Rapport de la phase 1 du Projet

Membres de l'équipe:

- *Aurélie Boivin-Champeaux*
- *Adam Lebrigui*

Ce rapport montre l'implémentation des différentes parties du code et en discute certains passages.

Node.jl ¶

Un changement est apporté à *node.jl* : nous considérons que les noeuds auront pour nom un entier et non une chaine de caractères

In [2]:

```
import Base.show

"""Type abstrait dont d'autres types de noeuds dériveront."""
abstract type AbstractNode{T} end

"""Type représentant les noeuds d'un graphe.

Exemple:

        noeud = Node(1, [π, exp(1)])
        noeud = Node(2, "guitar")
        noeud = Node(3, 2)

"""
mutable struct Node{T} <: AbstractNode{T}
  name::Int
  data::T
end
```

Out[2]:

```
Base.show
```

Edge.jl

Ensuite, on crée un fichier *edge.jl* qui définie les arêtes entre chaque noeud. La structure Edge considère 3 attributs : deux de type Noeud et un poirds de type entier.

Sa fonction show affiche une phrase indiquant entre quels noeuds se trouve l'arête et quel est son poids.

In [2]:

```
import Base.show

"""Type abstrait dont d'autres types d'arêtes dériveront."""
abstract type AbstractEdge{T} end

"""Type représentant les arêtes d'un graphe.

Exemple:

        arete = (Node1, Node2, 4)
        arete = (Node2, Node3, 1)

"""

mutable struct Edge{T} <: AbstractEdge{T}
  node1::AbstractNode{T}
  node2::AbstractNode{T}
  weight::Int
end

# on présume que tous les noeuds dérivant d'AbstractEdge
# posséderont des champs  'node1', 'node2' et `data`.

"""Renvoie les deux noeuds de l'arête."""
node1Edge(edge::AbstractEdge) = edge.node1;
node2Edge(edge::AbstractEdge) = edge.node2;

"""Renvoie les données contenues dans l'arête."""
weightOnEdge(edge::AbstractEdge) = edge.weight

"""Affiche une arête."""
function show(edge::AbstractEdge)
  messageToPrint = string("Edge from ", name(node1Edge(edge)), " to ", name(node2Edge(e
dge)), ", weighting ", weightOnEdge(edge), ".")
  println(messageToPrint)
end
```

```
UndefVarError: AbstractNode not defined

Stacktrace:
 [1] top-level scope at C:\Users\aurel\.julia\packages\IJulia\fRegO\src\ke
rnel.jl:52
```

Graph.jl

Ci dessous les ajouts au fichier original *graph.jl*. Un attribut de type Vecteur d'objet de type Arête contenant des Noeud de type T est ajouté.

La fonction add_edge! est quasi identique à celle pour les noeuds. Sa fonction show affiche une phrase indiquant combien de noeuds et d'arêtes sont présents.

In [3]:

```
import Base.show

"""Type abstrait dont d'autres types de graphes dériveront."""
abstract type AbstractGraph{T} end

mutable struct Graph{T} <: AbstractGraph{T}
  name::String
  nodes::Vector{Node{T}}
  edges::Vector{Edge{T}}
end

"""Ajoute une arête au graphe."""
function add_edge!(graph::Graph{T}, edge::Edge{T}) where T
  push!(graph.edges, edge)
  graph
end

"""Renvoie la liste des arêtes du graphe."""
edges(graph::AbstractGraph) = graph.edges

"""Renvoie le nombre d'arêtes du graphe."""
nb_edges(graph::AbstractGraph) = length(graph.edges)

"""Affiche un graphe"""
function show(graph::Graph)
  messageToPrint = string("Graph ", name(graph), " has ", nb_nodes(graph), "nodes and",
nb_edges(graph), " edges.")
  println(messageToPrint)
  for node in nodes(graph)
    messageToPrint = string(messageToPrint, "\n", show(node))
  end
  for edge in edges(graph)
    messageToPrint = string(messageToPrint, "\n", show(edge))
  end
  println(messageToPrint)
  return messageToPrint
end
```

```
UndefVarError: Node not defined

Stacktrace:
 [1] top-level scope at C:\Users\aurel\.julia\packages\IJulia\fRegO\src\ke
rnel.jl:52
```

Read_stsp

Read_edges()

On complète ici la fonction *read_edges()* de *read_stsp.jl* afin de lire les poids des arêtes.

Pour ce faire, on définit une variable **weight** de [**j+1**], dans la boucle j, et on l'intègre au contenu de chaque edge (qui contient désormais le noeud source, le noeud destination, ainsi que le poids de l'arête).

Le j+1 vient du fait que la variable j commence à 0 pour récupérer les données de la ligne qu'on étudie. Or en Julia, les tableaux commencent à l'indice 1 et non 0.

In [6]:

```
"""on récupère la valeur du poids de l'arête qui est lue """
            weight = parse(Int64,data[j+1])

            """on récupère les noeuds de l'arête en fonction du type de fichier lu et o
n ajoute le poids"""
            if edge_weight_format in ["UPPER_ROW", "LOWER_COL"]
              edge = (k + 1, i + k + 2, weight)
            elseif edge_weight_format in ["UPPER_DIAG_ROW", "LOWER_DIAG_COL"]
              edge = (k + 1, i + k + 1, weight)
            elseif edge_weight_format in ["UPPER_COL", "LOWER_ROW"]
              edge = (i + k + 2, k + 1, weight)
            elseif edge_weight_format in ["UPPER_DIAG_COL", "LOWER_DIAG_ROW"]
              edge = (i + 1, k + 1, weight)
            elseif edge_weight_format == "FULL_MATRIX"
              edge = (k + 1, i + 1, weight)
            else
              warn("Unknown format - function read_edges")
            end
            push!(edges, edge)
```

Out[6]:

plot_graph

Read_stsp()

Il a fallu changer le type d'edge_list d'Int en tuple d'Int (correspondant au deuxième noeud de l'arête et au poids de l'arête) ainsi que rajouter le poids dans graph_edges.

In [ ]:

```
"""Renvoie les noeuds et les arêtes du graphe."""
function read_stsp(filename::String)

    for k = 1 : dim
    edge_list = Tuple{Int, Int}[]
    push!(graph_edges, edge_list)


  """Le premier noeud de l'arête est représenté par la ligne du tableau edges. Chaque t
uple représente le deuxième noeud de l'arête et son poids associé """
  for edge in edges_brut
    if edge_weight_format in ["UPPER_ROW", "LOWER_COL", "UPPER_DIAG_ROW", "LOWER_DIAG_C
OL"]
      push!(graph_edges[edge[1]], (edge[2], edge[3]))
    else
      push!(graph_edges[edge[2]], (edge[1], edge[3]))
    end
  end
end
```

Main.jl

Pour construire un graphe à partir d'un fichier, nous avons implémenté une fonction construct_graph prenant en argument deux chaines de caractères :

- un nom de fichier
- un nom pour le graphe créé.

Elle se résume en 4 étapes :

1. Lire les données du fichier donné en paramètre grâce à la fonction joinpath qui permet de recréer l'adresse du fichier.
2. On implémente un vecteur de type Node. Si il n'existe pas de données, il vient créer un vecteur de Node avec comme attribut **name** implémenté par un compteur et **data** = 0. Si il existe des données pour les noeuds dans le fichier lu, alors il récupère les données dans le dictionnaire brut_nodes pour les convertir en Node.
3. On implémente un vecteur de type Edge. On fait 2 boucles indentées pour récupérer dans chaque vecteur ligne de brut_edges les tuples de facon séquentielle. On en extrait les données. Le numéro de la ligne donne l'identifiant (Attribut name) du premier noeud de l'arête. Le premier élément du tuple donne l'identifiant du deuxième noeud de l'arête et le deuxième élément du tuple donne le poids de l'arête.
4. On construit l'objet de type graph avec les données obtenues.

In [17]:

```
function construct_graph(filename::String, graph_name::String)
    """Récupérer les données du fichier """
    brut_nodes, brut_edges = read_stsp(joinpath(@__DIR__,"..","..","instances","stsp",f
ilename))
    brut_nodes = sort(brut_nodes) #on trie le tableau des noeuds


    """Pour tous les noeuds du graphe, on crée un objet de type noeud et on l'ajoute au
graphe"""
    nodes = []
    nb_nodes = length(brut_edges)
    #On implémente un tableau de noeuds avec les data vides si il n'y a pas de données
 à récupérer dans le fichier
    if (length(brut_nodes)==0)
        #les noeuds créés auront pour données un entier valant 0
        nodes = Node{Int64}[]
        for i in 1:nb_nodes
            push!(nodes,Node(i,0))
        end
    # Sinon pour chaque noeud, on crée un objet de type Noeud et on l'ajoute dans le ta
bleau
    else
        T = valtype(brut_nodes)
        nodes = Node{T}[]
        for node_id in keys(brut_nodes)
          node = Node(node_id, brut_nodes[node_id])
          push!(nodes, node)
        end
    end

    """Pour toutes les arêtes, on ajoute l'arête avec son poids au tableau d'arêtes"""
    edges = Edge{typeof(nodes[1].data)}[] #tableau d'arêtes à implémenter
    node1 = 1 #numéro de ligne étudiée (ce qui correspond au premier noeud de l'arête)


    #on vient récupérer chaque ligne k du tableau (le numéro de la ligne correspondant
 au nom du noeud) pour lire à quel noeud il est relié et avec quel poids
    for k in brut_edges
        for tuple in k
            edge = Edge(nodes[name_node1], nodes[tuple[1]], tuple[2])
            push!(edges, edge)
        end
        name_node1 = name_node1 + 1
    end

    """Création du graphe avec les données récupérées"""
    graph = Graph(graph_name, nodes, edges)

end
```

```
Reading of header :

SystemError: opening file "localhost:8890/edit/bayg29.tsp": Invalid argume
nt

Stacktrace:
 [1] #systemerror#44(::Nothing, ::typeof(systemerror), ::String, ::Bool) a
t .\error.jl:134
 [2] systemerror at .\error.jl:134 [inlined]
 [3] #open#311(::Bool, ::Nothing, ::Nothing, ::Nothing, ::Nothing, ::typeo
f(open), ::String) at .\iostream.jl:289
 [4] #open at .\none:0 [inlined]
 [5] open(::String, ::String) at .\iostream.jl:345
 [6] read_header(::String) at .\In[6]:6
 [7] read_stsp(::String) at .\In[6]:189
 [8] main() at .\In[17]:10
 [9] top-level scope at In[17]:14
```

Résultats

Par exemple, pour le fichier bays29.tsp, nous obtenons le graphe suivant :

Reading of header : ✓ Reading of nodes : ✓ Reading of edges : ✓ Graph test has 29 nodes. Node 1, data: [1150.0, 1760.0] Node 2, data: [630.0, 1660.0] Node 3, data: [40.0, 2090.0] Node 4, data: [750.0, 1100.0] Node 5, data: [750.0, 2030.0] Node 6, data: [1030.0, 2070.0] Node 7, data: [1650.0, 650.0] Node 8, data: [1490.0, 1630.0] Node 9, data: [790.0, 2260.0] Node 10, data: [710.0, 1310.0] Node 11, data: [840.0, 550.0] Node 12, data: [1170.0, 2300.0] Node 13, data: [970.0, 1340.0] Node 14, data: [510.0, 700.0] Node 15, data: [750.0, 900.0] Node 16, data: [1280.0, 1200.0] Node 17, data: [230.0, 590.0] Node 18, data: [460.0, 860.0] Node 19, data: [1040.0, 950.0] Node 20, data: [590.0, 1390.0] Node 21, data: [830.0, 1770.0] Node 22, data: [490.0, 500.0] Node 23, data: [1840.0, 1240.0] Node 24, data: [1260.0, 1500.0] Node 25, data: [1280.0, 790.0] Node 26, data: [490.0, 2130.0] Node 27, data: [1460.0, 1420.0] Node 28, data: [1260.0, 1910.0] Node 29, data: [360.0, 1980.0] and 406edges. Edge from 1 to 2, weighting 97. Edge from 1 to 3, weighting 205. Edge from 1 to 4, weighting 139. Edge from 1 to 5, weighting 86. Edge from 1 to 6, weighting 60. Edge from 1 to 7, weighting 220. Edge from 1 to 8, weighting 65. Edge from 1 to 9, weighting 111. Edge from 1 to 10, weighting 115. Edge from 1 to 11, weighting 227. Edge from 1 to 12, weighting 95. Edge from 1 to 13, weighting 82. Edge from 1 to 14, weighting 225. Edge from 1 to 15, weighting 168. Edge from 1 to 16, weighting 103. Edge from 1 to 17, weighting 266. Edge from 1 to 18, weighting 205. Edge from 1 to 19, weighting 149. Edge from 1 to 20, weighting 120. Edge from 1 to 21, weighting 58. Edge from 1 to 22, weighting 257. Edge from 1 to 23, weighting 152. Edge from 1 to 24, weighting 52. Edge from 1 to 25, weighting 180. Edge from 1 to 26, weighting 136. Edge from 1 to 27, weighting 82. Edge from 1 to 28, weighting 34. Edge from 1 to 29, weighting 145. Edge from 2 to 3, weighting 129. Edge from 2 to 4, weighting 103. Edge from 2 to 5, weighting 71. Edge from 2 to 6, weighting 105. Edge from 2 to 7, weighting 258. Edge from 2 to 8, weighting 154. Edge from 2 to 9, weighting 112. Edge from 2 to 10, weighting 65. Edge from 2 to 11, weighting 204. Edge from 2 to 12, weighting 150. Edge from 2 to 13, weighting 87. Edge from 2 to 14, weighting 176. Edge from 2 to 15, weighting 137. Edge from 2 to 16, weighting 142. Edge from 2 to 17, weighting 204. Edge from 2 to 18, weighting 148. Edge from 2 to 19, weighting 148. Edge from 2 to 20, weighting 49. Edge from 2 to 21, weighting 41. Edge from 2 to 22, weighting 211. Edge from 2 to 23, weighting 226. Edge from 2 to 24, weighting 116. Edge from 2 to 25, weighting 197. Edge from 2 to 26, weighting 89. Edge from 2 to 27, weighting 153. Edge from 2 to 28, weighting 124. Edge from 2 to 29, weighting 74. Edge from 3 to 4, weighting 219. Edge from 3 to 5, weighting 125. Edge from 3 to 6, weighting 175. Edge from 3 to 7, weighting 386. Edge from 3 to 8, weighting 269. Edge from 3 to 9, weighting 134. Edge from 3 to 10, weighting 184. Edge from 3 to 11, weighting 313. Edge from 3 to 12, weighting 201. Edge from 3 to 13, weighting 215. Edge from 3 to 14, weighting 267. Edge from 3 to 15, weighting 248. Edge from 3 to 16, weighting 271. Edge from 3 to 17, weighting 274. Edge from 3 to 18, weighting 236. Edge from 3 to 19, weighting 272. Edge from 3 to 20, weighting 160. Edge from 3 to 21, weighting 151. Edge from 3 to 22, weighting 300. Edge from 3 to 23, weighting 350. Edge from 3 to 24, weighting 239. Edge from 3 to 25, weighting 322. Edge from 3 to 26, weighting 78. Edge from 3 to 27, weighting 276. Edge from 3 to 28, weighting 220. Edge from 3 to 29, weighting 60. Edge from 4 to 5, weighting 167. Edge from 4 to 6, weighting 182. Edge from 4 to 7, weighting 180. Edge from 4 to 8, weighting 162. Edge from 4 to 9, weighting 208. Edge from 4 to 10, weighting 39. Edge from 4 to 11, weighting 102. Edge from 4 to 12, weighting 227. Edge from 4 to 13, weighting 60. Edge from 4 to 14, weighting 86. Edge from 4 to 15, weighting 34. Edge from 4 to 16, weighting 96. Edge from 4 to 17, weighting 129. Edge from 4 to 18, weighting 69. Edge from 4 to 19, weighting 58. Edge from 4 to 20, weighting 60. Edge from 4 to 21, weighting 120. Edge from 4 to 22, weighting 119. Edge from 4 to 23, weighting 192. Edge from 4 to 24, weighting 114. Edge from 4 to 25, weighting 110. Edge from 4 to 26, weighting 192. Edge from 4 to 27, weighting 136. Edge from 4 to 28, weighting 173. Edge from 4 to 29, weighting 173. Edge from 5 to 6, weighting 51. Edge from 5 to 7, weighting 296. Edge from 5 to 8, weighting 150. Edge from 5 to 9, weighting 42. Edge from 5 to 10, weighting 131. Edge from 5 to 11, weighting 268. Edge from 5 to 12, weighting 88. Edge from 5 to 13, weighting 131. Edge from 5 to 14, weighting 245. Edge from 5 to 15, weighting 201. Edge from 5 to 16, weighting 175. Edge from

5 to 17, weighting 275. Edge from 5 to 18, weighting 218. Edge from 5 to 19, weighting 202. Edge from 5 to 20, weighting 119. Edge from 5 to 21, weighting 50. Edge from 5 to 22, weighting 281. Edge from 5 to 23, weighting 238. Edge from 5 to 24, weighting 131. Edge from 5 to 25, weighting 244. Edge from 5 to 26, weighting 51. Edge from 5 to 27, weighting 166. Edge from 5 to 28, weighting 95. Edge from 5 to 29, weighting 69. Edge from 6 to 7, weighting 279. Edge from 6 to 8, weighting 114. Edge from 6 to 9, weighting 56. Edge from 6 to 10, weighting 150. Edge from 6 to 11, weighting 278. Edge from 6 to 12, weighting 46. Edge from 6 to 13, weighting 133. Edge from 6 to 14, weighting 266. Edge from 6 to 15, weighting 214. Edge from 6 to 16, weighting 162. Edge from 6 to 17, weighting 302. Edge from 6 to 18, weighting 242. Edge from 6 to 19, weighting 203. Edge from 6 to 20, weighting 146. Edge from 6 to 21, weighting 67. Edge from 6 to 22, weighting 300. Edge from 6 to 23, weighting 205. Edge from 6 to 24, weighting 111. Edge from 6 to 25, weighting 238. Edge from 6 to 26, weighting 98. Edge from 6 to 27, weighting 139. Edge from 6 to 28, weighting 52. Edge from 6 to 29, weighting 120. Edge from 7 to 8, weighting 178. Edge from 7 to 9, weighting 328. Edge from 7 to 10, weighting 206. Edge from 7 to 11, weighting 147. Edge from 7 to 12, weighting 308. Edge from 7 to 13, weighting 172. Edge from 7 to 14, weighting 203. Edge from 7 to 15, weighting 165. Edge from 7 to 16, weighting 121. Edge from 7 to 17, weighting 251. Edge from 7 to 18, weighting 216. Edge from 7 to 19, weighting 122. Edge from 7 to 20, weighting 231. Edge from 7 to 21, weighting 249. Edge from 7 to 22, weighting 209. Edge from 7 to 23, weighting 111. Edge from 7 to 24, weighting 169. Edge from 7 to 25, weighting 72. Edge from 7 to 26, weighting 338. Edge from 7 to 27, weighting 144. Edge from 7 to 28, weighting 237. Edge from 7 to 29, weighting 331. Edge from 8 to 9, weighting 169. Edge from 8 to 10, weighting 151. Edge from 8 to 11, weighting 227. Edge from 8 to 12, weighting 133. Edge from 8 to 13, weighting 104. Edge from 8 to 14, weighting 242. Edge from 8 to 15, weighting 182. Edge from 8 to 16, weighting 84. Edge from 8 to 17, weighting 290. Edge from 8 to 18, weighting 230. Edge from 8 to 19, weighting 146. Edge from 8 to 20, weighting 165. Edge from 8 to 21, weighting 121. Edge from 8 to 22, weighting 270. Edge from 8 to 23, weighting 91. Edge from 8 to 24, weighting 48. Edge from 8 to 25, weighting 158. Edge from 8 to 26, weighting 200. Edge from 8 to 27, weighting 39. Edge from 8 to 28, weighting 64. Edge from 8 to 29, weighting 210. Edge from 9 to 10, weighting 172. Edge from 9 to 11, weighting 309. Edge from 9 to 12, weighting 68. Edge from 9 to 13, weighting 169. Edge from 9 to 14, weighting 286. Edge from 9 to 15, weighting 242. Edge from 9 to 16, weighting 208. Edge from 9 to 17, weighting 315. Edge from 9 to 18, weighting 259. Edge from 9 to 19, weighting 240. Edge from 9 to 20, weighting 160. Edge from 9 to 21, weighting 90. Edge from 9 to 22, weighting 322. Edge from 9 to 23, weighting 260. Edge from 9 to 24, weighting 160. Edge from 9 to 25, weighting 281. Edge from 9 to 26, weighting 57. Edge from 9 to 27, weighting 192. Edge from 9 to 28, weighting 107. Edge from 9 to 29, weighting 90. Edge from 10 to 11, weighting 140. Edge from 10 to 12, weighting 195. Edge from 10 to 13, weighting 51. Edge from 10 to 14, weighting 117. Edge from 10 to 15, weighting 72. Edge from 10 to 16, weighting 104. Edge from 10 to 17, weighting 153. Edge from 10 to 18, weighting 93. Edge from 10 to 19, weighting 88. Edge from 10 to 20, weighting 25. Edge from 10 to 21, weighting 85. Edge from 10 to 22, weighting 152. Edge from 10 to 23, weighting 200. Edge from 10 to 24, weighting 104. Edge from 10 to 25, weighting 139. Edge from 10 to 26, weighting 154. Edge from 10 to 27, weighting 134. Edge from 10 to 28, weighting 149. Edge from 10 to 29, weighting 135. Edge from 11 to 12, weighting 320. Edge from 11 to 13, weighting 146. Edge from 11 to 14, weighting 64. Edge from 11 to 15, weighting 68. Edge from 11 to 16, weighting 143. Edge from 11 to 17, weighting 106. Edge from 11 to 18, weighting 88. Edge from 11 to 19, weighting 81. Edge from 11 to 20, weighting 159. Edge from 11 to 21, weighting 219. Edge from 11 to 22, weighting 63. Edge from 11 to 23, weighting 216. Edge from 11 to 24, weighting 187. Edge from 11 to 25, weighting 88. Edge from 11 to 26, weighting 293. Edge from 11 to 27, weighting 191. Edge from 11 to 28, weighting 258. Edge from 11 to 29, weighting 272. Edge from 12 to 13, weighting 174. Edge from 12 to 14, weighting 311. Edge from 12 to 15, weighting 258. Edge from 12 to 16, weighting 196. Edge from 12 to 17, weighting 347. Edge from 12 to 18, weighting 288. Edge from 12 to 19, weighting 243. Edge from 12 to 20, weighting 192. Edge from 12 to 21, weighting 113. Edge from 12 to 22, weighting 345. Edge from 12 to 23, weighting 222. Edge from 12 to 24, weighting 144. Edge from 12 to 25, weighting 274. Edge from 12 to 26, weighting 124. Edge from 12 to 27, weighting 165. Edge from 12 to 28, weighting 71. Edge from 12 to 29, weighting 153. Edge from 13 to 14, weighting 144. Edge from 13 to 15, weighting 86. Edge from 13 to 16, weighting 57. Edge from 13 to 17, weighting 189. Edge from 13 to 18, weighting 128. Edge from 13 to 19,

weighting 71. Edge from 13 to 20, weighting 71. Edge from 13 to 21, weighting 82. Edge from 13 to 22, weighting 176. Edge from 13 to 23, weighting 150. Edge from 13 to 24, weighting 56. Edge from 13 to 25, weighting 114. Edge from 13 to 26, weighting 168. Edge from 13 to 27, weighting 83. Edge from 13 to 28, weighting 115. Edge from 13 to 29, weighting 160. Edge from 14 to 15, weighting 61. Edge from 14 to 16, weighting 165. Edge from 14 to 17, weighting 51. Edge from 14 to 18, weighting 32. Edge from 14 to 19, weighting 105. Edge from 14 to 20, weighting 127. Edge from 14 to 21, weighting 201. Edge from 14 to 22, weighting 36. Edge from 14 to 23, weighting 254. Edge from 14 to 24, weighting 196. Edge from 14 to 25, weighting 136. Edge from 14 to 26, weighting 260. Edge from 14 to 27, weighting 212. Edge from 14 to 28, weighting 258. Edge from 14 to 29, weighting 234. Edge from 15 to 16, weighting 106. Edge from 15 to 17, weighting 110. Edge from 15 to 18, weighting 56. Edge from 15 to 19, weighting 49. Edge from 15 to 20, weighting 91. Edge from 15 to 21, weighting 153. Edge from 15 to 22, weighting 91. Edge from 15 to 23, weighting 197. Edge from 15 to 24, weighting 136. Edge from 15 to 25, weighting 94. Edge from 15 to 26, weighting 225. Edge from 15 to 27, weighting 151. Edge from 15 to 28, weighting 201. Edge from 15 to 29, weighting 205. Edge from 16 to 17, weighting 215. Edge from 16 to 18, weighting 159. Edge from 16 to 19, weighting 64. Edge from 16 to 20, weighting 126. Edge from 16 to 21, weighting 128. Edge from 16 to 22, weighting 190. Edge from 16 to 23, weighting 98. Edge from 16 to 24, weighting 53. Edge from 16 to 25, weighting 78. Edge from 16 to 26, weighting 218. Edge from 16 to 27, weighting 48. Edge from 16 to 28, weighting 127. Edge from 16 to 29, weighting 214. Edge from 17 to 18, weighting 61. Edge from 17 to 19, weighting 155. Edge from 17 to 20, weighting 157. Edge from 17 to 21, weighting 235. Edge from 17 to 22, weighting 47. Edge from 17 to 23, weighting 305. Edge from 17 to 24, weighting 243. Edge from 17 to 25, weighting 186. Edge from 17 to 26, weighting 282. Edge from 17 to 27, weighting 261. Edge from 17 to 28, weighting 300. Edge from 17 to 29, weighting 252. Edge from 18 to 19, weighting 105. Edge from 18 to 20, weighting 100. Edge from 18 to 21, weighting 176. Edge from 18 to 22, weighting 66. Edge from 18 to 23, weighting 253. Edge from 18 to 24, weighting 183. Edge from 18 to 25, weighting 146. Edge from 18 to 26, weighting 231. Edge from 18 to 27, weighting 203. Edge from 18 to 28, weighting 239. Edge from 18 to 29, weighting 204. Edge from 19 to 20, weighting 113. Edge from 19 to 21, weighting 152. Edge from 19 to 22, weighting 127. Edge from 19 to 23, weighting 150. Edge from 19 to 24, weighting 106. Edge from 19 to 25, weighting 52. Edge from 19 to 26, weighting 235. Edge from 19 to 27, weighting 112. Edge from 19 to 28, weighting 179. Edge from 19 to 29, weighting 221. Edge from 20 to 21, weighting 79. Edge from 20 to 22, weighting 163. Edge from 20 to 23, weighting 220. Edge from 20 to 24, weighting 119. Edge from 20 to 25, weighting 164. Edge from 20 to 26, weighting 135. Edge from 20 to 27, weighting 152. Edge from 20 to 28, weighting 153. Edge from 20 to 29, weighting 114. Edge from 21 to 22, weighting 236. Edge from 21 to 23, weighting 201. Edge from 21 to 24, weighting 90. Edge from 21 to 25, weighting 195. Edge from 21 to 26, weighting 90. Edge from 21 to 27, weighting 127. Edge from 21 to 28, weighting 84. Edge from 21 to 29, weighting 91. Edge from 22 to 23, weighting 273. Edge from 22 to 24, weighting 226. Edge from 22 to 25, weighting 148. Edge from 22 to 26, weighting 296. Edge from 22 to 27, weighting 238. Edge from 22 to 28, weighting 291. Edge from 22 to 29, weighting 269. Edge from 23 to 24, weighting 112. Edge from 23 to 25, weighting 130. Edge from 23 to 26, weighting 286. Edge from 23 to 27, weighting 74. Edge from 23 to 28, weighting 155. Edge from 23 to 29, weighting 291. Edge from 24 to 25, weighting 130. Edge from 24 to 26, weighting 178. Edge from 24 to 27, weighting 38. Edge from 24 to 28, weighting 75. Edge from 24 to 29, weighting 180. Edge from 25 to 26, weighting 281. Edge from 25 to 27, weighting 120. Edge from 25 to 28, weighting 205. Edge from 25 to 29, weighting 270. Edge from 26 to 27, weighting 213. Edge from 26 to 28, weighting 145. Edge from 26 to 29, weighting 36. Edge from 27 to 28, weighting 94. Edge from 27 to 29, weighting 217. Edge from 28 to 29, weighting 162.