

Graph Theorist Documentation

Comments, Questions, Suggestions?	3
Introduction	3
Typical Behavior	4
The Fundamentals	4
Locked - Create Vertices and Edges	4
Unlocked - Move Vertices Around	5
Selected Vertices	5
Selecting and Deselecting Vertices	6
Dragging Vertices	7
Undo All Recent Drag Actions	8
Modifying Graphs	8
Deleting Vertices	9
The Drawing Menu	9
Arrange as a Cycle	11
Show/Hide Grid	13
Cheating with a Tracing Image	14
Graph Operations	14
File Handling	18
Databases	18
Saving the Current Graph	19
Loading and Deleting Graphs	20
Variants	20
Actions on Variants	21
Loading Variants	21
Replacing Variants	22
Delete Variants.....	22
Rename Variants.....	22
Reorder Variants.....	22
Add Variants of Saved Graphs.....	22
Caveats and Nuances.....	22
Searching	23
Basics - Diameter, Radius, Girth	24
Maximum Independent Sets	25
Maximum Cliques	25
Minimum Dominating Sets	26
Longest Paths	27
Longest Induced Paths	28
Greedy Coloring	28
Connectivity	29
Straight Line Crossings	30

The Transform Popover	31
How does this work?.....	31
Scaling.....	31
Rotation	32
Flipping	33
Extra Add-ons	33
Circulants	34
What is an n -Circulant?.....	35
The Vertex/Label Inspector	35
Vertex Properties	36
Toggle Shadow	36
Vertex Labels.....	37
Positioning Labels	38
Custom Labels	39
The Edge Inspector	40
The Basics.....	40
Curved Edges.....	41
The Transparent Property of the Edge Inspector.....	41
Known Issues	42
Adornments	42
Geometric Adornments	43
Adjusting Adornment Properties.....	43
Text Adornments	44
Editing Adornments.....	44
Default Settings.....	45
General Default Settings.....	45
TikZ Output and LaTeX	46
Copy TikZ Code	47
Creating a New Graph	48
Saving/Loading Graphs.....	48
Saving/Loading Datasets.....	48
Support and Email	49
Graph Images.....	49
Sharing Graph Images	49
Copy Graph Image.....	49
More...	49

Comments, Questions, Suggestions?

The *Support* button at the top of the screen will open with the address of *Graph Theorist's* support. You can then send any comments, questions or suggestions you might have.

We want to hear from you! And we will respond promptly to any such emails!

Found a bug – let us know, and we will fix it as quickly as possible.

Don't like how something works? We'll see what we can do about it.

Is there something you particularly do like? We would love to hear about it!

And if you'd like, you can change the email address to that of a friend or colleague as well.

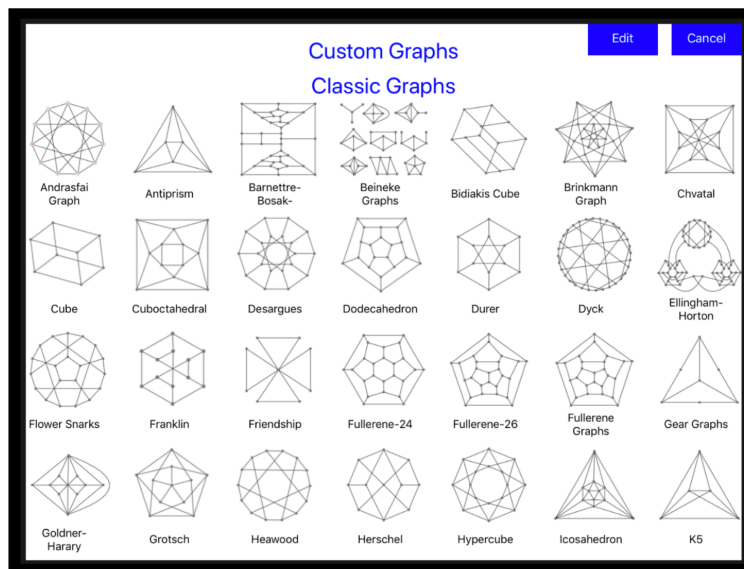
Introduction

Graph Theorist is an app for the Apple iPad and will eventually appear as a Macintosh App as well. Its purpose is to allow the user to:

- Conveniently draw, store and manipulate graphs.
- Print or save graph as png images.
- Output TikZ code for drawing the graph in LaTeX.
- Perform searches for many graph-theoretic properties.
- Save graphs in a local database

When you power up *Graph Theorist* for the very first time, there is a simple graph on the screen for demonstration purposes. This graph is known as the *Dodecahedron*. It is one of many special graphs stored in the *Graph Theorist* built-in database that is part of the app. You may want to experiment with this graph (and some of the others in the database as well) to get a feeling for how *Graph Theorist* works. To see the other built-in graphs tap on the *Graph* button in the upper-left of the iPad screen.

Note: Don't like to read user manuals? You can see *many* demos of Graph Theorist at <http://www.dpsumner.com>.



Graph Theorist's Built-in Database

Typical Behavior

Generally when using Graph Theorist to draw a graph, you will take advantage of symmetries as much as possible. You will typically, load special building blocks from the Add-ons popover and manipulate those using various drawing tools such as rotations, scaling and special constructions.

For video examples and demonstrations see: <http://www>.

The Fundamentals

The graph *canvas* (the area shown below and bounded by a gray square) is where all the action is. It is in here that you create and edit your projects.

The status of the *lock button* (enclosed in red in the image below) is crucial to the behavior of the graph canvas. When the canvas is locked vertices cannot be dragged around the canvas.

Locked – Create Vertices and Edges

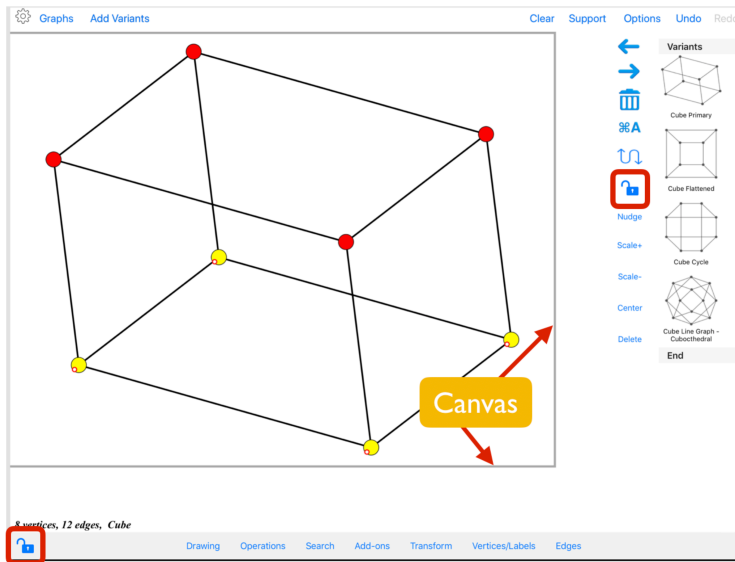
When the lock button is in locked position then a double-tap in the canvas area creates a new vertex at the tapped point. And dragging from one vertex to another creates an edge between those vertices.

Also, dragging over an existing edge deletes that edge from the graph.

In this mode, double tapping directly on a vertex *selects that vertex* (and you will see a tiny circle appear on the bottom left of the vertex). You'll see much more about selected vertices below.

Unlocked – Move Vertices Around

When the lock button is open, then tapping a vertex and dragging will move the vertex to a new location. A double-tap will still create a new vertex.

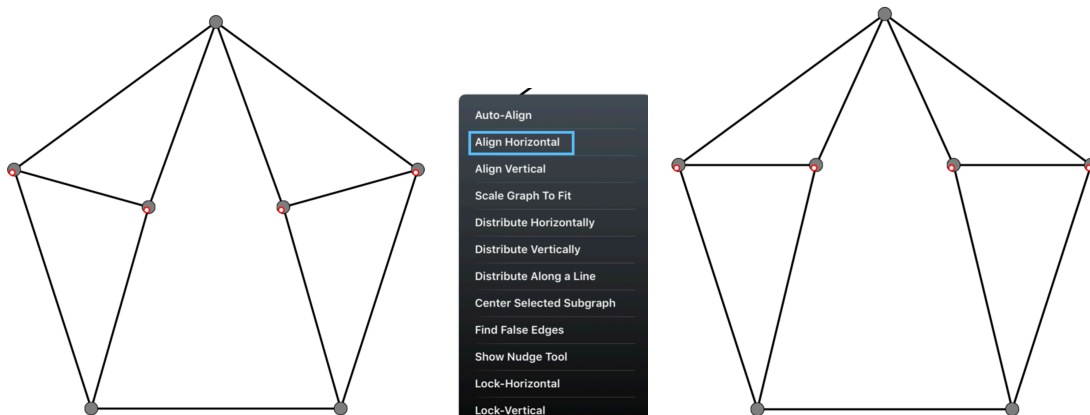


The lock button determines the behavior of the canvas.

Selected Vertices

Graph Theorist performs most of its operations on the currently selected vertices.

If you'd like to align some four vertices horizontally, then you would select the vertices and choose *Align Horizontal* from the *Drawing* menu.



First select the vertices to align.

Choose action.

The selected vertices align

Similarly, if you want to change the color for some vertices to red, then you would select them and open the *Vertices/Labels* window and select the color red. The actions of this window all take place on the currently selected vertices.

Note: *Graph Theorist* performs most of its magic on the selected vertices.

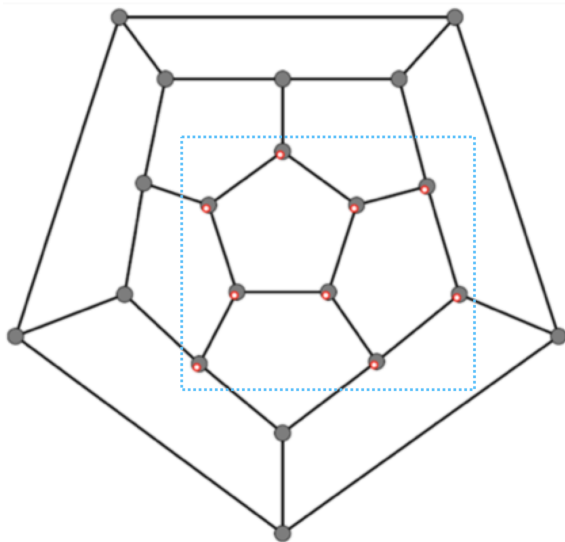
Selecting and Deselecting Vertices

There are two chief ways to select or deselect vertices.


You can double-tap on a vertex to select it. Or if it is already selected, double tapping will deselect it.

You may drag a *selection rectangle* as a shortcut to select a group of contiguous vertices. Just place your finger on any empty area of the graph canvas and drag out a rectangle to enclose the vertices you want to select. Drawing a selection rectangle deselects any vertices that are not inside the rectangle.

A selected vertex shows a small circle at the vertex's lower-left.



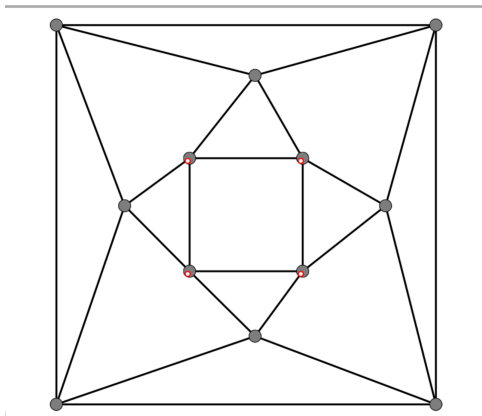
Selecting vertices via a selection rectangle

In addition to double tapping and drawing a selection rectangle, you may select all vertices at once by tapping the *Select All*  button.

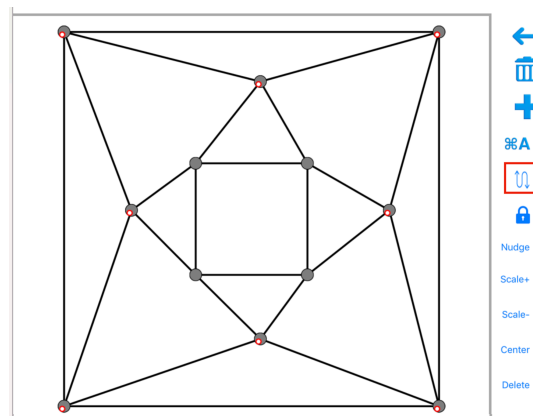
You can *deselect* all vertices by tapping the *Clear* button at the top of the screen.

Tip: As a quick alternative, you can *deselect* all vertices by dragging out a small selection rectangle that does not enclose any vertices.

Also, you can use the *inversion button* to select all the currently unselected vertices while simultaneously deselecting the currently selected vertices.



Initially there are four selected vertices



After the pressing the inversion button.

Note: Take care with double taps. A double tap in a blank portion of the display creates a new vertex, while double tapping on a vertex selects (or deselects) it

When in unlocked mode, double tapping near, but not on, a vertex may be interpreted by the app as an intention to drag it. Thus it is generally safer to perform double taps when in locked mode

Dragging Vertices

In unlocked mode, you can tap on any vertex and move it wherever you like.

Just as you can drag a single vertex around when in unlocked mode, you may move the entire set of selected vertices by dragging any one of them; all the selected vertices will move in unison and maintain their position relative to one another.

Note: If you want much greater control of the location of dragged vertices, you can use the *Nudge Tool*. (Choose *Drawing/Nudge Tool*) or tap the *Nudge* button along the side or bottom of the display.



The numbers at the bottom of the *Nudge Tool* allow you to choose the extent of the nudge in pixels. After selecting the distance you'd like the vertices to move, tapping one of the arrows moves all the selected vertices the chosen distance in the direction of the arrow.

Like almost all *Graph Theorist* actions, dragging vertices can be undone. Just select undo after a drag action and the vertices will return to their previous position.

Undo All Recent Drag Actions

So many of the actions you perform are simple dragging of vertices. It can be unpleasant to have to undo all of those one at a time, so *Graph Theorist* provides an option to undo all of the most recent drag actions at once. Just choose the *Undo Consecutive Drags* from the *Options* menu on the top right of the display, and the application will undo all the drag operations up to the last undoable action that was not a drag.

Note: There is no redo consecutive drag option – after choosing to undo the most recent drag actions the Redo button will redo them one at a time.

Note: Graph Theorist interprets movements of the vertices using the Nudge Tool as dragging actions, and so the menu selection *Undo Consecutive Drags* includes all nudge actions as well.

Modifying Graphs

You can modify the appearance and/or structure of the current graph in many ways. You can add a vertex by double-tapping any blank portion of the screen.

You can add an edge in locked mode by dragging from one vertex to another.

You can delete an edge in locked mode by dragging over it.

Deleting Vertices

You can delete the selected vertices from the graph by tapping the *Delete* button at the right side (or bottom) of the display.

In addition to this, you can perform many special actions on the set of selected vertices by using the popover menus. Use the buttons at the bottom of the display to activate these menus.

Note: Every *modification* that *Graph Theorist* performs on the current graph can be undone, and *Graph Theorist* supports an unlimited number of undos.

However, the creation of a new graph, or loading a graph from the local database are not undoable actions, and the original graph cannot be recovered unless it was saved previously. Similarly, as described later, loading a variant is an action that cannot be undone.

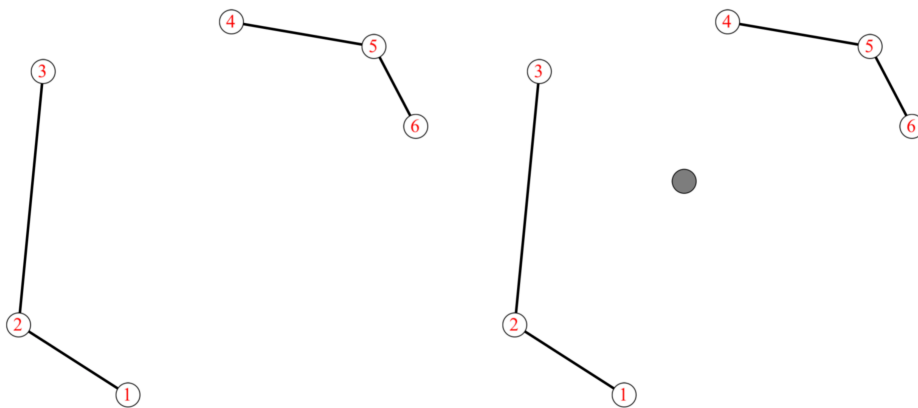
The Drawing Menu

The Drawing Menu supports fairly natural and standard actions on graphs.

Add Vertex at Polygon Centroid

First *select the vertices in the polygon in the order in which they appear on the boundary*. Then this option will create a new vertex at the centroid (i.e., *center of gravity* of that polygon).

The image below shows the result of adding a vertex at the centroid of the polygon on the left with the vertices chosen in the order shown.

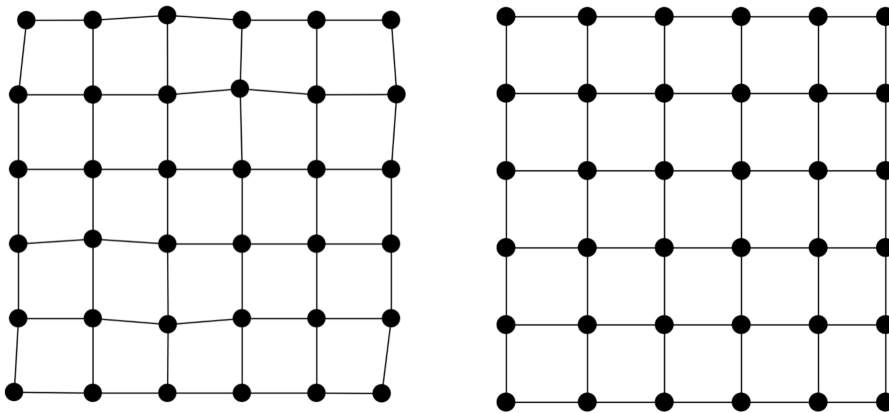


The result of adding a vertex at the centroid of a polygon.
Note there are two missing graph edges in the boundary of the selected polygon.

Note: It is important to choose the vertices in the order they appear in the polygon – otherwise unpredictable things might happen. It is also important that the polygon not be self-intersecting.

Auto-Align Choose this option, and Graph Theorist will search for edges that are almost horizontal or almost vertical and make them truly horizontal or vertical.

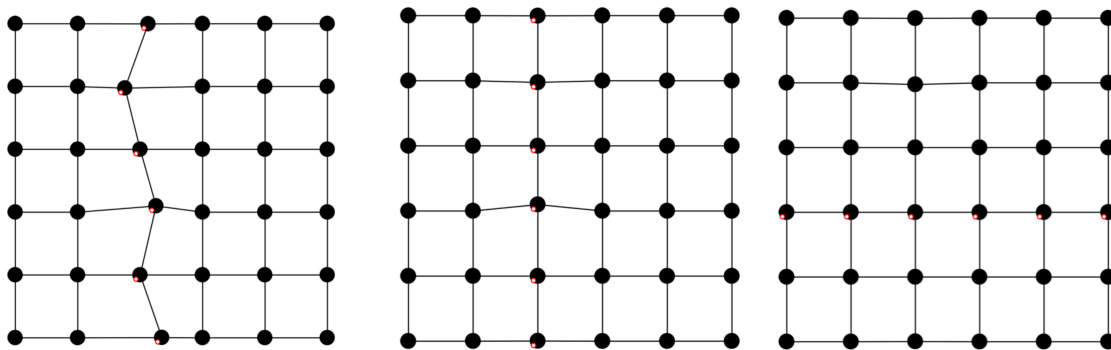
Thus you can be just a *little* sloppy when drawing your graph, and Graph Theorist will fix the alignments for you.



Before and after applying auto-align

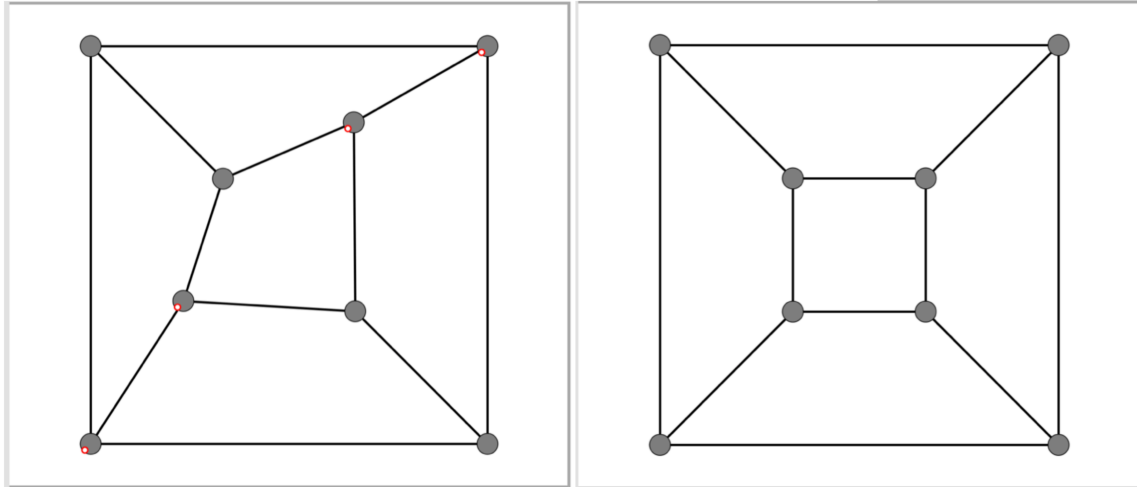
Align Horizontal – the selected vertices are all aligned in a horizontal line *determined by the initially selected vertex*; i.e., all selected vertices will be in an imaginary horizontal line that extends from the initially selected vertex.

Align Vertical - the selected vertices are all aligned in a vertical line *determined by the initially selected vertex*.



Before and after successively applying Align Vertical and then Align Horizontal

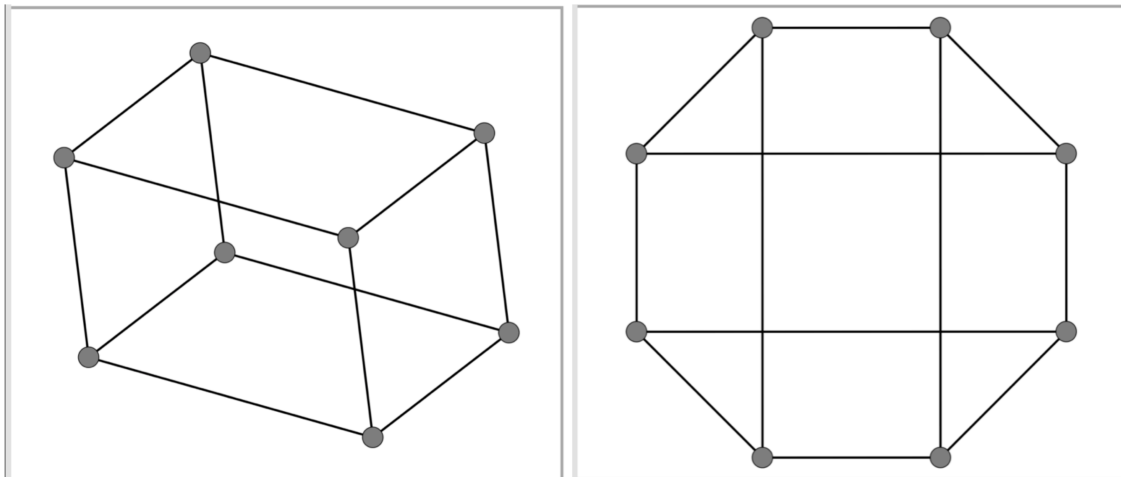
Distribute Along a Line – the selected vertices are aligned along the line determined by the first and last selected vertex. In addition, the vertices are distributed evenly along this line.



The Four Selected Vertices Selected From Bottom Left to Top Right Aligned in a Line

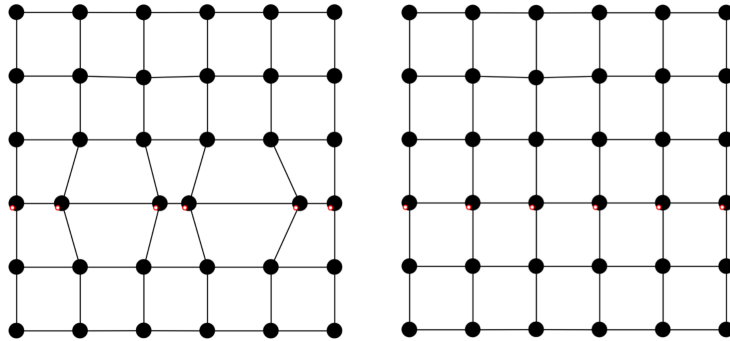
Arrange as a Cycle

Select the vertices that you'd like to arrange in a circle and *select them precisely in the order you want them to appear*. Then choosing this drawing action will arrange the selected vertices in a circle centered at the canvas center.



Before and After Using the Arrange in a Circle Action.

Distribute Horizontally – The selected vertices are arranged so that their horizontal centers are all the same distance apart. The leftmost and rightmost selected vertices remain in their initial position.

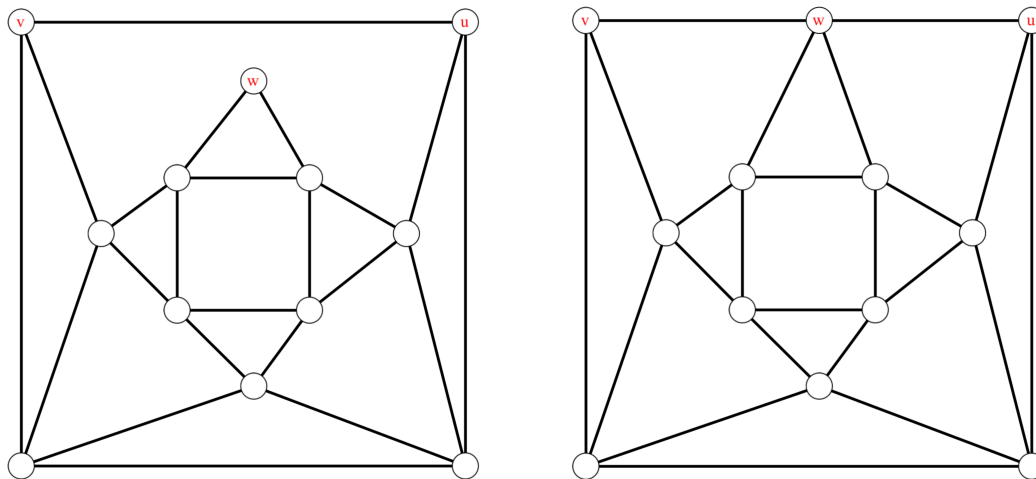


Before and after applying Distribute Horizontally

Distribute Vertically – The selected vertices are arranged so that their vertical centers are all the same distance apart. The topmost and bottommost vertices remain in their initial position.

Warning: All alignment actions can produce *false edges*. Graph Theorist attempts to alert you when this occurs, but you should be mindful of this possibility slipping through the cracks.

A *false edge* occurs when the edge between two vertices v, u passes through a vertex w making it appear that v and u are not adjacent but that v, w and u, w are adjacent.



Aligning v, w, u in a line produces false edges

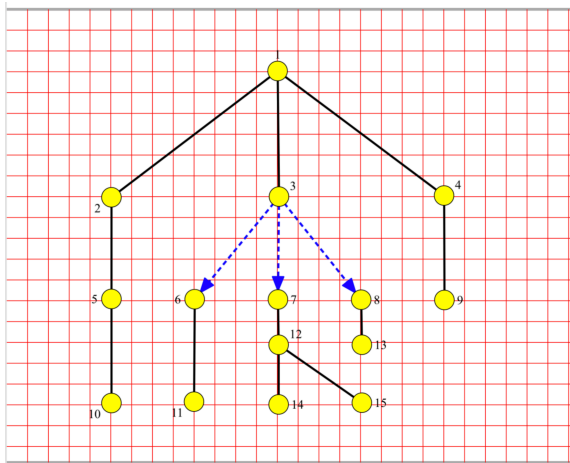
Find False Edges – If you suspect that a drag or other action/operation produced false edges, you can try this option to check for them. If Graph Theorist finds any

false edges then it alerts you to that. However, it is conceivable that false edges may not be caught, so you should always be alert to this possibility.

Center Selected Subgraph – Graph Theorist finds the smallest rectangle that encloses the set of selected vertices, called the *Selection Rectangle*, and moves that rectangle (along with the selected vertices) so that its center aligns with the graph canvas center.

Show/Hide Grid

This produces (or hides) a grid on the canvas to help you better place vertices.



The Grid on the Canvas

Scale Graph to Fit – Unlike most actions, this acts on the entire graph ignoring selected vertices. This scales the current graph so that it fits optimally inside the graph canvas.

Warning: If the graph contains curved edges, it is possible that portions of the edges will lie outside the scaled graph.

Lock Horizontal – choosing this item prevents dragged vertices from moving away from the horizontal. This places a checkmark next to the *Lock Horizontal* menu item and the option stays in effect until you choose the menu item a second time to deselect it.

Lock Vertical – choosing this item prevents dragged vertices from moving away from the vertical. This places a checkmark next to the *Lock Vertical* menu item and the option stays in effect until you choose the menu item a second time to deselect it.

Cheating with a Tracing Image

If you have a picture of the graph you want to create, Graph Theorist provides a nice option to produce it directly.

You can copy the graph image into the iPad Clipboard and paste it into Graph Theorist's canvas. No need to worry about the size; Graph Theorist will scale it to fit.

Then choose *Paste Tracing Image* from the Options Menu.

After that simply double-tap on the vertices in the image to create vertices in the canvas. And draw edges between vertices as they appear in the tracing image.

Also, if you have a picture of the graph you want in your photo's library then you can load it as a tracing image using Option Menu's *Load Tracing Image*.

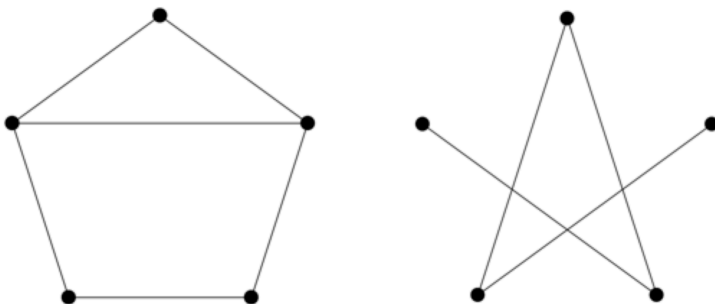
Once you are finished using the tracing image, choose *Clear Tracing Image* from the Options Menu.

Graph Operations

A *drawing action* modifies how the graph appears. In contrast, an *operation* on a graph is an action that modifies the structure of the graph in some way to produce a new graph. Graph Theorist allows you to perform several operations on the entire graph or on the subgraph of selected vertices.

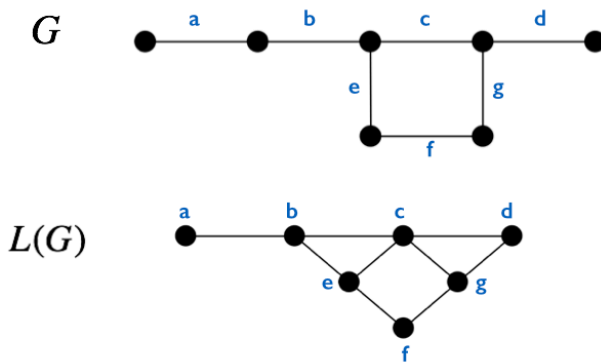
Tapping on the *Operations* button at the bottom of the display brings up a popover menu showing the available operations.

Complement Graph – For the current graph G , this forms the graph *complement* \bar{G} . Two vertices are adjacent in the graph complement if and only if they are not adjacent in G .



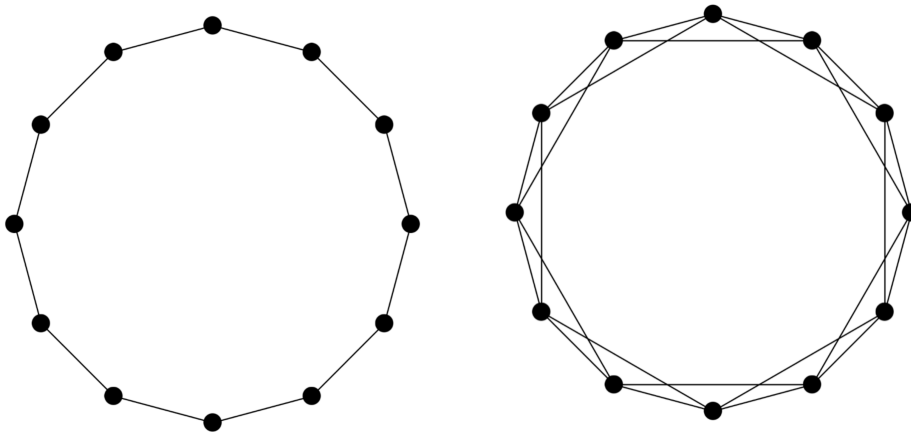
A graph and its complement

Line Graph – This operation replaces the graph by an isomorphic copy of its *Line Graph*. The line graph, $L(G)$, of a graph G has the edges of G as its vertex set and two of the vertices of $L(G)$ are adjacent if their corresponding edges are incident (see example below).



A graph G and its line graph $L(G)$

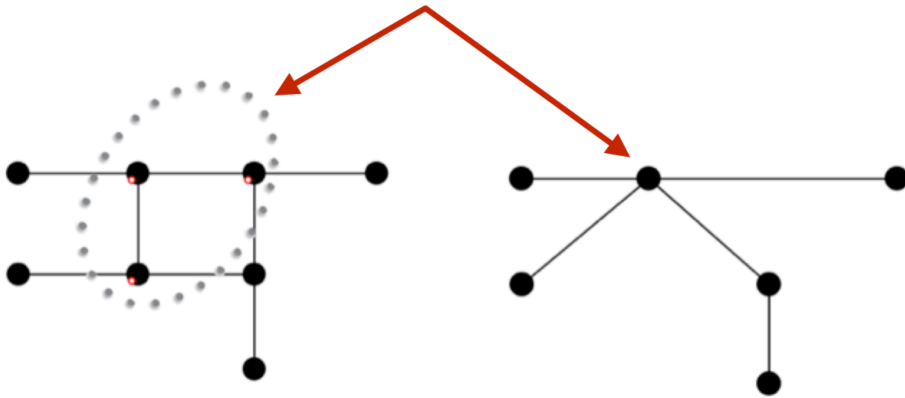
Square Graph – produces the *square* of the current graph. This amounts to adding an edge between any two vertices that were originally a distance 2 apart (the *distance* between two vertices is the minimum number of edges between the vertices).



A graph and its square (on the right)

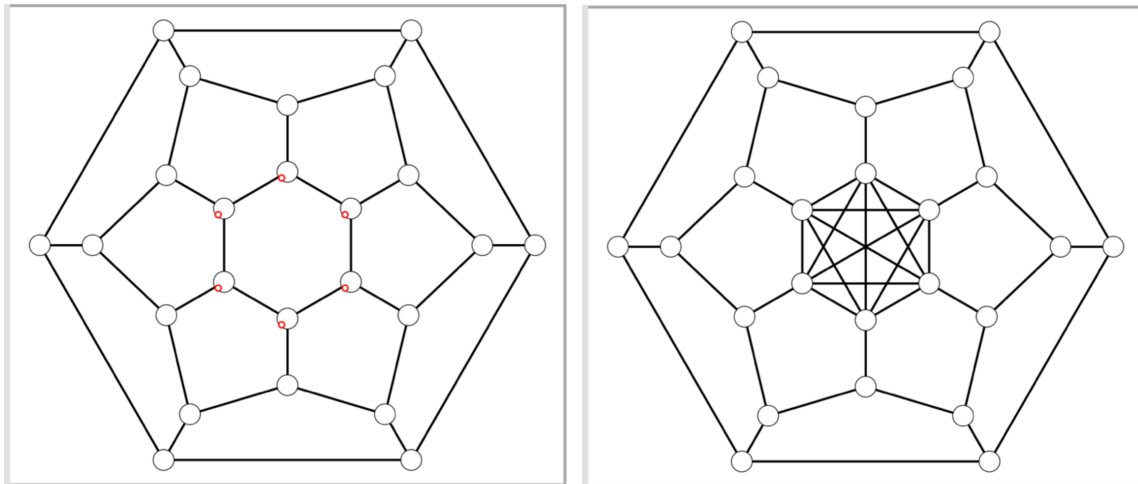
Cube Graph – produces the cube of the current graph. This amounts to adding an edge between any two vertices that were originally a distance 3 apart.

Identify Selected Vertices – Replaces the set of selected vertices by a single new vertex that is adjacent to any vertex that was adjacent to one of the originally selected vertices.



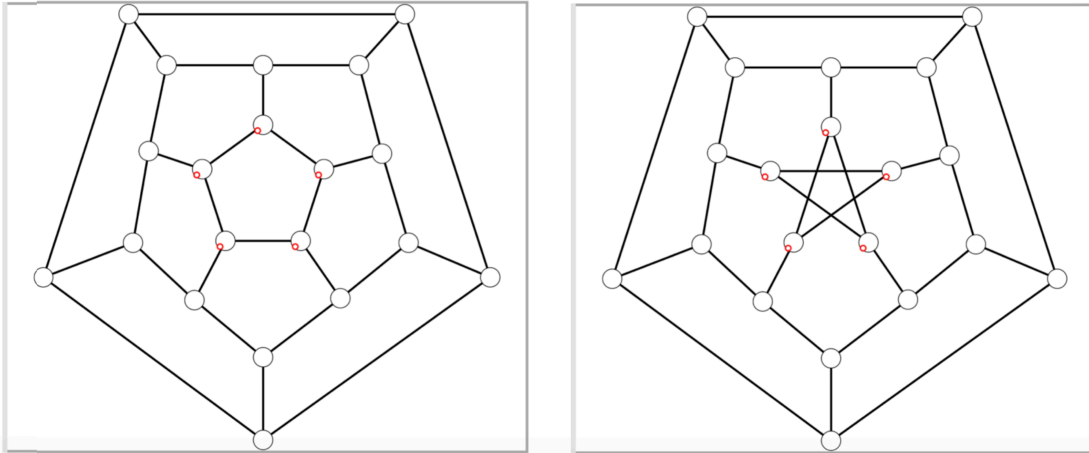
The graph on the right is obtained from the one on the left by *identifying* the three selected vertices into a single vertex.

Complete Selection – Adds all the edges that are missing between any two of the selected vertices.



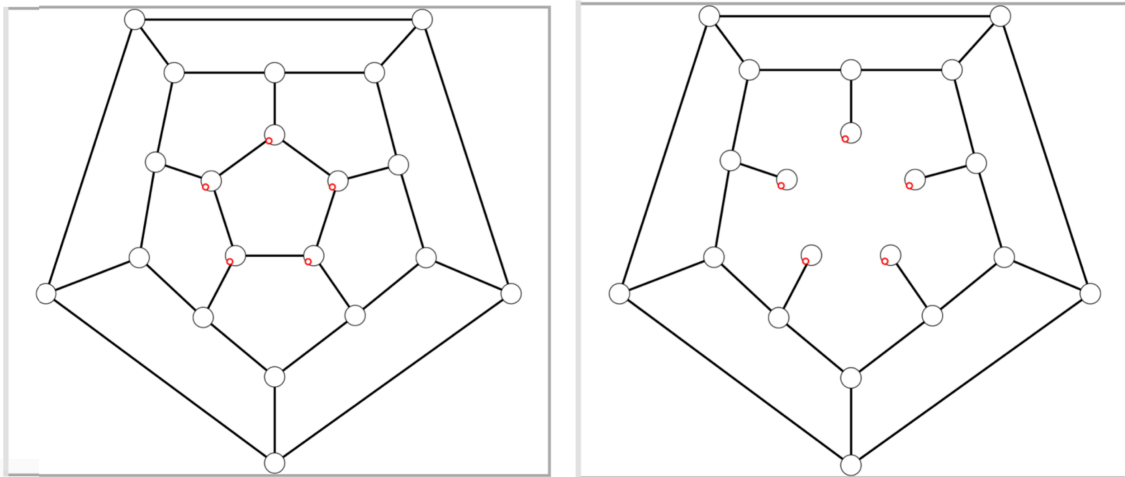
Before and after the *Complete Selected Edges* operation

Complement Selection – All the edges between any two selected vertices are removed and all the edges that were originally missing between any two selected vertices are added into the graph. In other words, this operation forms the complement of the subgraph determined by the selected vertices.



Before and after the *Complement Selection Edges* operation

Delete Selected Edges – Removes all the edges joining any two of the selected vertices.



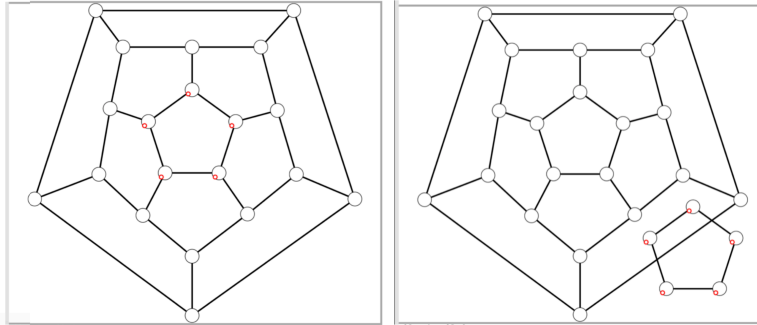
Before and after the *Delete Selected Edges* operation

Subdivide Selected Edges – For each edge $e = xy$ joining two selected vertices x and y , a new vertex u is added that is adjacent to each of x and y and then the edge $e = xy$ is removed. That's a complicated (but more precise) way of just saying, “place a new vertex in the middle of each of the selected edges” (but the original edge is deleted).



Subdividing an Edge

Duplicate Selected Subgraph– Adds to the current graph an exact duplicate of the subgraph determined by the set of selected vertices. Each of the original selected vertices is deselected and all the vertices in the duplicate are selected.



Before and after the *Duplicate Selected Subgraph* operation
(The resulting 5-cycle was dragged to its position after the operation)

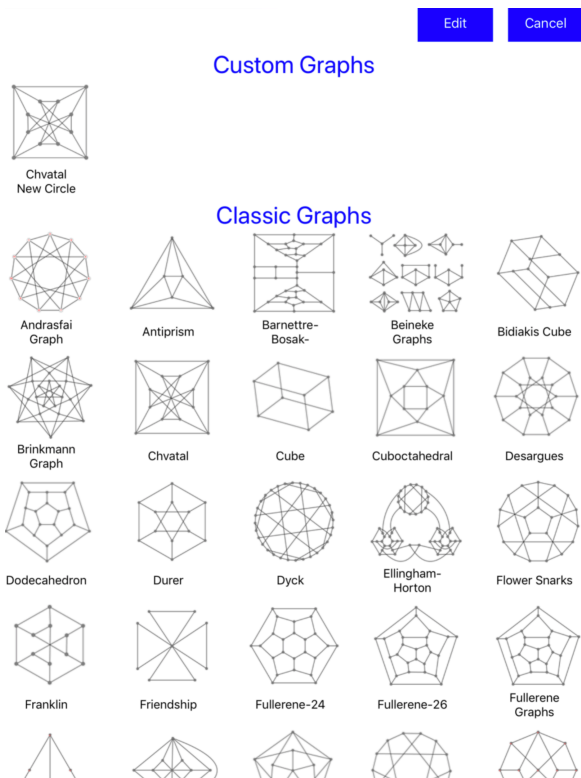
File Handling

Databases

Graph Theorist includes two distinct databases.

There is the built-in *Classic Graphs* database. And there is the *Custom Graphs* database where you can save and load the graphs you create.

You access both of these databases by tapping on the *Graphs* button at the top left of the screen. When you do this, you are taken to a display that looks somewhat like the image below.



You can both save and load the custom graphs you create yourself.

The Classic Graphs however are read only. Never the less, there is nothing stopping you from saving a modification of a classic graph into your custom database.

Note that in both databases, graphs are stored in alphabetical order.

Acknowledgement: The drawings of many of the graphs in the classical database are based on Wikipedia and Wolfram Mathworld.

Saving the Current Graph

You can save the current graph and all its variants to the local database by choosing the action *Save to Local Database* from the *Options* menu.

When you save a graph it also saves all the associated *variants* (see the next section for a discussion of variants).

If you attempt to load a new graph while the current graph has not been saved, then you will be alerted that you might want to save the current graph first.

Loading and Deleting Graphs

To load an existing graph, tap on the Graphs button on the top left and then simply tap the graph you want to load.

If you'd like to delete one of your custom graphs, then first tap the *Edit* button on the database screen. When you do that the *Edit* button's title becomes *Delete*. Now tap on all the graphs you want to delete each selected graph becomes colored green.

Finally tap *Delete* and the graph and all its variants will be deleted. Be careful, because this is not an undoable action.

You cannot delete the built-in classic graphs.

Variants

While working with a graph, you might want to

- Save various drawings of the graph,
- Save versions such as its line graph, or
- Save portions of the construction for an easy return to that start.

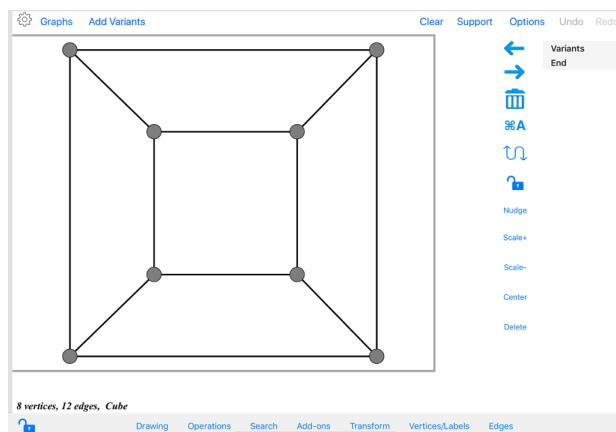
And more.

Variants are the way to handle each of these situations.

To save the current graph as a variant, simply press the *right-arrow* button and to load a variant to the canvas, first tap on the variant to select it (a checkmark appears to the right of the thumbnail) and then tap *the left-arrow* button.

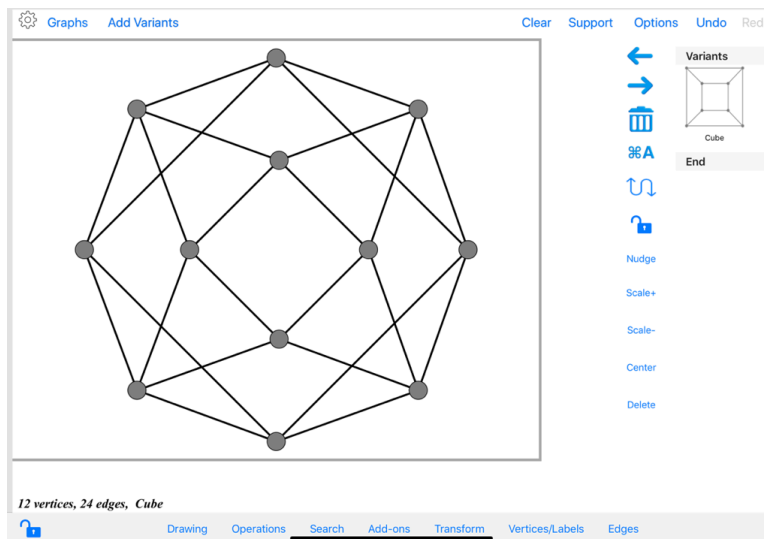
Example 1: Creating Different Versions

Suppose you are working with the graph below and decide to create its line graph. You'd like to keep this line graph, but also want to continue to work on the original graph.



The solution is to make the current line graph a variant. That's as easy as tapping on the *right arrow* icon in the right sidebar, or choosing *Create Variant* from the *Options Menu*.

You'll be asked to name the variant. If you leave this option blank, the variant name will be set to "Untitled".



Now once you are finished working on the line graph, you can save that as a variant as well and then load the original graph to resume work on it.

Example 2: Saving State

Suppose for example that you have constructed a graph up to a point, but are concerned about having to undo a bunch of future edits in case you want to return to this state. Then save the current state of the graph as a variant and then you may return to that state at any time.

Example 3: Additional Drawings

You'd like to save an alternative drawing of the current main graph before continuing on.

Actions on Variants

Loading Variants

You load a variant by first tapping on the variant to select it. You will see a checkmark appear next to the variant. Now tapping on the *Left-Pointing arrow* will load the variants into the canvas.

Warning: Loading a variant cannot be undone and it also flushes all the current undo/redo's. So before loading a variant be sure that you do not want to save the current state of the graph as a variant itself.

Replacing Variants

When tapping the right arrow button to save the current graph as a variant, you will get an option to replace the currently selected variant (if there is one).

Delete Variants

You can delete a variant by tapping on the trash icon and then tapping on the red minus icon. To exit this deletion mode, simply tap the trash icon a second time. This action cannot be undone.

Rename Variants

Swipe on the variants table from left to right to select a variant to rename.

Reorder Variants

You are not stuck with the default ordering of variants. To move a variant to a new position, just press down on the variant image, and when the image appears to pop up, drag it to the desired position.

Add Variants of Saved Graphs

If you tap the *Add Variants* button at the top left of the display, then you are taken to the graph database and when you select a graph, all of its variants are appended to the variants list of the current graph.

Caveats and Nuances

Note that the thumbnail image of a variant is made directly from the graph canvas. So if the graph only takes up a small portion of the canvas then the thumbnail image will look smaller than you might desire. You might want to use the Drawing Option to *Scale Graph to Fit* in order to make the thumbnail as prominent as possible.

When you save a graph to the local database, all its variants are saved with it.

Tip: If you tap the *Add Variants* button, then instead of opening the selected graph, Graph Theorist adds all the variants of the selected graph to the current list of variants.

Searching

Using the *Search* popover you can search the current graph for some of the most common graph theoretic entities.

Note: Searching is restricted to graphs having fewer than 64 vertices.

At the beginning of each search, a red spinner appears at the bottom right of the display, and continues to spin until the search either finishes or you choose to stop it. In addition, in most cases, a percentage-completed value appears to the right of the bounding box in landscape mode and below the box in portrait mode. This indicator is a rough indication of how far along the search is. In some cases, this percentage increases much faster as the search progresses.

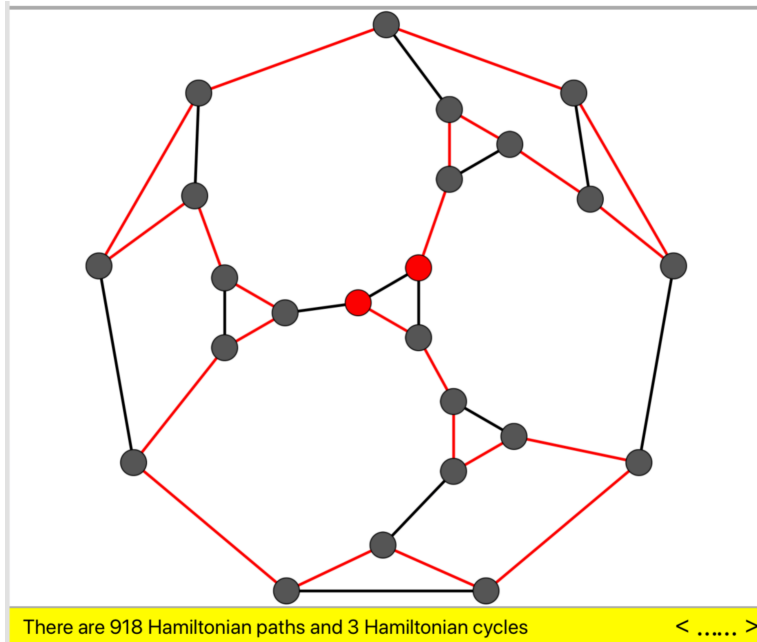
As soon as the search begins, the *Clear* button at the top of the display turns into a red *Abort* button; tapping this button at any time curtails the search. Graph Theorist disables the remaining top buttons for the duration of the search.



Pressing the red *Abort* Button stops the search.

Graph Theorist shows the progress of the search and also the final results in the *info-strip* at the bottom of the display (just above the button-bar, see figure below). You can use the white < ... > buttons on the far right of the info-strip to navigate back and fourth among the search results.

Tip: The effective way to use the < ... > buttons is to tap slightly inside the < or >.



The search shows one of the 918 Hamiltonian paths (a cycle in this case).
 The edges that comprise the path (or cycle) appear in red.
 Tap on the < > buttons to view all the resulting outcomes.

When the result of a search is a set of vertices and/or edges, *Graph Theorist* displays the elements of each found set in red, and all the found vertices are set as selected.

Note: The display of found vertices (or edges) in red is just temporary! As soon as you press the *Clear* button, the vertices and edges will again show whatever properties you've set on them.

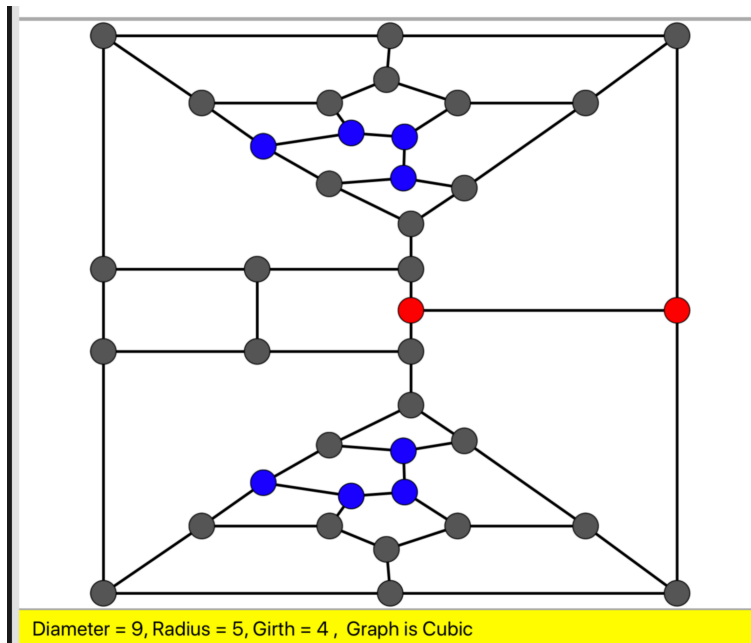
Press the *Clear* button to return the graph to its normal state and hide the info-strip.

Here are the things you can search for at the moment.

Basics – Diameter, Radius, Girth ...

This option determines the radius, diameter, girth, diametrical vertices, and the central vertices, the maximum and minimum degree and displays those values on the bottom info-strip. It also identifies trees, regular graphs, and bipartite graphs.

The vertices that comprise the *center* of the current graph (i.e., the vertices with eccentricity equal to the radius) are depicted in red, and diametrical vertices (those that are the end vertices of some diameter) appear in blue.



The vertices shown in red are the central vertices. Those shown in blue are diametrical.

Maximum Independent Sets

This search finds *all* the maximum independent sets of vertices in the current graph (an *independent set* is a set of vertices no two of which are adjacent).

The bottom info-strip, shows the number of maximum independent sets and the <...> buttons on the right of the info-strip allow you to navigate through all the sets.

As the search progresses, *Graph Theorist* consistently displays the most recently found independent set.

If you choose to stop the search before completion, the info-strip displays what it has found so far.

Maximum Cliques

A *clique* is a set of mutually adjacent vertices, i.e., the dual concept to that of an independent set.

This search finds *all* the maximum cliques in the current graph.

The bottom info-strip, shows the number of maximum cliques, and the <...> buttons on the right of the info-strip allow you to navigate through all of them.

As the search progresses, *Graph Theorist* consistently displays the most recently found clique.

If you choose to stop the search before completion, the info-strip displays what it has found so far.

Minimum Dominating Sets

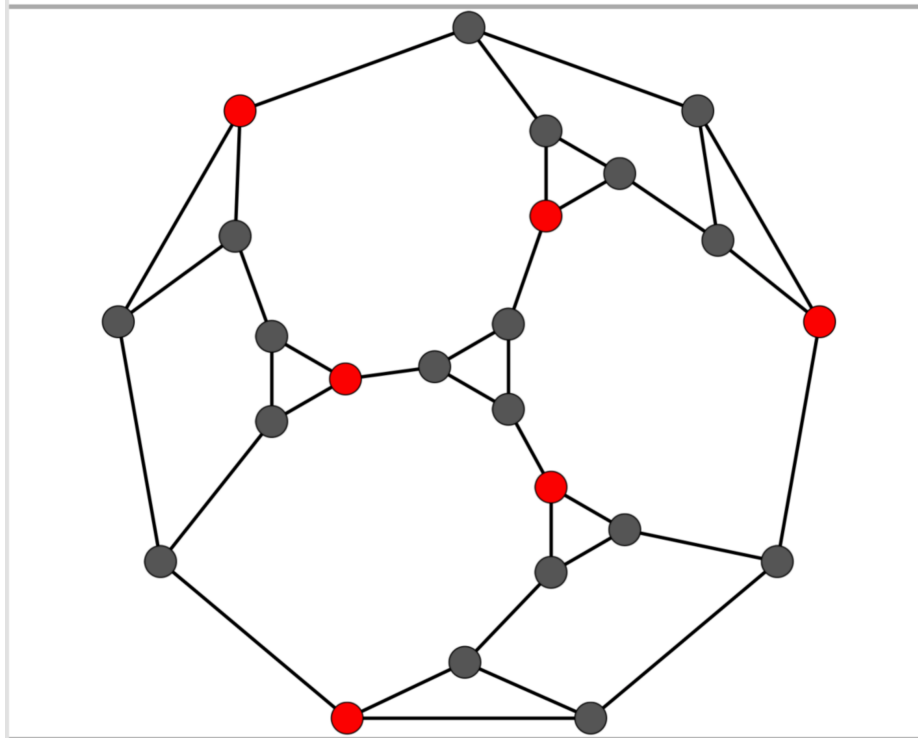
A set S of vertices of a graph is a *dominating set of vertices* if every vertex that is not in S is adjacent to a vertex inside S . The *domination number* of a graph is the cardinality of a smallest domination set.

This search option finds all possible dominating sets that have minimum cardinality.

The bottom info-strip, shows the number of minimum dominating sets, and the <...> buttons on the right of the info-strip allow you to navigate through all of them.

As the search progresses, Graph Theorist consistently displays the most recently found minimum dominating set. If you choose to stop the search before completion, the info-strip displays what it has found so far.

Example:



Domination number is 6 – there are 2 minimum dominating sets. < >

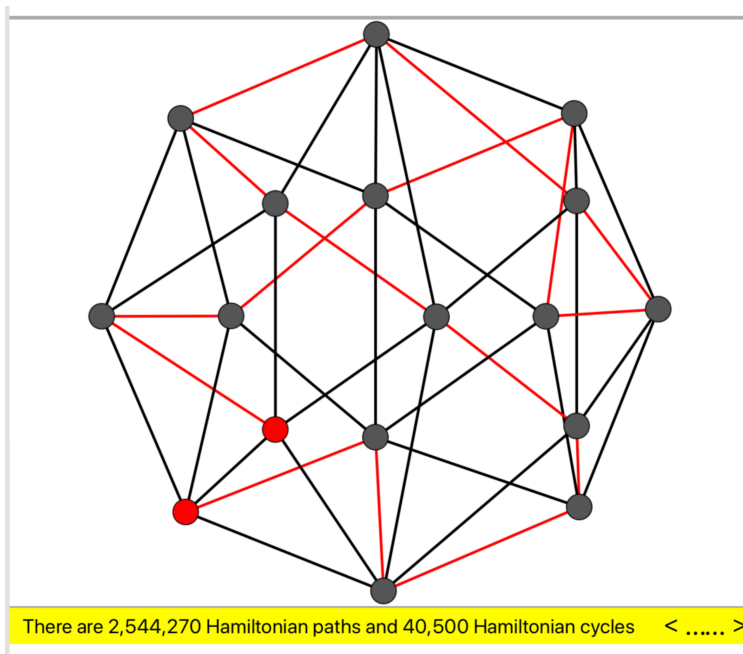
Note that each of the black vertices is adjacent to at least one of the red vertices. The red vertices represent one of the two minimum dominating sets of cardinality 6.

Longest Paths

You can search for the paths of greatest length. If there is a path that passes through every vertex, then obviously it is a longest path and such a path is denoted as a *Hamiltonian path*. Similarly, a cycle that passes through all the vertices is called a *Hamiltonian cycle*.

The search alerts you whenever it finds a Hamiltonian path or cycle. And if the search runs to completion all the Hamiltonian paths and cycles are available for display.

As you can see from the figure below, the initial and end vertices of the path appear as red vertices and the edges of the path are red.



The search shows one of the 2,544,270 Hamiltonian paths in this graph.

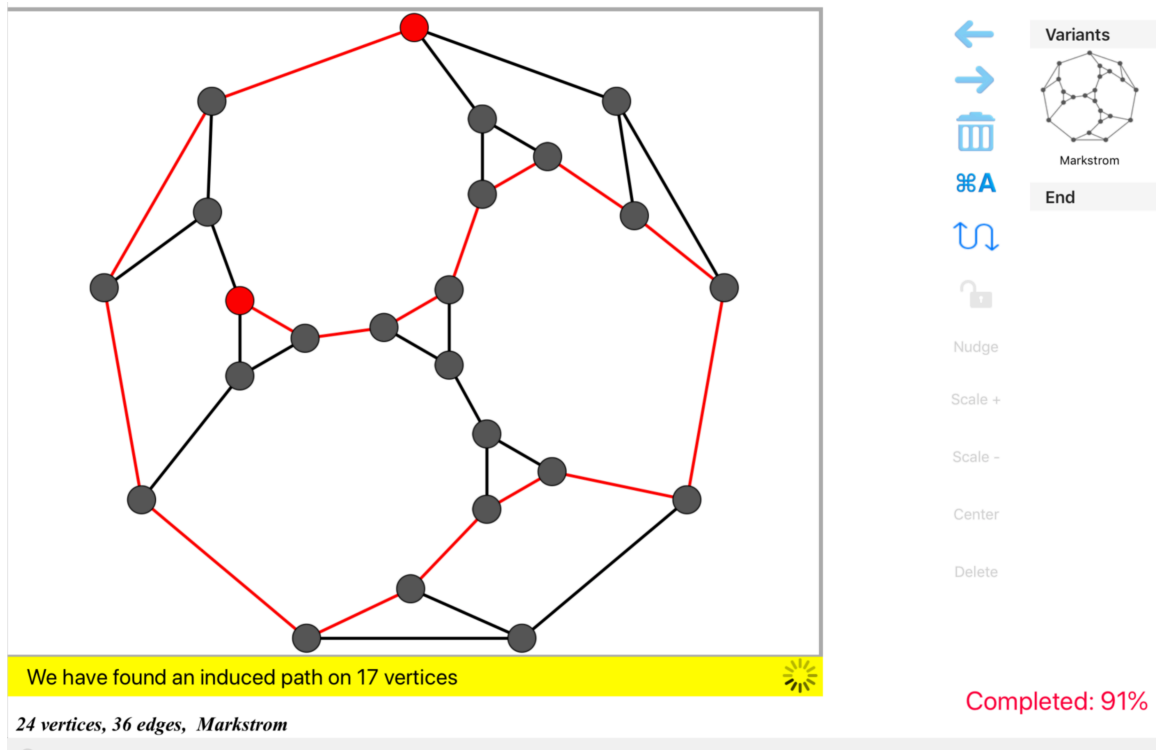
If there is a Hamiltonian cycle then the search displays that as soon as one is found and it remains on the display until the search terminates or you abort the search.

As with all searches, the <....> buttons to the right let you navigate through all the maximum paths (and Hamiltonian cycles should they exist).

Note: Paths are considered undirected here i.e., a path P from a to b , is considered identical to the reversed path from b to a . However, you may choose to change this in the General Settings so that paths are considered directed.

Longest Induced Paths

This does essentially what the search for longest paths does; only it looks for *induced* paths. A path is *induced* if none of its vertices are adjacent to vertices of the path other than its neighbors in the path. The image below shows such a search in progress.



The search shows the longest induced path it has found so far.

Once the search finishes, you'll discover that there are 216 induced paths on 17 vertices, and the picture below shows the fourth found path out of 108.

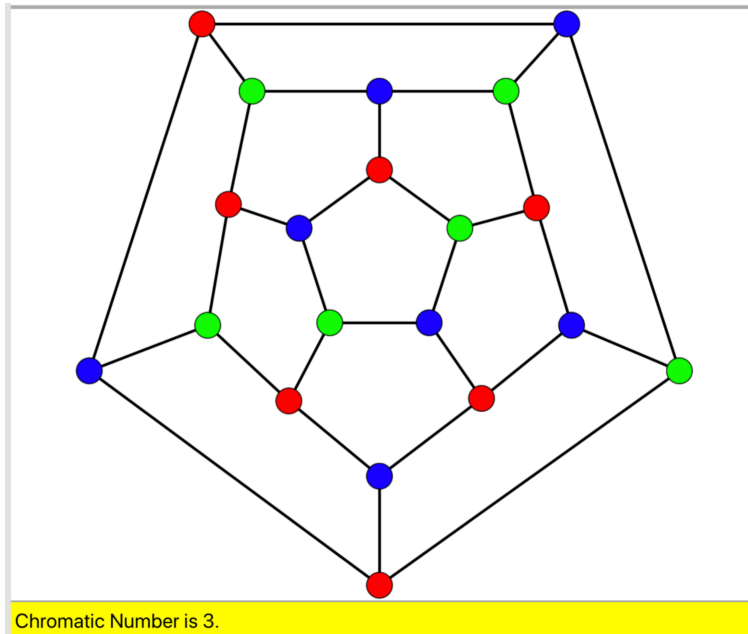
Greedy Coloring

The *chromatic number* of a graph is the minimum number of colors needed to color the vertices in such a way that no two adjacent vertices have the same color.

Graph Theorist does not attempt, at this time, to determine the chromatic number of the graph, but it does estimate this value.

Graph Theorist (very quickly) performs a million random greedy colorings and presents you with the best result it found. It also provides an actual coloring using that number of colorings.

In some special circumstances, Graph Theorist returns the actual chromatic number.



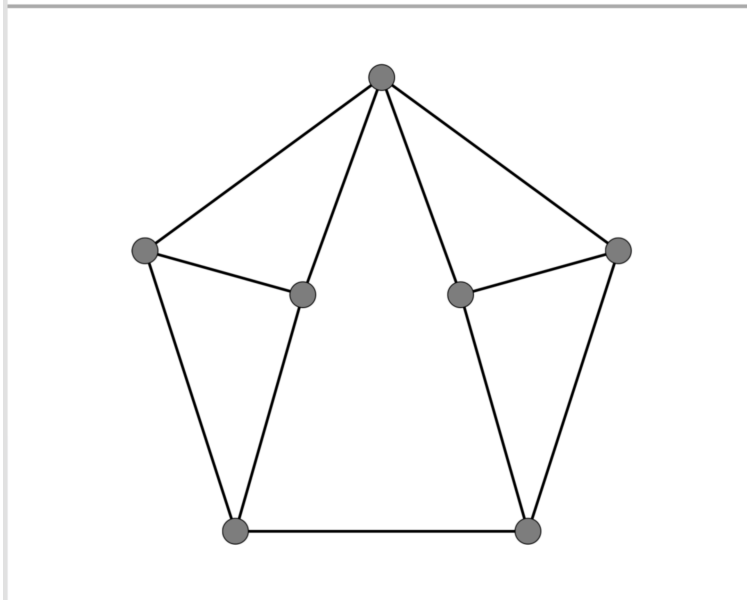
The Dodecahedron has chromatic number 3.

Connectivity

The *connectivity* of a connected graph is the smallest number of vertices that you must remove in order to disconnect the graph. This search determines the connectivity of the graph and displays all the possible separating sets of vertices (i.e., all minimum sets of vertices whose removal disconnects the graph).

The search only displays *non-trivial separating sets*. A separating set is *trivial* if it consists of the neighbors of a single vertex.

In the graph below, the three red vertices form the only 3-element, non-trivial, separating set of vertices.



Connectivity = 2; there are 2 cutsets of 2 elements and 2 non-trivial < >

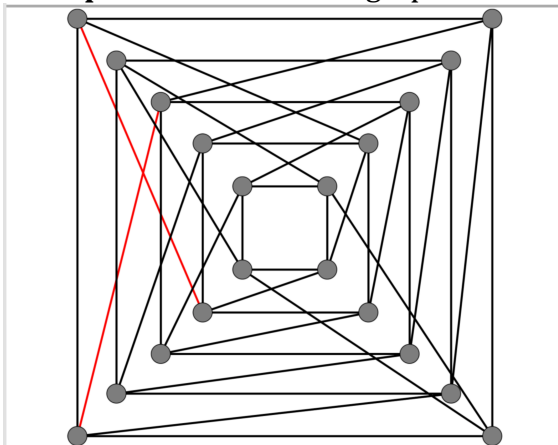
This graph has connectivity 3 and has exactly one non-trivial separating set of vertices.

Straight Line Crossings

Graph Theorist will determine all pairs of edges that cross one another and show the number of such crossings as well as allow you to see each such pair using the navigation buttons on the info-strip.

Note: Graph Theorist allows you to draw curved edges, however the search ignores any crossing in which at least one edge is curved.

Example: Note that in the graph below there are 40 crossings.



There are 40 straight-line crossings. < >