->3485->3823->3770->1680->1421->4074->3400->4405->2304->4
779->4304->1729->3829->4471->2648->4199->3790->4607->3427->2850->3415->3517->1630->1761-
```

```
>2026->4139->3910->3391->4750->3405->3092->2714->2371->2756->4215->2557->3263->3049->226
0->3830->4349->3566->2419->4206->3165->4944->2523->4152->1948->4158->3859->1916->1831->1
208->2638->1047->4426->1768->2678
Maximum Bandwidth with dijkstra:  9984
Time taken:  2.9996070861816406
403->4665->2079->31->3540->4712->642->3507->1177->4052->490->3386->3418->4694->1889->930
->322->2972->4187->3583->2953->4171->1973->39->648->2154->1403->4419->507->4680->695->49
34->3967->2237->1397->892->560->2638->1047->4426->1768->2678
Maximum Bandwidth with dijkstra with heap:  9984
Time taken:  0.6991534233093262
403->4665->2079->31->3540->4712->642->3507->1177->2987->747->23->3647->2044->4461->990->
4293->1473->4443->4209->1542->3359->4343->369->3975->320->3928->2480->3913->3649->588->1
916->1831->1208->2638->1047->4426->1768->2678
Maximum Bandwidth with kruskal:  9984
Time taken:  229.00215816497803
1356->4057->294->4104->1401->3386->3149->1490->2074->2984->974->4131->3268->211->1247->1
232->4858->3936->635->2886->1626->299->795->2796->1810->1270->4328->1294->3415->2850->34
27->513->4134->4657->1948->4158->2614
Maximum Bandwidth with dijkstra:  9988
Time taken:  2.53818678855896
1356->3787->4446->848->4694->1889->930->322->2972->4187->3583->2953->4171->1973->39->648
->2154->1403->4419->507->4680->695->4934->3967->2237->1397->892->560->2638->1208->1831->
1916->588->3649->3913->2480->3928->320->3975->649->1257->4169->3697->4431->1780->1267->4
017->4726->536->4721->500->3242->3094->1102->795->2796->1810->1270->4328->1294->3415->28
50->3427->513->4134->4657->1948->4158->2614
Maximum Bandwidth with dijkstra with heap:  9988
Time taken:  1.39595365524292
1356->4057->294->4104->1401->3386->3149->1490->2074->2984->974->4131->3268->211->1247->1
232->4858->3936->635->2886->1626->299->795->2796->1810->1270->4328->1294->3415->2850->34
27->513->4134->4657->1948->4158->2614
Maximum Bandwidth with kruskal:  9988
Time taken:  234.79291987419128
3931->3671->4775->3256->3125->4976->3317->4484->3102->3446->3213->3976->4191->3592->4185
->3859->4454->3663->4941->3559->2296->3463->2313->2641->2974->2002->751->4623
Maximum Bandwidth with dijkstra:  9963
Time taken:  4.828453063964844
3931->3671->2260->3830->4349->3566->2419->53->4200->658->4605->4953->1934->2300->2912->4
899->2725->4469->1973->4171->2953->3583->4187->2972->322->1673->3903->1091->3883->4104->
1401->3386->490->2734->2856->4836->877->1002->2974->2002->751->4623
Maximum Bandwidth with dijkstra with heap:  9963
Time taken:  1.9212427139282227
3931->3671->2260->3830->4349->3566->2419->53->4200->658->4605->4953->1934->2300->2912->4
899->2725->4469->1973->39->648->2154->1403->4419->507->4680->695->4934->3967->2237->1397
->892->560->2638->1208->1831->1916->588->3649->3913->2480->3928->320->3975->649->1257->4
169->3697->4431->1780->1267->4017->4726->536->4721->500->3242->3094->1102->795->299->162
6->2886->635->3936->4858->1232->1247->211->3268->4131->974->2984->2074->1490->1037->2116
->1965->3022->696->1295->2002->751->4623
Maximum Bandwidth with kruskal:  9963
Time taken:  238.01468563079834
4732->3699->4481->1758->3233->392->4084->1086->1812->1152->2566->2128->12->4235->3127->8
26->3216->3402->1596->3811->4288->685->725->4660->1699->4103->2408->1912->2501
Maximum Bandwidth with dijkstra:  9987
Time taken:  3.104135036468506
4732->3699->4481->1758->3233->3783->4664->299->795->1102->3094->3242->500->4721->536->47
26->4017->1267->1780->4431->3697->4169->1257->649->3975->369->4343->3359->1542->4209->44
43->1473->4293->990->4461->2044->3647->23->836->1667->2949->2433->1309->2120->834->4818-
>1029->4354->2691->4616->1383->4279->2819->1560->2291->654->3133->3508->3846->3464->1524
->2554->329->2501
Maximum Bandwidth with dijkstra with heap:  9987
Time taken:  1.6609513759613037
4732->3699->4481->1758->3233->3783->4664->299->1626->2886->635->3936->4858->1232->1247->
211->3268->4131->974->2984->2074->1490->3149->3386->3418->4694->1889->930->3970->1485->1
231->2125->4978->186->2221->4725->1476->4950->3441->4932->651->2231->218->1560->2291->65
4->3133->3508->3846->3464->1524->2554->329->2501
Maximum Bandwidth with kruskal:  9987
```

```
Time taken:  247.67233419418335
314->3269->618->4815->47->1534->1672->2603->1691->3965->3943->2716->3297->3733->3487->48
94->2530->3102->3446->2249->2908->4865->3870->2069->2133->2178->4382->4642->3134->4637->
2670->3611->2523->4944->3165->4206->2419->3566->4349->3830->2260->3049->3263->2557->4215
->2756->2371->2714->3092->3405->4750->3391->3910->4139->2026->1761->1630->3517->3415->25
91->4379->4952->2460->4196->4664->3783->3233->1758->4481->3699->4732->4574->2222->1476->
4725->380->2517->3414->4460
Maximum Bandwidth with dijkstra:  9979
Time taken:  6.0034239292144775
314->3269->618->4815->47->1534->1672->4999->487->23->3647->2044->4461->990->4293->1473->
4443->4209->1542->3359->4343->369->3975->649->1257->4169->2732->4766->186->2221->4725->3
80->2517->3414->4460
Maximum Bandwidth with dijkstra with heap:  9979
Time taken:  3.2184622287750244
314->3269->618->4815->47->165->4638->2756->4215->2557->3263->3049->2260->3830->4349->356
6->2419->53->4200->658->4605->4953->1934->2300->2912->4899->2725->4469->1973->39->648->2
154->1403->4419->507->4680->695->4934->3967->2237->1397->892->560->2638->1208->1831->191
6->588->3649->3913->2480->3928->320->3975->649->1257->4169->3697->4431->1780->1267->4017
->4726->536->4721->500->3242->3094->1102->795->299->1626->2886->635->3936->4858->1232->1
247->211->3268->4131->974->2984->2074->1490->3149->3386->3418->4694->1889->930->3970->14
85->1231->2125->4978->186->2221->4725->380->2517->3414->4460
Maximum Bandwidth with kruskal:  9979
Time taken:  235.95013999938965
1796->46->2189->2465->2537->3361->2192->2473->3105->4500->4210->2603->4376->3189->4646->
4308->3973->2621->3446->4727->3654->4614->3663->4233->3551->3059->3699->3792->4396->3038
->3711->3241->4411->3473->3584->2713->2922->4351->3897->4110->2416->2909->3913->2517->49
87->3204->4397->1307->25
Maximum Bandwidth with dijkstra:  9979
Time taken:  5.501233339309692
1796->46->2189->4837->399->4244->3209->4988->3602->2067->1614->2550->4224->1386->2819->1
751->4018->632->2891->94->4495->2686->1835->1564->4953->2175->3501->4615->2079->4978->40
07->3932->1370->4382->641->1692->1037->2660->3143->2968->3016->87->127->1307->25
Maximum Bandwidth with dijkstra with heap:  9979
Time taken:  1.1882059574127197
1796->46->2189->4837->78->656->4800->3178->255->1883->4046->1173->995->1663->1588->581->
1305->1749->3174->1959->2990->1948->3295->1474->4079->2244->1307->25
Maximum Bandwidth with kruskal:  9979
Time taken:  236.86822509765625
4605->437->3886->4055->3269->1586->3861->3130->1037->2660->3143->2968->1024->857->1160->
3709->1423->2049->514->3776->3475->3687->162->2136->3432->431->1167->3960
Maximum Bandwidth with dijkstra:  9984
Time taken:  4.62956166267395
4605->437->3886->4055->3269->1586->3861->3130->1037->2660->3143->2968->1024->857->1160->
3709->1423->2049->514->3776->3475->3687->162->2136->3432->431->1167->3960
Maximum Bandwidth with dijkstra with heap:  9984
Time taken:  2.611456871032715
4605->437->3886->4055->1857->2882->3240->2599->465->4382->1370->3932->4007->4978->2079->
4615->1190->3016->2968->1024->857->1160->3709->1423->2049->514->3776->3475->3687->162->2
136->3432->431->1167->3960
Maximum Bandwidth with kruskal:  9984
Time taken:  235.63864541053772
3951->3648->4014->3208->4170->4102->3669->3596->4359->3791->3766->4436->3615->4693->3864
->3632->3503->4838->4384->4335->3738->4829->4691->3590->4524->4860->4210->4816->4716->44
38->3866->4204->4896->3540->3846->3589->4223->4178->4539->4839->4969->4955->3720->4882
Maximum Bandwidth with dijkstra:  9963
Time taken:  0.25467491149902344
3951->3648->2439->2473->3105->2971->3036->1740->1264->2952->3449->3657->2832->1018->1780
->4720->3391->1044->447->3151->4361->2763->1347->4192->1200->4600->2666->86->421->2719->
1207->3312->1952->1977->4907->3562->409->102->2237->150->1051->2571->1694->4028->1541->1
189->4338->4882
Maximum Bandwidth with dijkstra with heap:  9963
Time taken:  2.9650702476501465
3951->3648->2439->2473->3105->1565->4397->1307->2244->4079->1474->4615->3501->2175->4953
->1564->1835->2686->2449->2954->2020->1873->1584->987->396->3095->3544->1472->2683->1449
->1235->2292->4177->2493->4437->2622->3933->1276->3720->4882
```

```
Maximum Bandwidth with kruskal:  9963
Time taken:  238.49475240707397
1754->4951->4694->1018->2832->3657->3449->4078->4846->2058->3361->2537->2465->2155->4818
->3295->1948->2990->1959->3174->4763->3297->1689->2166->1667->1628->1583->1527->2703->26
85->2146->4043->2908->3397->4295->1615->1559->2405->3045->2220->4894->4592->4518->3182->
2176->2343->4633->1865->2763->4361->2197->4122->3702->2948->3787->3653->3055->2028->4556
->4301->2130->1613->4762->3772->2787->4780->2954->2020->4786->3556->3318
Maximum Bandwidth with dijkstra:  9984
Time taken:  1.1132786273956299
1754->4951->4694->4975->446->3921->2963->597->3776->1478->689->3532->4700->315->3242->29
66->3719->2114->2562->2197->4361->3151->447->2752->3211->321->312->4864->2503->998->3237
->149->131->580->2256->198->4874->437->3886->4055->3269->1586->3861->3130->1037->2660->3
143->2968->3016->1190->4615->3501->2175->4953->1564->1835->2686->2449->2954->2020->4786-
>3556->3318
Maximum Bandwidth with dijkstra with heap:  9984
Time taken:  0.41214585304260254
1754->4951->4694->1018->3089->3708->400->2389->255->1883->4046->1173->995->1663->1588->5
81->1305->1749->3174->1959->2990->1948->3295->1474->4615->3501->2175->4953->1564->1835->
2686->2449->2954->2020->4786->3556->3318
Maximum Bandwidth with kruskal:  9984
Time taken:  259.27004528045654
800->2429->481->3748->3675->4811->3815->899->2079->4978->4007->3932->4412->3045->2405->1
559->1615->4295->3397->1547->2774->4523->906->3336->3789->3306
Maximum Bandwidth with dijkstra:  9973
Time taken:  5.120205402374268
800->1689->3297->576->2799->1661->3901->382->1625->3813->65->779->4687->4696->1824->1615
->4295->3397->1547->2774->4523->906->3336->3789->3306
Maximum Bandwidth with dijkstra with heap:  9973
Time taken:  3.0651752948760986
800->2429->481->3748->3675->4811->3815->899->2079->4615->1190->79->1292->418->508->4364-
>2861->745->2667->4320->4869->2650->4203->4755->1126->4987->3204->2314->134->3247->2156-
>230->397->1923->3259->1983->4147->2481->897->383->4557->3002->792->2387->3256->2929->24
43->4523->906->3336->3789->3306
Maximum Bandwidth with kruskal:  9973
Time taken:  240.83807849884033
306->3071->820->1921->4701->1773->4717->4728->3599->307->1210->4533->4366->4204->1905->1
634->1319->800->4991->4432->2470->592->3491->4157->3642->3377->2316->1228->2175->2875->3
489->4022->3428->865->1651->3291->2397->1062->1353->1086->697->2365->2548->2104->1883->2
636->2511->4137->1201->2332->3140->4883->3454->1981->2617->1400->1275->1751->1630->3990-
>2028->2880->1373->4663->743->4172->777->2850->2392->483->4450->3523->4091->3890->4600->
2942->2211
Maximum Bandwidth with dijkstra:  9977
Time taken:  5.112835645675659
306->3071->820->1921->4701->1773->531->66->1724->3214->526->4661->255->4293->418->2907->
4184->115->20->367->4220->1953->3912->3838->1620->4900->4410->672->3760->2369->2567->282
8->1221->440->3289->4524->678->4107->1954->4150->1941->1167->4053->750->55->4848->907->2
789->1130->4738->1143->745->1969->3250->2787->2040->3998->2330->2286->1216->3621->3393->
2870->4397->1668->4408->1610->777->2850->2392->483->4450->3523->4091->3890->4600->2942->
2211
Maximum Bandwidth with dijkstra with heap:  9977
Time taken:  3.465949296951294
306->3071->820->1921->4701->1773->531->66->1724->3214->526->4661->255->4293->418->2907->
4184->115->20->367->4220->3030->3891->4578->1369->4830->277->935->1959->328->3235->1243-
>4606->2358->639->4653->3236->3164->3267->4067->1062->1353->1086->697->2365->2548->2104-
>1883->2636->2511->4137->1201->2332->3140->4883->3454->1981->2617->1400->1275->1751->163
0->3990->2028->2880->1373->4663->743->4172->777->2850->2392->483->4450->3523->4091->3890
->4600->2942->2211
Maximum Bandwidth with kruskal:  9977
Time taken:  242.11258673667908
4302->4614->38->2910->1799->3000->4073->4607->4022->3428->1966->3128->3573->1860->4454->
3036->1206
Maximum Bandwidth with dijkstra:  9983
Time taken:  2.5847113132476807
4302->4614->38->2910->1799->3000->4073->4607->4022->3428->865->1651->3291->2397->1062->4
067->3267->3164->3236->4653->639->2358->4606->1243->3235->328->1959->935->277->4830->136
```

```
9->4578->3891->3030->4220->1953->3912->3838->1620->4900->4410->672->3760->2369->2567->28
28->1221->440->3289->4524->678->4107->1954->4150->1941->1514->3036->1206
Maximum Bandwidth with dijkstra with heap:  9983
Time taken:  2.2156100273132324
4302->4614->38->2910->1799->3000->4073->4607->4022->3428->865->995->4669->4338->4916->53
8->2125->1381->3006->3919->4254->2170->2->4355->4105->1056->2423->1668->4408->1610->777-
>4172->743->4663->1373->2880->2028->3990->1630->1751->1275->1400->2617->1981->3454->4883
->3140->2332->1201->4137->2511->2636->1883->2104->2548->2365->697->1086->1353->1062->406
7->3267->3164->3236->4653->639->2358->4606->1243->3235->328->1959->935->277->4830->1369-
>4578->3891->3030->4220->1953->3912->3838->1620->4900->4410->672->3760->2369->2567->2828
->1221->440->3289->4524->678->4107->1954->4150->1941->1167->4053->750->55->4848->907->27
89->1130->4738->1143->745->1969->3250->4740->1849->662->830->1689->1206
Maximum Bandwidth with kruskal:  9983
Time taken:  249.87706303596497
2074->4023->4568->3720->1519->742->1214->3144->4029->3727->2763->3174->4122->1340->2536-
>4074->4137->1739->3590->1625->3497->1722->2166->4581->4613->4748->1521->1557->4107->195
4->4150->3229->2578->2568->2495->3104->4374->3912->1953->4220->3030->3891->2784->4212->3
996->1857->2823->1852->3563->4643->1743->1830->3554->4424->3874->3188->2323->3444->3959-
>1889->3360->2980->4869->2414->4968->2564->3524->4883->4357->3997
Maximum Bandwidth with dijkstra:  9973
Time taken:  5.676219701766968
2074->4023->4568->3720->1519->742->1214->3144->4029->3727->2763->3174->4122->1340->1688-
>2675->935->1959->328->3235->1243->4606->2358->639->4653->3236->3164->3267->4067->1062->
1353->1086->697->2365->2548->2104->1883->2636->2511->4137->1201->2332->3140->4883->4357-
>3997
Maximum Bandwidth with dijkstra with heap:  9973
Time taken:  3.142496109008789
2074->4023->4568->3720->1519->742->1214->3144->4029->3727->2763->3174->4122->1340->1688-
>2675->935->1959->328->3235->1243->4606->2358->639->4653->3236->3164->3267->4067->1062->
1353->1086->697->2365->2548->2104->1883->2636->2511->4137->1201->2332->3140->4883->4357-
>3997
Maximum Bandwidth with kruskal:  9973
Time taken:  246.20138883590698
1814->3377->3642->4157->3491->592->2470->4432->4991->800->1319->1634->1905->4204->4366->
4533->1210->307->3599->4728->4717->981->1212->439->1723->987->146->3310->3957
Maximum Bandwidth with dijkstra:  9988
Time taken:  2.8243658542633057
1814->3377->2316->1228->2175->2875->3489->4022->3428->865->1651->3291->2397->1062->4067-
>3267->3164->3236->4653->639->2358->4606->1243->3235->328->1959->935->2675->1688->1340->
4122->3174->2603->987->146->3310->3957
Maximum Bandwidth with dijkstra with heap:  9988
Time taken:  1.6236376762390137
1814->3377->3642->4157->3491->592->2470->4432->4991->800->1319->1634->1905->4204->4366->
4533->1210->307->3599->4728->4717->981->1212->439->1723->987->146->3310->3957
Maximum Bandwidth with kruskal:  9988
Time taken:  243.10160207748413
50->3025->3569->2613->4269->2633->3833->2325->1554->2598->1545->3376->2152->2128->2833->
2945->2493->3903->2711
Maximum Bandwidth with dijkstra:  9983
Time taken:  1.734266757965088
50->3008->4972->1957->1157->2290->4083->4184->2907->418->4293->255->4661->526->3214->172
4->1332->3750->3664->4255->1281->4160->126->831->881->3053->1025->1376->153->462->3218->
95->1890->2508->3173->1618->449->2235->2839->4580->3582->3732->256->4724->3529->3762->25
70->864->813->2287->1527->4417->3903->2711
Maximum Bandwidth with dijkstra with heap:  9983
Time taken:  1.8989918231964111
50->3025->3569->2613->4269->2633->3833->2325->1554->3495->3914->3034->4181->981->4717->1
773->531->66->1724->3214->526->4661->255->4293->418->2907->4184->115->20->367->4220->195
3->3912->3838->1620->4900->4410->672->3760->2369->2567->2828->1221->440->3289->4524->678
->4107->1954->4150->1941->1167->4053->750->55->4848->907->2789->1130->4738->1143->745->1
969->3250->4740->1849->662->171->2493->3903->2711
Maximum Bandwidth with kruskal:  9983
Time taken:  246.51756620407104
```

In [ ]: