# Graph Theory 2 Terms

ryamada

2020/5/24

# Neighbor 隣

- When two vertices are connected with an edge, one vertex is a neighbor of the other. Also they are adjacent each other. ２頂点の間に辺がある時、片方の頂点はもう片方の頂点の隣と言う。「隣接しあう」とも言う
- The vertices that have an edge with a vertex V are called “neighbor set of V”. ある頂点Vとの間に辺を持つ頂点は、Vの近傍と言う
- When a vertex is one of two vertices of an edge, the vertex is said to be “incident to” the edge. ある頂点がある辺の２頂点の１つであるとき、その頂点はその辺に「接続している」と言う

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

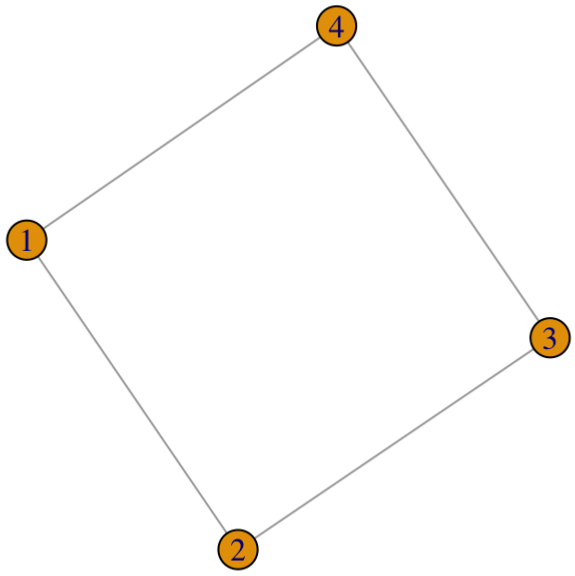
```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
edge.list <- rbind(c(1,2),c(2,3),c(3,4),c(4,1))
edge.list
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    2    3
## [3,]    3    4
## [4,]    4    1
```

```
g1 <- graph.edgelist(edge.list, directed = FALSE) # undirected graph
plot(g1)
```




Neighbor set of V2, Adjacency vertices of V2 近傍・隣接頂点集合

```
neighbors(g1,2)
```

```
## + 2/4 vertices, from a30b8af:
## [1] 1 3
```

```
adjacent_vertices(g1,2)
```

```
## [[1]]
## + 2/4 vertices, from a30b8af:
## [1] 1 3
```

Incident edges 接続辺

```
incident_edges(g1,2)
```

```
## [[1]]
## + 2/4 edges from a30b8af:
## [1] 1--2 2--3
```

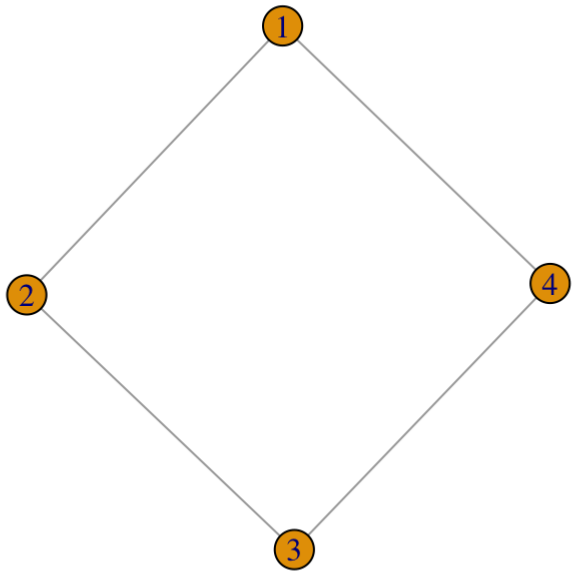
# Degree of vertex 頂点の次数

Degree of vertex is the number of incident edges of the vertex. 頂点の次数とは、頂点に接続する辺の数。

```
adj <- matrix(0,4,4) # 4: no. vertices
adj[1,2] <- adj[2,1] <- adj[2,3] <- adj[3,2] <- adj[3,4] <- adj[4,3] <- adj[4,1] <- adj[1,4] <-
1
adj
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    0    1
## [2,]    1    0    1    0
## [3,]    0    1    0    1
## [4,]    1    0    1    0
```

```
g2 <- graph.adjacency(adj,mode="undirected")
plot(g2)
```




Degree of V3 頂点３の次数

```
degree(g2,3)
```

```
## [1] 2
```

Degrees of all vertices 全ての頂点の次数

```
degree(g2)
```

```
## [1] 2 2 2 2
```

Calculate degree of vertices from an adjacency matrix. 隣接行列から頂点次数を計算する。

Degree of V3 V3の頂点次数

```
sum(adj[3,])
```

```
## [1] 2
```

Degrees of all vertices 全ての頂点の次数

```
apply(adj,1,sum)
```

```
## [1] 2 2 2 2
```

# Size 大きさ

$|V|$ is the number of veritices and $|E|$ is the number of edges.

$|V|$は頂点の数、$|E|$は辺の数

Enumerate edges and vertices

辺と頂点を全列挙する

```
E(g1)
```

```
## + 4/4 edges from a30b8af:
## [1] 1--2 2--3 3--4 1--4
```

```
V(g1)
```

```
## + 4/4 vertices, from a30b8af:
## [1] 1 2 3 4
```

Get $|E|$ and $|V|$ from the enumerated list.

全列挙リストから$|E|$と$|V|$ を出す

```
length(E(g1))
```

```
## [1] 4
```

```
length(V(g1))
```

```
## [1] 4
```

# Walk, trail and path; cycle, ウォーク(歩道)、トレイル、パス; サイクル

# Walk

A walk in a graph is a sequence of edges which joins a sequence of vertices.

グラフのウォークは、共有する頂点を介して途切れることなく繋がる辺の列。

You can walk particular edges and vertices multiple times.

同じ辺・同じ頂点を何度も通ってもウォークである。

```
The edge sequence (1--2, 2--3, 3--2, 2--3, 3--4, 4--1, 1--2) of the square graph is a Walk.
```

# Trails

Trails are walks but the edges in tralis appear no more than once.

トレイルはウォークに含まれるが、トレイルでは、各辺は最大１度までしか歩けない。

# Paths

Paths are walks (and trails) but the vertices in paths appear no more than once.

パスはウォークに含まれる(トレイルにも含まれる)が、各頂点は最大１度までしか歩けない。

All paths from V4 to V2 of g1 is enumerated as;

g1のV4からV2へのパスは以下のように全列挙できる;

The word “simple” in the function all_simple_path() explicitly states “path-in-narrow-sense”.

関数all_simple_path()の中に現れる“simple”と言う単語は、狭義のパスを扱うことを明示している。

```
all_simple_paths(g1,4,2)
```

```
## [[1]]
## + 3/4 vertices, from a30b8af:
## [1] 4 1 2
##
## [[2]]
## + 3/4 vertices, from a30b8af:
## [1] 4 3 2
```

In some cases, the term “path” is used for all of walks, trails and paths.

場合によっては、パスと言う用語でウォーク・トレイル・パスの全てを総称することもある。

# Cycles サイクル

Cycles are walks whose starting and ending verices are the same.

サイクルは、起始頂点と終着頂点が同一であるウォークのこと。

Polygon graphs are cycles themselves. 多角形グラフがそれ全体がサイクルである。

# Shortest paths and graph distance

The length of path is the sum of length of edges of the path.

The shortest path is the path whose length is shortest.

パスの長さは、パスの構成辺の長さの和。最短パスとは、長さが最短であるようなパスのこと。

The graph distance of two vertices of a graph is the length of shortest path of the vertices.

グラフの２頂点間の「グラフ距離」は、２頂点を結ぶ最短パスの長さ。

The shortes path is called geodesic, as well.

最短パスは測地線とも呼ばれる。

The shortes path from V4 to V2 is given as;

V4からV2への最短パスは次のようにして得られる。

vpath and epath are the sequence of vertices and edges, respectively.

vpathとepathと言う出力は、頂点の並び・辺の並びをそれぞれ意味する。

```
shortest_paths(g1,4,2,output="both")
```

```
## $vpath
## $vpath[[1]]
## + 3/4 vertices, from a30b8af:
## [1] 4 1 2
##
##
## $epath
## $epath[[1]]
## + 2/4 edges from a30b8af:
## [1] 1--4 1--2
##
##
## $predecessors
## NULL
##
## $inbound_edges
## NULL
```

Enumerate all shortest paths from V4 to all vertices of g1.

V4から、全ての頂点への最短パスを全て挙げることもできる。

```
all_shortest_paths(g1,4)
```

```
## $res
## $res[[1]]
## + 2/4 vertices, from a30b8af:
## [1] 4 1
##
## $res[[2]]
## + 3/4 vertices, from a30b8af:
## [1] 4 3 2
##
## $res[[3]]
## + 3/4 vertices, from a30b8af:
## [1] 4 1 2
##
## $res[[4]]
## + 2/4 vertices, from a30b8af:
## [1] 4 3
##
## $res[[5]]
## + 1/4 vertex, from a30b8af:
## [1] 4
##
##
## $nrgeo
## [1] 1 2 1 1
```

The shortest paths are related with graph distances. Graph distances can be calculated without listing paths.

最短パスとグラフ距離は密接な関係にある。 パスを列挙しないでグラフ距離のみを計算することもできる。

```
distances(g1,4,2) # distance from V4 to V2
```

```
##      [,1]
## [1,]    2
```

```
distances(g1) # distances of all vertex-pairs
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    2    1
## [2,]    1    0    1    2
## [3,]    2    1    0    1
## [4,]    1    2    1    0
```

# Subgraphs 部分グラフ

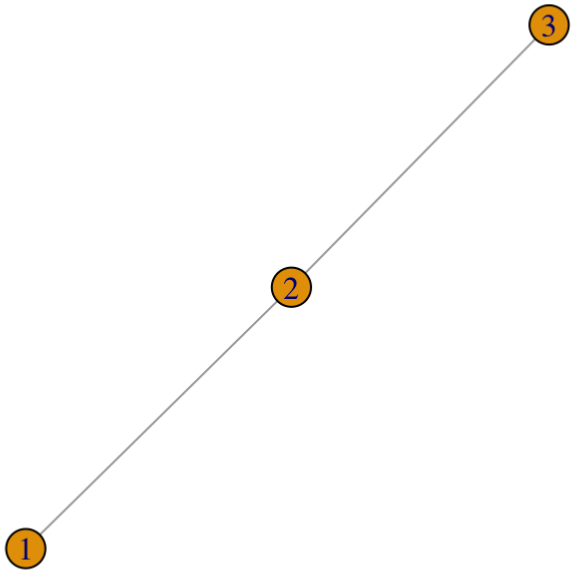
Subgraphs of a graph G is graphs whose vertices and edges are subsets of veritices and edges of G.

グラフGの部分グラフとは、Gの頂点集合と辺集合の部分集合からなるグラフのこと

igraph package has a function induced_subgraph(), that takes a subset of verties and returns a subgraph with the vertices. igraphでは、頂点部分集合を指定して、サブグラフを取り出すことができる。

Make a subgraph with V1,V2 and V3. V1,V2,V3をもつサブグラフを作る。

```
subg <- induced_subgraph(g2, c(1,2,3))
plot(subg)
```




# Regular graph 正則グラフ

A regular graph is a graph where all vertices have the same degree.

全ての頂点次数が等しいグラフを正則グラフと言う。

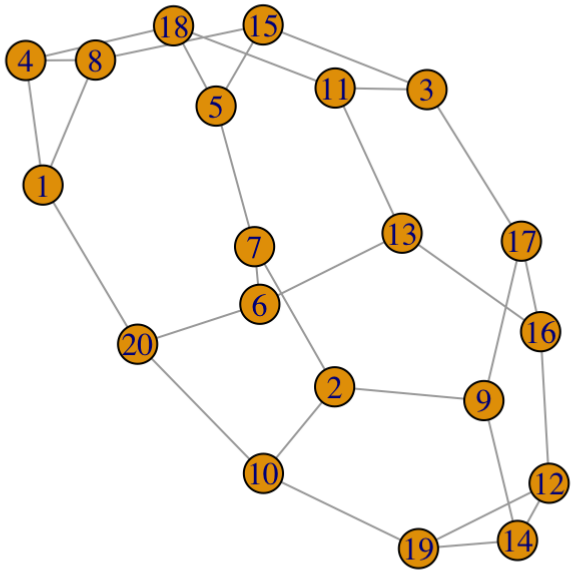
Let's make a regular graph randomly.

正則グラフをランダムに作ってみる。

$|V| = 20$ and 3-regular graph is randomly generated as below.

$|V| = 20$である3-正則グラフは以下のようにしてランダム発生できる。

```
g3 <- sample_k_regular(20,3)
plot(g3)
```




```
degree(g3)
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

# Planar graph 平面グラフ

A planar graph is a graph that can be embedded in a plane without crossings of edges.

平面グラフは、辺を交叉させることなく平面に描くことができるグラフ。

Planar graphs can be pasted on a sphere.

平面グラフは球面に貼り付けることができる。

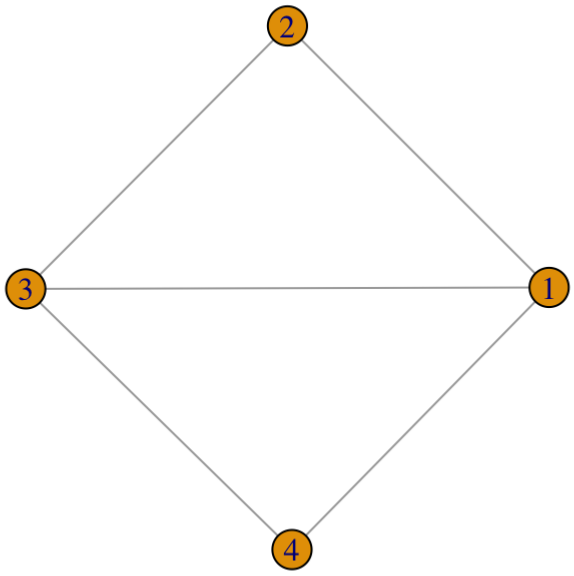
Polyhedron graphs are planar.

多面体グラフは平面グラフである。

Let's make a square with one diagonal graph.

正方形に１本の対角線を加えたグラフを作る。

```
el <- rbind(c(1,2),c(2,3),c(3,4),c(4,1),c(1,3))
g.planar <- graph.edgelist(el,directed=FALSE)
plot(g.planar)
```




Functions of planality are not available in igraph package but in RBGL package. Therefore install RBGL package from Bioconductor and convert the igraph object to the graph object of RBGL package.

平面グラフ関連の関数がigraphパッケージにはないので、BioconductorからRBGLパッケージをインストールし、igraphオブジェクトをRBGLオブジェクトに変換して、平面グラフの取り扱いをしてみる。

```
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")

#BiocManager::install("RBGL")
```

```
library(RBGL)
```

```
## Loading required package: graph
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:igraph':
##
##     normalize, path, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
```

```
##
## Attaching package: 'graph'
```

```
## The following objects are masked from 'package:igraph':
##
##     degree, edges, intersection
```

```
##
## Attaching package: 'RBGL'
```

```
## The following objects are masked from 'package:igraph':
##
##     bfs, dfs, transitivity
```

```
# convert igraph object to RBGL object
#g.RBGL <- igraph.to.graphNEL(as.undirected(g.planar))
g.RBGL <- as_graphnel(g.planar)
# Test whether or not planar
boyerMyrvoldPlanarityTest(g.RBGL)
```

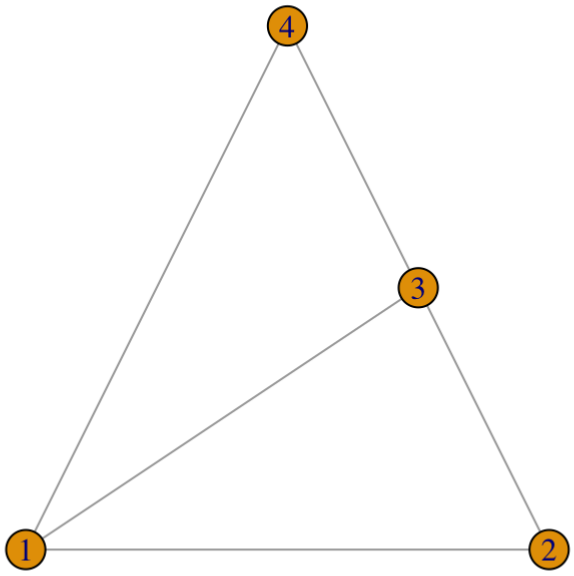
```
## [1] TRUE
```

Planar graphs can be embed in a plane. RBGL's chrobakPayneStraightLineDrawing() function calculates vertex 2D coordinates so that straight edges can be drawn without crossings if planar. Using this function, vertex coordinates should be determined and let the plot function of igraph package draw a graph.

平面グラフは平面埋め込み可能。 RBGLパッケージのchrobakPayneStraightLineDrawing()関数は、辺を直線で引いて交叉しないような頂点座標を算出する関数。

それを使って、頂点座標を決め、igraphオブジェクトの頂点座標を指定してグラフ描図する。

```
x.nocross <- chrobakPayneStraightLineDrawing(g.RBGL)
plot(g.planar, layout = t(x.nocross))
```




# Dual of a planar graph 平面グラフの双対

When a graph is planar, it has its dual graph.

グラフが平面的であるとき、双対と呼ばれるグラフが定義される。

Take a vertex for each face of the graph and make an edge for pairs of vertices when their corresponding faces in the original graph share a edge.

グラフの面を頂点に対応づけ、オリジナルのグラフの２つの面が辺を共有している場合に、その対応する頂点のペアを結ぶ辺をとる。

The planar graph that is a square with one diagonal, has 3 faces, 2 faces inside of the square and the 3rd face is the outside of the square.

正方形に１本の対角線を引いたグラフは、正方形の内側に２面、外側に１面の計３面ある。

Therefore, its dual has three vertices.

したがって、双対グラフの頂点数は３。

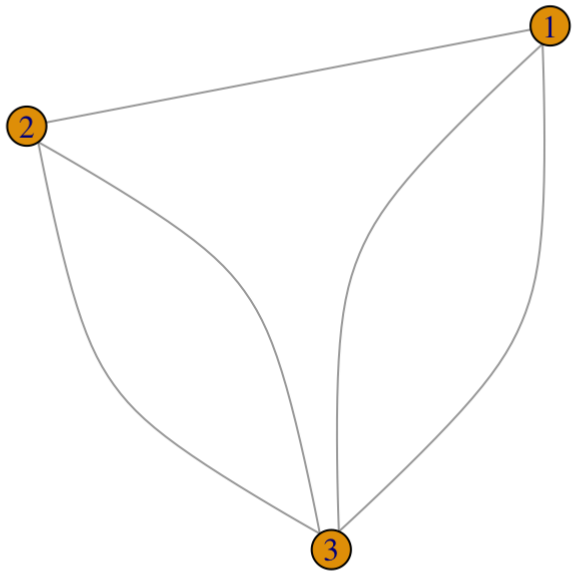
Of note, there are two edges between the faces inside of the square and the face outside.

正方形の内部の２面と、外部の面との間には２本のエッジが引かれることに注意する。

The dual of dual is the original.

グラフの双対のそのまた双対は、元のグラフになる。

```
el.dual <- rbind(c(1,2),c(1,3),c(1,3),c(2,3),c(2,3))
g.dual <- graph.edgelist(el.dual,directed=FALSE)
plot(g.dual)
```




# Assingment 課題

- Make a pyramid graph that has one square base and four oblique triangles. ピラミッドグラフを作れ(正方形の底面と、４つの三角形斜面からなる)
- Make the dual of the pyramid graph. ピラミッドグラフの双対グラフを作れ
- Make a 3-regular graph with 30 verties. 頂点数30の3-正則グラフを作れ。Draw a histogram of its pair-wise graph distances. そのグラフの頂点ペアのグラフ距離のヒストグラムを描け