Graphonline API

Version 1.0

Описание API для разработки алгоритмов http://graphonline.ru/create_algorithm

Базовый класс для алгоритмов - BaseAlgorithm

BaseAlgorithm является базовым классом для алгоритмов. Для написания собственного алгоритма необходимо наследоваться от него и переопределять необходимые методы.

Конструктор класса принимает 2 параметра:

function BaseAlgorithm (graph, app)

- graph - объект графа (http://graphonline.ru/script/Graph.js).
- app - объект для взаимодействия со средой.

Обычно для создания нового алгоритма придётся переопределить метод getMessage, чтобы выводить какое-то сообщение пользователю. Все вычисления рекомендуется производить в методе result. Сервис будет вызывать этот метод в нужный момент.

Ниже описаны методы класса BaseAlgorithm:

- BaseAlgorithm.prototype.getName = function(local)
  Метод должен возвращать имя вашего алгоритма. local - текущий язык, поддерживаемые значения “ru” или “en”.

- BaseAlgorithm.prototype.getId = function()
  Функция должна возвращать уникальный идентификатор для алгоритма. Принят формат: "your id"."algorithm id", то есть: OlegSh.MinPath. При запуске алгоритма из браузера на странице http://graphonline.ru/create_algorithm функция должна возвращать "user.algorithm".

- BaseAlgorithm.prototype.getMessage = function(local)
  Функция возвращает текущее сообщение для пользователя. Это может быть результат работы алгоритма или какая-та информация, необходимая для расчёта, или информация об ошибке. local - текущий язык, поддерживаемые значения “ru” или “en”.

- BaseAlgorithm.prototype.result = function(resultCallback)
  В этом методе должен происходить расчёт алгоритма. Также вы можете делать с графом всё необходимое в этом методе. Например: поменять текст над вершинами, удалить или добавить вершины.
  Параметр resultCallback - необходимо использовать только если ваш алгоритм работает асинхронно. Этот параметр является объект такого же формата, как возвращаемое

значение функции result. Если ваш алгоритм работает в асинхронном режиме, то result должен вернуть null.
Если текущий вызов произвел вычисления, то результатом должен быть объект с полем “version”, равным 1. Например:

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

Если при данных условиях результат не может быть рассчитан, то функция должна вернуть null.
Кроме того, если ваш алгоритм вычисляет какой-то путь в графе, который вы бы хотели выделить для пользователя анимацией, то result должен содержать массив "paths". Каждый элемент массива - это путь, заданный список id вершин пути. Класс вершины будет описан ниже.
Например:

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

Пример заполнения поля вы можете найти в алгоритме поиска кратчайшего пути http://graphonline.ru/script/plugins/ShortestPath.js

- BaseAlgorithm.prototype.getObjectSelectedGroup = function(object)
  Метод возвращает группы выделения для объекта: вершины или дуги.
  Параметр object - класс объекта (дуга или вершина).
  Для невыделенных значение должно быть 0. Для выделенных - 1 или более. Если ваш алгоритм группирует результат по разным группам, то для каждой группы должно быть уникальное число. Пример метода:

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

- BaseAlgorithm.prototype.needRestoreUpText = function()
  Если алгоритм поменял upText вершины и значение должно остаться после работы алгоритма, тогда верните false. Во всех иных случаях и по умолчанию возвращайте true.

- BaseAlgorithm.prototype.selectVertex = function(vertex)
  Метод вызывается средой, когда пользователь выбрал вершину.
  vertex - объект вершины (BaseVertex)

- BaseAlgorithm.prototype.selectEdge = function(edge)
  Метод вызывается средой, когда пользователь выбрал дугу.
  edge - объект дуга (BaseEdge)

- BaseAlgorithm.prototype.deselectAll = function()
  Пользователь сбросил выделение вершин и дуг.

- BaseAlgorithm.prototype.instance = function()
  Если ваш алгоритм может произвести расчёт без действий со стороны пользователей, то метод должен возвращать true. В этом случае методы selectVertex/selectEdge не буду вызываться.
  Если для расчёта необходимы действия пользователей, то метод должен возвращать false.
  Например, поиск кратчайшего пути ждёт выбора вершин от пользователя:
  http://graphonline.ru/script/plugins/ShortestPath.js

- BaseAlgorithm.prototype.messageWasChanged = function()
  Данный метод будет вызван, когда ваше сообщение пользователю будет выведено на экран. В этом методе вы можете добавить обработчики на какие-то элементы управления, если такие есть в вашем сообщении для пользователя. Например, алгоритм поиска пути предоставляет возможность выбора типа отчёта:
  http://graphonline.ru/script/plugins/ShortestPath.js

Класс графа - Graph (this.graph)

Ваш алгоритм получает доступ к графу через класс Graph (http://graphonline.ru/script/Graph.js). Вы можете использовать член класса this.graph. Этот класс предоставляет вам возможность делать с графом всё, что угодно. Помимо чтения параметров графа, вы можете вносить в граф изменения (в случае, если это необходимо вашему алгоритму).

Граф содержит список вершин и дуг. Опишем каждый из этих типов:

BaseVertex - класс вершины (http://graphonline.ru/script/BaseVertex.js)

- position - тип Point - задаёт позицию вершины на рабочей области.
- id - тип int - уникальный идентификатор объекта.
- mainText - тип string - текст в центре вершины.
- upText - тип string - текст над вершиной.

BaseEdge - класс дуги (http://graphonline.ru/script/BaseEdge.js)

- vertex1 - тип BaseVertex - вершина, из которой идёт дуга.
- vertex2 - тип BaseVertex - вершина, в которую входит дуга.
- isDirect - тип bool - является ли дуга ориентированной или нет.
- weight - тип float - вес дуги.
- hasPair - тип bool - имеет ли дуга парную дугу, только ориентированные дуги могут иметь парную дугу.
- useWeight - тип bool - использовать вес дуги или нет.
- id - тип int - уникальный идентификатор объекта.

Ниже описаны основные методы графа, которые вы можете использовать для получения параметров графа, вершин и дуг:

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

Список вершин графа. Можете использовать этот массив для обхода всех вершин графа.

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

Список всех дуг графа.

- Graph.prototype.FindVertex = function(id)
  Поиск вершины по id. Вернёт null, если ничего не нашёл.

- Graph.prototype.FindEdge = function(id1, id2)
  Поиск дуги по двум вершинам заданными id, которые она должна соединять.

- Graph.prototype.hasDirectEdge = function ()
  Имеет ли граф ориентированные дуги или нет.

Кроме того, вы можете воспользоваться функцией:

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

Она принимает граф и флажок true/false и возвращает массив, где каждым элементом является список соединённых вершин с данной. Функция описана в http://graphonline.ru/script/Algorithms.js

Регистрация алгоритма

В js файле необходимо зарегистрировать алгоритм. Для этого создайте функцию, которая создаёт экземпляр вашего алгоритма. Например:

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

А после этого зарегистрируйте эту функцию-фабрику:

```
RegisterAlgorithm (CreateAlgorithmSample);
```

Имена вашего класса алгоритма и фабричной функции должны быть уникальными.

Класс взаимодействия с приложением this.app

Объект this.app используется для взаимодействия со средой. Он содержит следующие методы:

- SetCurrentValue = function(paramName, value)

Сохраняет значение по заданному имени.

- GetCurrentValue = function(paramName, defaultValue)

Возвращает ранее сохранённое значение

- redrawGraph = function()

Перерисовывает граф. В обычных случаях данный метод вызывать не надо. Он необходим, если вы создаёте собственные элементы управления и хотите перерисовать граф по нажатию на них.

Пример использования этого объекта вы можете найти в алгоритме поиска кратчайшего пути http://graphonline.ru/script/plugins/ShortestPath.js

Пример алгоритма

Пример алгоритма вы можете найти по ссылке:
http://graphonline.ru/script/plugins/VerticesDegree.js

Этот алгоритм рассчитывает степень каждой вершины.

Рассмотрим пример более подробно:

1. Наследуемся от базового класса:
```
VerticesDegree.prototype = Object.create(BaseAlgorithm.prototype);
```

2. Задаём сообщение для пользователя в зависимости от локали:
```
VerticesDegree.prototype.getMessage = function(local)
{
   return (local == "ru" ? "Максимальная степень вершин графа равна " : "The maximum degree of a graph is ") + this.maxDegree;
}
```

3. Рассчитываем степень для каждой вершины. Для ориентированного графа считаем полустепень исхода. Кроме того, сохраняем максимальную степень вершины в this.maxDegree. Строка “vertex.upText = currentDegree;” задаёт текст над вершинами.
```
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. Возвращаем группу для каждой вершины, которая равна её степени:

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

5. Задаём фабричную функцию и регистрируем её:

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

RegisterAlgorithm (CreateAlgorithmVerticesDegree);
```

Поддержка и обратная связь

Все вопросы и предложения присылайте на электронные адреса: soft_support@list.ru, admin@unick-soft.ru