Graphonline API

Version 1.0

Description of API for algorithm development http://graphonline.ru/en/create_algorithm

Base class for algorithms - BaseAlgorithm

BaseAlgorithm is base class for algorithms. In order to write your own algorithm you have to inherit from it and override essential methods.

Class constructor has 2 parameters:

function BaseAlgorithm (graph, app)

- graph – object of a graph (http://graphonline.ru/script/Graph.js).
- app – object for interaction with the environment.

Usually in order to create a new algorithm one have to override getMessage method to output a message to user. It is recommended to conduct all the calculations in the Result method. The service will call this method at the right time.

Here are the methods of BaseAlgorithm class:

- BaseAlgorithm.prototype.getName = function(local)
  The method should return the name of your algorithm. local is the current language; “ru” or “en” are supported values.

- BaseAlgorithm.prototype.getId = function()
  The function should retun the unique id for the algorithm. The following form is supported: "your id"."algorithm id", i.e: OlegSh.MinPath. Starting the algorithm from browser on the http://graphonline.ru/create_algorithm page, the function should return "user.algorithm".

- BaseAlgorithm.prototype.getMessage = function(local)
  The function also returns the message for the user. This could be the result of the algorithm's performance or some piece of information needed for calculation, or information about error. local is the current language; “ru” or “en” are supported values.

- BaseAlgorithm.prototype.result = function(resultCallback)
  The calculation of the algorithm shoul be in this method. Moreover, you can do all the necessary things with the graph in this method. For example, you can change the text above the vertices, remove or add the vertices.
  resultCallback parameter shoud be used in case your algorithm performs asynchronously. This parameter is the object of the same form as return value of the Result function. If your algorithms works asynchronously, then result should return "null". If the current call made

the calculations, then the rusult should be the object with the field "version", equal to 1. For example,

```
  var result = {};
  result["version"] = 1;
```

If taking into account the terms the result cannot be calculated, then the function should return "null".
Moreover, if your algorithm calculates some path in the graph, which you would like to highlight with the animation for user, then result should contain array "paths". Every array's element is the path, it is set of id of vertices.Class of the vertex will be described below.
For example:

```
var nodesPath = this.GetNodesPath(results, 1, results.length - 1);
outputResult["paths"] = [];
outputResult["paths"].push(nodesPath);
```

How to fill result you may find in the algorithm of searching the shortest path
http://graphonline.ru/script/plugins/ShortestPath.js

- BaseAlgorithm.prototype.getObjectSelectedGroup = function(object)
  The method returns the groups of selection for the object, such as vertices and edges.
  Parameter "object" is the class of object (edge or vertex).
  Value should be "0" for the non-selected. Для невыделенных значение должно быть 0.
  Values "1" or more should be for highlighted. Для выделенных - 1 или более. If your algorithm groups the result into various groups, then for every group there should be a unique number.
  Here is an example of the method:

```
FindConnectedComponentNew.prototype.getObjectSelectedGroup = function(object)
{
   return (object.id in this.selectedObjects) ? this.selectedObjects[object.id] : 0;
}
```

- BaseAlgorithm.prototype.needRestoreUpText = function()
  If the algorithm changed upText of the vertex and the value should stay after the performance of the algorithm, then restore "false". In any other cases restore "true" in default.

- BaseAlgorithm.prototype.selectVertex = function(vertex)
  The method is called by the framework when a user has chosen a vertex.
  "vertex" is the object of a vertex (BaseVertex)

- BaseAlgorithm.prototype.selectEdge = function(edge)
  The method is called by the framework when a user chose an edge.
  "edge" is the Edge object (BaseEdge)

- BaseAlgorithm.prototype.deselectAll = function()
  A user resets the selection of the vertices and edges.

- BaseAlgorithm.prototype.instance = function()
  If your algorithm can conduct the calculation without any users' actions, then the method should return "true". In this case selectVertex/selectEdge methods will not be called.
  If the users' actions are necessary for calculations, then the method should return "false".
  For example, the search of the shortest path waits the selection of the vertices from a user:
  http://graphonline.ru/script/plugins/ShortestPath.js

- BaseAlgorithm.prototype.messageWasChanged = function()
  This method will be called when your message is displayed. In this case you may add event handlers to controls if there are some in your message for user. В этом методе вы можете добавить обработчики на какие-то элементы управления, если такие есть в вашем сообщении для пользователя. For example, the algorithm of path search gives the possibility to choose the report type: http://graphonline.ru/script/plugins/ShortestPath.js

Class of the graph - Graph (this.graph)

Your algorithm gets access to a graph through the class "Graph" (http://graphonline.ru/script/Graph.js). You may use "this.graph" class member. This calss gives you possibility to do with a graph wtahtever you wish. You can both read the graph's parameters and edit graph, if it is necessary for your algorithms.

A graph contains the list of vertices and edges. Let's describe each of these types:

BaseVertex – class of a vertex (http://graphonline.ru/script/BaseVertex.js)

- position – type Point determines position of a vertex on the working area.
- id – type int is a unique id of an object.
- mainText – type string, a text is in the center of the vertex.
- upText – type string, a text is above a vertex.

BaseEdge – class of an edge (http://graphonline.ru/script/BaseEdge.js)

- vertex1 – type BaseVertex, a vertex where an edge starts.
- vertex2 – type BaseVertex, a vertex where an edge finishes.
- isDirect – type bool determines whether an edge is oriented or not.
- weight – type float is weight of an edge.
- hasPair – type bool, does an edge have a pair, only oriented edges may have a pair.
- useWeight – type bool – do we need to use the weight of an edge or not
- id – type int is a unique id of an object.

Here are the major metods of graph, which can be used to get graph's parameters, vertices, and edges:

```
this.vertices = [];
```

The list of graph's vertices. You may yse this arrey to go round all the graph's vertices.

```
this.edges   = [];
```

The list of all the graph's edges.

- Graph.prototype.FindVertex = function(id)
  Search of a vertex based on id. It will return "null", if it does not find anything.

- Graph.prototype.FindEdge = function(id1, id2)
  Search of an edge based on 2 vertices defined by id, which it should connect.

- Graph.prototype.hasDirectEdge = function ()
  Does a graph have oriented edges or not?

Moreover, you may use the function:

```
function getVertexToVertexArray(graph, ignoryDirection)
```

Input is graph and an flag "true/false" and returns the arrey where every element is the list of connected vertices with this one. The function is described here: http://graphonline.ru/script/Algorithms.js

How to register the algorithm

You have to register the algorithm in the JavaScript file. In order to do that you have to create a function which creates a instance of your algorithm. For example:

```
function CreateAlgorithmSample(graph, app)
{
  return new AlgorithmSample(graph, app)
}
```

After that register that factory-function:

```
RegisterAlgorithm (CreateAlgorithmSample);
```

The names of your algorithm class and factory-function should be unique.

Interaction with application "this.app" class

this.app is used to interact with the environment. It contains the following methods:

- SetCurrentValue = function(paramName, value)

It saves the value according to specified name.

- GetCurrentValue = function(paramName, defaultValue)

It return the value that has been saved before

- redrawGraph = function()

It redraws a graph. Usually this method should not be called. It is necessary to use it if you create your own controls and want to redraw a graph by using them.

The example of this object usage you may find in the shortest path search method http://graphonline.ru/script/plugins/ShortestPath.js

Example of the algorithm

You may follow the link to find the example of the algorithm: http://graphonline.ru/script/plugins/VerticesDegree.js

This algorithm calculates degree of every vertex.

Let's examine the example:

1. Inherit from base class:
```
VerticesDegree.prototype = Object.create(BaseAlgorithm.prototype);
```

2. Determine the message for a user depending on the language:
```
VerticesDegree.prototype.getMessage = function(local)
{
   return (local == "ru" ? "Максимальная степень вершин графа равна " : "The maximum degree of a graph is ") + this.maxDegree;
}
```

3. Calculate exponent for every vertex. Calculate outdegree for an oriented graph. Moreover, save maximum exponent of a vertex in "this.maxDegree". The line “vertex.upText = currentDegree;” determines a text above the vertices.
```
VerticesDegree.prototype.result = function(resultCallback)
{
   this.degree = {};
   this.maxDegree = 0;

   var result = {};
   result["version"] = 1;
   this.degree = getVertexToVertexArray(this.graph, false);
   var graph = this.graph;

   for (var i = 0; i < graph.vertices.length; i++)
   {
      var vertex = graph.vertices[i];
      var currentDegree = 0;

      if (this.degree.hasOwnProperty(vertex.id))
      {
         currentDegree = this.degree[vertex.id].length;
         this.maxDegree = Math.max(this.maxDegree, currentDegree);
      }

      vertex.upText = currentDegree;
   }
```

```
  return result;
}
```

4. Return group for every vertex which equals its degree:

```
VerticesDegree.prototype.getObjectSelectedGroup = function(object)
{
  return (this.degree.hasOwnProperty(object.id)) ? this.degree[object.id].length: 0;
}
```

5. Determine function фабричную функцию and register it:

```
function CreateAlgorithmVerticesDegree(graph, app)
{
  return new VerticesDegree(graph, app)
}

RegisterAlgorithm (CreateAlgorithmVerticesDegree);
```

Support and feedback

Please send questions, comments, and suggestions to soft_support@list.ru, admin@unick-soft.ru