|  |  |  |  |
| --- | --- | --- | --- |
| **УТВЕРЖДЕНО**  **RU.17701729.04.01-01 ТП 01–1-ЛУ** | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.04.01-01 ТП 01-1** | | **ПРИЛОЖЕНИЕ ДЛЯ ВИЗУАЛИЗАЦИИ**  **АЛГОРИТМОВ ИЛИ СТРУКТУР ДАННЫХ**  **Текст программы**  **RU.17701729.04.01-01 ТП 01–1**    **Листов 30** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |

**2022**

# **Аннотация**

В данном документе приведет текст «Приложение для визуализации алгоритмов или структур данных».

Программа разработана на языке JavaScript с поддержкой HTML и CSS.   
Среда разработки - WebStorm 2021.3.2.

Приложение позволяет строить графы и решать их с помощью алгоритмов поиска мостов и точек сочленения. Приложение также предоставляет пошаговое интерактивное решение графов с подсказками.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101–77 Виды программ и программных документов [1];

2) ГОСТ 19.102–77 Стадии разработки [2];

3) ГОСТ 19.103–77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104–78 Основные надписи [4];

5) ГОСТ 19.105–78 Общие требования к программным документам [5];

6)ГОСТ 19.106–78 Требования к программным документам, выполненным печатным способом [6];

7) ГОСТ 19.401–78 Текст программы. Требования к содержанию и оформлению [7].

Изменения к данному документу оформляются согласно ГОСТ 19.603–78 [8], ГОСТ 19.604–78 [9].

СОДЕРЖАНИЕ

[Аннотация 2](#_Toc100870349)

[Текст программы 4](#_Toc100870350)

[**1.1** **Код класса Graph.js** 4](#_Toc100870351)

[**1.2** **Код класса Help.js** 17](#_Toc100870352)

[**1.3** **Код класса SiteMaker.js** 18](#_Toc100870353)

[**1.4** **Код класса Pair.js** 19](#_Toc100870354)

[**1.5** **Код класса GraphMaker.js** 19](#_Toc100870355)

[**1.6** **Код style.css** 21](#_Toc100870356)

[**1.7** **Код GraphMakerSolver.html** 26](#_Toc100870357)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 30](#_Toc100870358)

# **Текст программы**

## **Код класса Graph.js**

**var maxCountOfNodes = 30**

**function checkEdges(){**

**console.log("checkEdges")**

**for(var i of edgesNames) {**

**console.log(i.source.name + " " + i.target.name)**

**}**

**// console.log(edgesNames)**

**console.log("checkEdges")**

**console.log("checkNodes")**

**for(var i of nodesNames) {**

**console.log(i.name)**

**}**

**// console.log(nodesNames)**

**console.log("checkNodes")**

**}**

**var bridges = Array(maxCountOfNodes)**

**function whiteEdges(){**

**for(var i of edgesNames){**

**i.data.color = 'white'**

**}**

**}**

**var APs = Array(maxCountOfNodes)**

**function greenNodes(){**

**for(var i of nodesNames){**

**i.data.color = 'green'**

**}**

**}**

**var mapArray = Array()**

**var mapArray2 = Array()**

**var openRange = false**

**var openRange2 = false**

**class GraphMaker {**

**constructor(vertices){**

**this.V= vertices**

**this.graph = new Array(this.V)**

**for(var i = 0; i < this.graph.length; i++)**

**this.graph[i] = new Array(0)**

**this.Time = 0**

**}**

**addEdge(u, v){**

**console.log('u ' + u + ' - ' + this.graph[u] + ' + ' + v)**

**// console.log(v + ' ' + this.graph[v])**

**// if(this.graph[u] == undefined)**

**// this.graph[u] = Array(0)**

**// if(this.graph[v] == undefined)**

**// this.graph[v] = Array(0)**

**this.graph[u].push(v)**

**console.log('v ' + v + ' - ' + this.graph[v] + ' + ' + u)**

**this.graph[v].push(u)**

**}**

**bridgeUtil(u, visited, parent, low, disc){**

**visited[u] = true**

**disc[u] = this.Time**

**low[u] = this.Time**

**this.Time += 1**

**for(const v of this.graph[u]){**

**if (visited[v] == false){**

**mapArray2.push(new Pair(u, v))**

**parent[v] = u**

**this.bridgeUtil(v, visited, parent, low, disc)**

**low[u] = Math.min(low[u], low[v])**

**if (low[v] > disc[u]) {**

**bridges[u].push(v)**

**}**

**}**

**else**

**if (v != parent[u])**

**low[u] = Math.min(low[u], disc[v])**

**}**

**}**

**APUtil(u, visited, ap, parent, low, disc){**

**mapArray.push(u)**

**var children = 0**

**visited[u] = true**

**disc[u] = this.Time**

**low[u] = this.Time**

**this.Time += 1**

**for(const v of this.graph[u]){**

**if(visited[v] == false) {**

**parent[v] = u**

**children += 1**

**this.APUtil(v, visited, ap, parent, low, disc)**

**low[u] = Math.min(low[u], low[v])**

**if(parent[u] == -1 && children > 1)**

**ap[u] = true**

**if(parent[u] != -1 && low[v] >= disc[u])**

**ap[u] = true**

**}**

**else**

**if(v != parent[u]){**

**low[u] = Math.min(low[u], disc[v])**

**}**

**}**

**}**

**AP(){**

**const visited = Array(this.V).fill(false)**

**const disc = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const low = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const parent = Array(this.V).fill(-1)**

**const ap = Array(this.V).fill(false)**

**for(var i = 0; i < this.V; i++){**

**if(visited[i] == false)**

**this.APUtil(i, visited, ap, parent, low, disc)**

**}**

**for(var i = 0; i < this.V; i++){**

**if(ap[i] == true) {**

**// console.log(i + ' AP')**

**APs[i] = 1**

**}**

**}**

**}**

**bridge(){**

**const visited = Array(this.V).fill(false)**

**const disc = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const low = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const parent = Array(this.V).fill(-1)**

**for(var i = 0; i < this.V; i++){**

**if(visited[i] == false)**

**this.bridgeUtil(i, visited, parent, low, disc)**

**}**

**// console.log("Bridge alg")**

**}**

**}**

**function findBridges(){**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "visible"**

**openRange2 = true**

**mapArray2 = new Array()**

**counter2 = 0**

**for(var i = 0; i < bridges.length; i++)**

**bridges[i] = new Array(0)**

**whiteEdges()**

**var g3 = new GraphMaker(nodesNames.size + delCount)**

**// checkEdges()**

**console.log("findBridges")**

**for(var i of edgesNames)**

**{**

**// console.log(i.source.name + " " + i.target.name)**

**console.log(g3.graph)**

**g3.addEdge(parseInt(i.source.name), parseInt(i.target.name))**

**}**

**g3.bridge()**

**for(var i = 0; i < bridges.length; i++) {**

**if(bridges[i].length != 0)**

**for(var j of bridges[i]) {**

**for (var e of edgesNames) {**

**if (i == parseInt(e.source.name) && j == parseInt(e.target.name) || j == parseInt(e.source.name) && i == parseInt(e.target.name)) {**

**// e.data({'color':'red'})**

**// console.log(e.source.name + " " + e.target.name + " find bridges")**

**e.data.color = 'red'**

**}**

**}**

**}**

**}**

**console.log(mapArray2)**

**}**

**var delCount = 0**

**function findAP(){**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "visible"**

**mapArray = new Array()**

**counter = 0**

**openRange = true**

**for(var i = 0; i < APs.length; i++)**

**APs[i] = -1**

**greenNodes()**

**const g3 = new GraphMaker(nodesNames.size + delCount)**

**// checkEdges()**

**console.log("findAPs")**

**for(var i of edgesNames)**

**{**

**// console.log(i.source.name + " " + i.target.name)**

**try {**

**g3.addEdge(parseInt(i.source.name), parseInt(i.target.name))**

**}**

**catch {**

**g3.addEdge(parseInt(i.target.name), parseInt(i.source.name))**

**}**

**}**

**g3.AP()**

**for(var i = 0; i < APs.length; i++) {**

**if(APs[i] != -1)**

**for (var n of nodesNames){**

**if(n.name == i)**

**n.data.color = 'red'**

**}**

**}**

**// var rng = document.getElementById('r1')**

**// rng.setAttribute('max', mapArray.length)**

**// console.log('AP alg')**

**}**

**var sys = arbor.ParticleSystem()**

**sys.parameters({gravity:false})**

**sys.renderer = Renderer("#viewport")**

**var nodesNames = new Set()**

**var edgesNames = new Set()**

**var n1 = sys.addNode('0',{'color':'green','shape':'dot','label':'node\_0'})**

**var n2 = sys.addNode('1',{'color':'green','shape':'dot','label':'node\_1'})**

**var e1 = sys.addEdge(n1, n2)**

**nodesNames.add(n1)**

**nodesNames.add(n2)**

**edgesNames.add(e1)**

**e1.data.color = 'white'**

**function selectorOptAdd(link, input){**

**var myselect = document.querySelector(link)**

**var currentText = input**

**var objOption = document.createElement("option")**

**objOption.text = currentText**

**objOption.value = currentText**

**//myselect.add(objOption)**

**myselect.options.add(objOption)**

**}**

**function createDot(){**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**// var input = document.querySelector('#dotInput').value**

**var input = 0;**

**for(var i = 0; i < maxCountOfNodes; i++){**

**if(sys.getNode(i) == undefined) {**

**input = i**

**break**

**}**

**}**

**console.log(input + '\_input')**

**// if (isNaN(input)) {**

**// console.log('неправльный инпут')**

**// return**

**// }**

**// else**

**// console.log('парс прошел')**

**if(sys.getNode(input) == undefined && input < maxCountOfNodes) {**

**var n = sys.addNode(input, {'color': 'green', 'shape': 'dot', 'label': "node\_" + input})**

**nodesNames.add(n)**

**selectorOptAdd('#select1dot', input)**

**selectorOptAdd('#select2dot', input)**

**}**

**else {**

**console.log('много точек > ' + maxCountOfNodes + ' или точка уже создана')**

**alert('много точек > ' + maxCountOfNodes + ' или точка уже создана')**

**}**

**}**

**function deleteDot() {**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**var input = document.querySelector('#dotInput').value**

**if (nodesNames.size == 2) {**

**alert('Не удалить, если вершин 2')**

**return**

**}**

**// checkEdges()**

**// console.log("Deletion")**

**if (sys.getNode(input) != undefined) {**

**delCount += 1**

**for(const i of nodesNames) {**

**if(i.name == input) {**

**var edges = sys.getEdgesFrom(i)**

**for(var edge of edges)**

**if(edge.source.name == sys.getNode(input).name || edge.target.name == sys.getNode(input).name) {**

**// console.log(edge.source.name + ' ' + edge.target.name)**

**sys.pruneEdge(edge)**

**edgesNames.delete(edge)**

**}**

**var edges = sys.getEdgesTo(i)**

**for(var edge of edges)**

**if(edge.source.name == sys.getNode(input).name || edge.target.name == sys.getNode(input).name) {**

**// console.log(edge.source.name + ' ' + edge.target.name)**

**sys.pruneEdge(edge)**

**edgesNames.delete(edge)**

**}**

**sys.pruneNode(i)**

**nodesNames.delete(i)**

**}**

**}**

**var selectobject1 = document.querySelector('#select1dot')**

**for (var i=0; i<selectobject1.length; i++) {**

**if (selectobject1.options[i].value == input)**

**selectobject1.remove(i)**

**}**

**var selectobject2 = document.querySelector('#select2dot')**

**for (var i=0; i<selectobject2.length; i++) {**

**if (selectobject2.options[i].value == input)**

**selectobject2.remove(i)**

**}**

**}**

**else {**

**alert('Такой вершины нет (node\_' + input + ')')**

**}**

**console.log("ShowAfterDeletion")**

**// checkEdges()**

**}**

**window.onload = function(){**

**for(const i of nodesNames){**

**selectorOptAdd('#select1dot', i.name)**

**selectorOptAdd('#select2dot', i.name)**

**}**

**selectorOptAdd('#ownGraphSelect', 'firstGraph')**

**selectorOptAdd('#ownGraphSelect', 'secondGraph')**

**}**

**function createEdge(){**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**var input1 = document.querySelector('#select1dot').selectedIndex**

**var input2 = document.querySelector('#select2dot').selectedIndex**

**var options1 = document.querySelector('#select1dot').options**

**var options2 = document.querySelector('#select2dot').options**

**for(const i of edgesNames){**

**if(i.source.name == options1[input1].text && i.target.name == options2[input2].text || i.source.name == options2[input2].text && i.target.name == options1[input1].text){**

**console.log('ребро уже существует')**

**alert('ребро уже существует')**

**return**

**}**

**}**

**var edge = sys.addEdge(options1[input1].text, options2[input2].text);**

**edge.data.color = 'white'**

**edgesNames.add(edge)**

**}**

**function deleteEdge(){**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**var input1 = document.querySelector('#select1dot').selectedIndex**

**var input2 = document.querySelector('#select2dot').selectedIndex**

**var options1 = document.querySelector('#select1dot').options**

**var options2 = document.querySelector('#select2dot').options**

**for(const i of edgesNames){**

**if(i.source.name == options1[input1].text && i.target.name == options2[input2].text || i.source.name == options2[input2].text && i.target.name == options1[input1].text){**

**sys.pruneEdge(i)**

**edgesNames.delete(i)**

**}**

**}**

**}**

**function generateGraph() {**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**// var rng = document.getElementById('r1')**

**// rng.value = 0**

**var nodes = document.querySelector('#nInput').value**

**var edges = document.querySelector('#eInput').value**

**if(edges>(nodes\*(nodes-1)/2)){**

**alert('Слишком много ребер')**

**return**

**}**

**if(parseInt(nodes) > maxCountOfNodes || parseInt(nodes) < 2 || nodes == '') {**

**alert('Слишком много вершин > ' + maxCountOfNodes + ", или их слишком мало < 2, или задано пустое значение")**

**return**

**}**

**if(edges < 0) {**

**alert('Отрицательное количество ребер')**

**return**

**}**

**// edgesNames.forEach(function deletion(value1,value2,edgesNames)**

**// {**

**// sys.pruneEdge(value1)**

**// edgesNames.delete(value1);**

**// })**

**for(var i of edgesNames){**

**sys.pruneEdge(i)**

**edgesNames.delete(i)**

**}**

**for(var j of nodesNames)**

**{**

**sys.pruneNode(j)**

**nodesNames.delete(j)**

**var selectobject1 = document.querySelector('#select1dot')**

**for (var i=0; i<selectobject1.length; i++) {**

**selectobject1.remove(i)**

**}**

**var selectobject2 = document.querySelector('#select2dot')**

**for (var i=0; i<selectobject2.length; i++) {**

**selectobject2.remove(i)**

**}**

**}**

**// nodesNames.forEach(function deletion(value1,value2,nodesNames)**

**// {**

**// sys.pruneNode(value1)**

**// nodesNames.delete(value1);**

**// var selectobject1 = document.querySelector('#select1dot')**

**// for (var i=0; i<selectobject1.length; i++) {**

**// selectobject1.remove(i)**

**// }**

**// var selectobject2 = document.querySelector('#select2dot')**

**// for (var i=0; i<selectobject2.length; i++) {**

**// selectobject2.remove(i)**

**// }**

**// })**

**for(var i=0; i<nodes; i++){**

**selectorOptAdd('#select1dot', i)**

**selectorOptAdd('#select2dot', i)**

**var n = sys.addNode(i, {'color': 'green', 'shape': 'dot', 'label': 'node\_' + i})**

**nodesNames.add(n)**

**}**

**var k = 0**

**while(k < edges){**

**var a = Math.floor(Math.random() \* nodes)**

**var b = Math.floor(Math.random() \* nodes)**

**if((a!=b)&&((sys.getEdges(a, b).length==0)&&(sys.getEdges(b, a).length==0))){**

**var n = sys.addEdge(a, b)**

**n.data.color = 'white'**

**edgesNames.add(n)**

**k++;**

**}**

**}**

**}**

**var counter = 0**

**var keyHelp = true**

**function showSolveMap(a){**

**var excCounter = 0**

**if(openRange == false)**

**return**

**if(keyHelp == true)**

**{**

**alert('Обрати внимание, что алгоритм начинает работу с вершины "Node\_0"')**

**keyHelp = false**

**}**

**// var rng = document.getElementById('r1')**

**// rng.setAttribute('max', mapArray.length)**

**if(a == 1) {**

**if(counter >= mapArray.length - delCount) {**

**// sys.getNode(mapArray[it - 1]).data.color = 'green'**

**// greenNodes()**

**// for(var i = 0; i < APs.length; i++) {**

**// if(APs[i] != -1)**

**// for (var n of nodesNames){**

**// if(n.name == i)**

**// n.data.color = 'red'**

**// }**

**// }**

**counter = mapArray.length - delCount**

**return**

**}**

**// sys.getNode(mapArray[it - 1]).data.color = 'green'**

**try {**

**if (sys.getNode(mapArray[counter]).data.color == 'red')**

**sys.getNode(mapArray[counter]).data.color = 'orange'**

**else**

**sys.getNode(mapArray[counter]).data.color = 'blue'**

**counter += a**

**document.getElementById('showIt').innerHTML = counter + '/' + (mapArray.length - delCount)**

**} catch{**

**excCounter += 1**

**counter += a**

**document.getElementById('showIt').innerHTML = (counter - excCounter) + '/' + (mapArray.length - delCount)**

**}**

**}**

**if(a == -1) {**

**counter -= 1**

**if(counter <= 0) {**

**// sys.getNode(mapArray[it - 1]).data.color = 'green'**

**// findAP()**

**greenNodes()**

**for(var i = 0; i < APs.length; i++) {**

**if(APs[i] != -1)**

**for (var n of nodesNames){**

**if(n.name == i)**

**n.data.color = 'red'**

**}**

**}**

**counter = 0**

**return**

**}**

**// sys.getNode(mapArray[it - 1]).data.color = 'green'**

**try {**

**if (sys.getNode(mapArray[counter]).data.color == 'orange')**

**sys.getNode(mapArray[counter]).data.color = 'red'**

**else**

**sys.getNode(mapArray[counter]).data.color = 'green'**

**document.getElementById('showIt').innerHTML = counter + '/' + (mapArray.length - delCount)**

**} catch{**

**document.getElementById('showIt').innerHTML = (counter - excCounter) + '/' + (mapArray.length - delCount)**

**}**

**}**

**if(counter < 0)**

**counter = 0**

**if(counter > mapArray.length)**

**counter = mapArray.length - delCount**

**}**

**var counter2 = 0**

**function getEdge(n1, n2){**

**for(var edge of sys.getEdgesFrom(n1)){**

**if(edge.source.name == n1 && edge.target.name == n2 || edge.source.name == n2 && edge.target.name == n1){**

**console.log(edge)**

**return edge**

**}**

**}**

**for(var edge of sys.getEdgesTo(n1)){**

**if(edge.source.name == n1 && edge.target.name == n2 || edge.source.name == n2 && edge.target.name == n1){**

**console.log(edge)**

**return edge**

**}**

**}**

**}**

**function showSolveMap2(a){**

**if(openRange2 == false)**

**return**

**console.log("openRange2")**

**if(keyHelp == true)**

**{**

**alert('Обрати внимание, что алгоритм начинает работу с вершины "Node\_0"')**

**keyHelp = false**

**}**

**if(a == 1) {**

**if(counter2 >= mapArray2.length) {**

**counter2 = mapArray2.length**

**return**

**}**

**console.log(mapArray2[counter2].n1 + ' ' + mapArray2[counter2].n2)**

**if ((getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2)).data.color == 'red')**

**getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2).data.color = 'orange'**

**else**

**getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2).data.color = 'blue'**

**counter2 += a**

**document.getElementById('showIt2').innerHTML = counter2 + '/' + (mapArray2.length)**

**}**

**if(a == -1) {**

**counter2 -= 1**

**if(counter2 <= 0) {**

**whiteEdges()**

**for(var i = 0; i < bridges.length; i++) {**

**if(bridges[i].length != 0)**

**for(var j of bridges[i]) {**

**for (var e of edgesNames) {**

**if (i == parseInt(e.source.name) && j == parseInt(e.target.name) || j == parseInt(e.source.name) && i == parseInt(e.target.name)) {**

**e.data.color = 'red'**

**}**

**}**

**}**

**}**

**counter2 = 0**

**return**

**}**

**console.log(2222)**

**if (getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2).data.color == 'orange')**

**getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2).data.color = 'red'**

**else**

**getEdge(mapArray2[counter2].n1, mapArray2[counter2].n2).data.color = 'white'**

**document.getElementById('showIt2').innerHTML = counter2 + '/' + (mapArray2.length)**

**}**

**if(counter2 < 0)**

**counter2 = 0**

**if(counter2 > mapArray2.length)**

**counter2 = mapArray2.length**

**}**

**function generateFirstGraph(){**

**var n1 = sys.addNode('0',{'color':'green','shape':'dot','label':'node\_0'})**

**var n2 = sys.addNode('1',{'color':'green','shape':'dot','label':'node\_1'})**

**var n3 = sys.addNode('2',{'color':'green','shape':'dot','label':'node\_2'})**

**var n4 = sys.addNode('3',{'color':'green','shape':'dot','label':'node\_3'})**

**var n5 = sys.addNode('4',{'color':'green','shape':'dot','label':'node\_4'})**

**var n6 = sys.addNode('5',{'color':'green','shape':'dot','label':'node\_5'})**

**var e1 = sys.addEdge(n1, n2)**

**var e2 = sys.addEdge(n2, n3)**

**var e3 = sys.addEdge(n2, n4)**

**var e4 = sys.addEdge(n4, n5)**

**var e5 = sys.addEdge(n4, n6)**

**nodesNames.add(n1)**

**nodesNames.add(n2)**

**nodesNames.add(n3)**

**nodesNames.add(n4)**

**nodesNames.add(n5)**

**nodesNames.add(n6)**

**for(const i of nodesNames){**

**selectorOptAdd('#select1dot', i.name)**

**selectorOptAdd('#select2dot', i.name)**

**}**

**edgesNames.add(e1)**

**edgesNames.add(e2)**

**edgesNames.add(e3)**

**edgesNames.add(e4)**

**edgesNames.add(e5)**

**}**

**function generateSecondGraph(){**

**var n1 = sys.addNode('0',{'color':'green','shape':'dot','label':'node\_0'})**

**var n2 = sys.addNode('1',{'color':'green','shape':'dot','label':'node\_1'})**

**var n3 = sys.addNode('2',{'color':'green','shape':'dot','label':'node\_2'})**

**var n4 = sys.addNode('3',{'color':'green','shape':'dot','label':'node\_3'})**

**var n5 = sys.addNode('4',{'color':'green','shape':'dot','label':'node\_4'})**

**var n6 = sys.addNode('5',{'color':'green','shape':'dot','label':'node\_5'})**

**var e1 = sys.addEdge(n1, n2)**

**var e2 = sys.addEdge(n2, n3)**

**var e3 = sys.addEdge(n4, n1)**

**var e4 = sys.addEdge(n5, n2)**

**var e5 = sys.addEdge(n6, n3)**

**var e6 = sys.addEdge(n3, n1)**

**nodesNames.add(n1)**

**nodesNames.add(n2)**

**nodesNames.add(n3)**

**nodesNames.add(n4)**

**nodesNames.add(n5)**

**nodesNames.add(n6)**

**for(const i of nodesNames){**

**selectorOptAdd('#select1dot', i.name)**

**selectorOptAdd('#select2dot', i.name)**

**}**

**edgesNames.add(e1)**

**edgesNames.add(e2)**

**edgesNames.add(e3)**

**edgesNames.add(e4)**

**edgesNames.add(e5)**

**edgesNames.add(e6)**

**}**

**function generateOwn(){**

**for(var i of edgesNames){**

**sys.pruneEdge(i)**

**edgesNames.delete(i)**

**}**

**for(var j of nodesNames)**

**{**

**sys.pruneNode(j)**

**nodesNames.delete(j)**

**var selectobject1 = document.querySelector('#select1dot')**

**for (var i=0; i<selectobject1.length; i++) {**

**selectobject1.remove(i)**

**}**

**var selectobject2 = document.querySelector('#select2dot')**

**for (var i=0; i<selectobject2.length; i++) {**

**selectobject2.remove(i)**

**}**

**}**

**var divBlockMap = document.querySelector('#divBlockMap')**

**divBlockMap.style.visibility = "hidden"**

**counter = 0**

**mapArray = new Array()**

**openRange = false**

**var divBlockMap2 = document.querySelector('#divBlockMap2')**

**divBlockMap2.style.visibility = "hidden"**

**counter2 = 0**

**mapArray2 = new Array()**

**openRange2 = false**

**var input1 = document.querySelector('#ownGraphSelect').selectedIndex**

**var options1 = document.querySelector('#ownGraphSelect').options**

**if(options1[input1].text == 'firstGraph')**

**generateFirstGraph()**

**if(options1[input1].text == 'secondGraph')**

**generateSecondGraph()**

**}**

## **Код класса Help.js**

**var modal = document.getElementById('myModal')**

**var btn = document.getElementById('myBtn')**

**var span = document.getElementsByClassName('close')[0]**

**var p = document.getElementById('p1')**

**btn.onclick = function(){**

**modal.style.display = "block"**

**document.querySelector('#p1').innerHTML = "Для построение графа используйте панель иснтрументов для построения графа, чтобы пройтись по шагам алгоритма, вам следует " +**

**"решить алгоритм с помощью 2 кнопок Блока 'GraphSolver'" +**

**". Блок 'Node' нужен для построения/удаления вершины, блок 'Edge' нужен для построения/удаления ребра, блок 'GenGraph' нужен для построения произвольного графа" +**

**" с заданными количествами вершин и ребер. "**

**document.querySelector('#p1').innerHTML +=**

**"При прохождении по алгоритму точек сочленения, пройденные точки окрашиваются в синий, если они не являются точками сочленения, в противном случае окрашиваются в желтый. " +**

**"При прохождении по алгоритму нахождения мостов в графе, пройденные мосты окрашиваются в красный, если они являются мостами, в противном случае окрашиваются в синий. Если же он остается белым," +**

**" то он не был посещен и не является мостом (не был посещен из-за того, что мост ведет в посещенную вершину)."**

**}**

**span.onclick = function(){**

**modal.style.display = "none"**

**}**

**// p.onclick = function (){**

**// modal.style.display = "none"**

**// }**

**window.onclick = function(event){**

**if(event.target == modal){**

**modal.style.display = "none"**

**}**

**}**

**document.getElementById('myModal').onmouseover = function(event){**

**var target = event.target**

**if(target.className == 'close'){**

**span.style.cursor = 'pointer'**

**}**

**// if(target.className == 'p1'){**

**// p.style.cursor = 'pointer'**

**// }**

**if(target.className != 'modal-content' && target.className == 'modal'){**

**modal.style.cursor = 'pointer'**

**}**

**else{**

**modal.style.cursor = 'default'**

**}**

**}**

## **Код класса SiteMaker.js**

var slideIndex = 1

showSlides(slideIndex)

function plusSlides(n){

showSlides(slideIndex += n)

}

function currentSlide(n){

showSlides(slideIndex = n)

}

function showSlides(n){

var i

var slides = document.getElementsByClassName("mySlides")

var dots = document.getElementsByClassName("dot")

if(n > slides.length){

slideIndex = 1

}

if(n < 1){

slideIndex=slides.length

}

for (i = 0; i < slides.length; i++){

slides[i].style.display = "none"

}

for (i = 0; i < dots.length; i++){

dots[i].className = dots[i].className.replace("active", "")

}

slides[slideIndex - 1].style.display = "block"

dots[slideIndex - 1].className += " active"

}

// function drawPoint(){

// var canvas = document.querySelector('.canvas')

// canvas.insertAdjacentHTML(

// 'beforeend',

// `<div class="point"></div>`

// )

// }

//

// function drawEdge(){

// var canvas = document.querySelector('.canvas')

// canvas.insertAdjacentHTML(

// 'beforeend',

// `<div class="edge"></div>`

// )

// }

// document.querySelector('.canvas').onmouseover = function (event){

// var target = event.target

// if(target.className == 'point' || target.className == 'edge'){

// document.addEventListener('keyup', function (eve){

// if(eve.key == 'Delete'){

// target.remove()

// }

// })

// }

// }

## **Код класса Pair.js**

class Pair{

constructor(n1, n2){

this.n1 = n1

this.n2 = n2

}

}

## **Код класса GraphMaker.js**

**class GraphMaker {**

**constructor(vertices){**

**this.V= vertices**

**this.graph = new Array(this.V)**

**for(var i = 0; i < this.graph.length; i++)**

**this.graph[i] = new Array(0)**

**this.Time = 0**

**}**

**addEdge(u, v){**

**console.log('u ' + u + ' - ' + this.graph[u] + ' + ' + v)**

**// console.log(v + ' ' + this.graph[v])**

**// if(this.graph[u] == undefined)**

**// this.graph[u] = Array(0)**

**// if(this.graph[v] == undefined)**

**// this.graph[v] = Array(0)**

**this.graph[u].push(v)**

**console.log('v ' + v + ' - ' + this.graph[v] + ' + ' + u)**

**this.graph[v].push(u)**

**}**

**bridgeUtil(u, visited, parent, low, disc){**

**visited[u] = true**

**disc[u] = this.Time**

**low[u] = this.Time**

**this.Time += 1**

**for(const v of this.graph[u]){**

**if (visited[v] == false){**

**mapArray2.push(new Pair(u, v))**

**parent[v] = u**

**this.bridgeUtil(v, visited, parent, low, disc)**

**low[u] = Math.min(low[u], low[v])**

**if (low[v] > disc[u]) {**

**bridges[u].push(v)**

**}**

**}**

**else**

**if (v != parent[u])**

**low[u] = Math.min(low[u], disc[v])**

**}**

**}**

**APUtil(u, visited, ap, parent, low, disc){**

**mapArray.push(u)**

**var children = 0**

**visited[u] = true**

**disc[u] = this.Time**

**low[u] = this.Time**

**this.Time += 1**

**for(const v of this.graph[u]){**

**if(visited[v] == false) {**

**parent[v] = u**

**children += 1**

**this.APUtil(v, visited, ap, parent, low, disc)**

**low[u] = Math.min(low[u], low[v])**

**if(parent[u] == -1 && children > 1)**

**ap[u] = true**

**if(parent[u] != -1 && low[v] >= disc[u])**

**ap[u] = true**

**}**

**else**

**if(v != parent[u]){**

**low[u] = Math.min(low[u], disc[v])**

**}**

**}**

**}**

**AP(){**

**const visited = Array(this.V).fill(false)**

**const disc = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const low = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const parent = Array(this.V).fill(-1)**

**const ap = Array(this.V).fill(false)**

**for(var i = 0; i < this.V; i++){**

**if(visited[i] == false)**

**this.APUtil(i, visited, ap, parent, low, disc)**

**}**

**for(var i = 0; i < this.V; i++){**

**if(ap[i] == true) {**

**// console.log(i + ' AP')**

**APs[i] = 1**

**}**

**}**

**}**

**bridge(){**

**const visited = Array(this.V).fill(false)**

**const disc = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const low = Array(this.V).fill(Number.MAX\_SAFE\_INTEGER)**

**const parent = Array(this.V).fill(-1)**

**for(var i = 0; i < this.V; i++){**

**if(visited[i] == false)**

**this.bridgeUtil(i, visited, parent, low, disc)**

**}**

**// console.log("Bridge alg")**

**}**

**}**

## **Код style.css**

.slideshow-container{

max-width: 1500px;

position: relative;

margin: auto;

}

.prev, .next {

cursor: pointer;

position: absolute;

top: 50%;

width: auto;

margin-top: -22px;

padding: 16px;

color: white;

font-weight: bold;

font-size: 18px;

transition: 0.6s ease;

border-radius: 0 3px 3px 0;

}

.next{

right: 0px;

border-radius: 3px 0 0 3px;

}

.prev:hover, .next:hover{

background-color: rgba(0, 0, 0, 0.8);

}

.text{

color: #f2f2f2;

font-size: 20px;

padding: 8px 12px;

width: 100%;

position: absolute;

bottom: 0;

}

.dot{

cursor: pointer;

height: 13px;

width: 13px;

margin: 0 2px;

background-color: #bbb;

border-radius: 50%;

display: inline-block;

transition: background-color 0.6s ease;

}

.active, .dot:hover{

background-color: #717171;

}

.fade{

-webkit-animation-name: fade;

-moz-animation-duration: 2s;

animation-name: fade;

animation-duration: 2s;

text-align: center;

}

@-webkit-keyframes fade {

from{opacity: 0.5}

to{opacity: 1}

}

@keyframes fade {

from{opacity: 0.5}

to{opacity: 1}

}

.rectangle{

width: 100%;

height: 900px;

background: #162528;

}

.circles{

margin-left: 50%;

}

.tools{

height: 900px;

width: 15%;

position: absolute;

left: 85%;

background: #124f57;

/\*border: 1px double black;\*/

}

.canvasCont{

height: 900px;

width: 85%;

position: absolute;

/\*border: 1px solid black;\*/

}

.canvas{

height: 90%;

width: 90%;

position: absolute;

border: 1px solid black;

margin: 3% 5%;

z-index: 3;

}

.toolBlock{

position: relative;

width: 100%;

height: 12%;

}

.toolText{

color: #f2f2f2;

font-size: 20px;

text-align: center;

width: 80%;

height: 10%;

position: absolute;

left: 20%;

}

.pointTool{

height: 20px;

width: 20px;

background-color: greenyellow;

position: absolute;

border-radius: 50%;

margin-left: 10%;

}

/\*.point{\*/

/\* height: 20px;\*/

/\* width: 20px;\*/

/\* background-color: greenyellow;\*/

/\* border-radius: 50%;\*/

/\* cursor: pointer;\*/

/\*}\*/

.edgeTool{

height: 8px;

width: 40px;

background-color: greenyellow;

position: absolute;

margin-left: 10%;

margin-top: 8px;

}

/\*.edge{\*/

/\* height: 8px;\*/

/\* width: 40px;\*/

/\* background-color: greenyellow;\*/

/\* cursor: pointer;\*/

/\*}\*/

/\*#viewport{\*/

/\* width: 850px;\*/

/\* height: 540px;\*/

/\*}\*/

#projetName{

width: 100%;

position: absolute;

top: 0;

text-align: center;

color: #c98484; font-family: ‘Lato’, sans-serif;

font-size: 50px;

font-weight: 300;

line-height: 58px;

margin: 0 0 70px;

margin-top: 20px;

}

.modal{

display: none;

position: fixed;

z-index: 5;

left: 0;

top: 0;

width: 100%;

height: 100%;

overflow: auto;

background-color: rgba(0,0,0,0.8);

}

.modal-content{

background-color: #fefefe;

margin: 15% auto;

padding: 20px;

border: 1px solid #888;

width: 80%;

}

.close{

color: #aaa;

float: right;

font-size: 28px;

font-weight: bold;

}

#myModal .modal-content #p1{

/\*background-color: black;\*/

padding: 0px;

width: 100%;

}

## **Код GraphMakerSolver.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>GraphSolver</title>**

**<link rel="stylesheet" href = "style.css">**

**<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>**

**<script src="/lib/arbor.js"></script>**

**<script src="/lib/graphics.js"></script>**

**<script src="/lib/renderer.js"></script>**

**<script src="/lib/arbor-tween.js"></script>**

**<style>**

**body {**

**background-size: cover;**

**background-image: url(https://images-wixmp-ed30a86b8c4ca887773594c2.wixmp.com/f/b0b7c577-941a-496d-a656-32c5732d82ff/d676ife-5ce481e4-9a57-40ca-8dbc-b67a78fed490.gif?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJ1cm46YXBwOjdlMGQxODg5ODIyNjQzNzNhNWYwZDQxNWVhMGQyNmUwIiwic3ViIjoidXJuOmFwcDo3ZTBkMTg4OTgyMjY0MzczYTVmMGQ0MTVlYTBkMjZlMCIsImF1ZCI6WyJ1cm46c2VydmljZTpmaWxlLmRvd25sb2FkIl0sIm9iaiI6W1t7InBhdGgiOiIvZi9iMGI3YzU3Ny05NDFhLTQ5NmQtYTY1Ni0zMmM1NzMyZDgyZmYvZDY3NmlmZS01Y2U0ODFlNC05YTU3LTQwY2EtOGRiYy1iNjdhNzhmZWQ0OTAuZ2lmIn1dXX0.d89DEDmSf7O5iq6sjsHU-VGZoCcFe--SX3usSSrRsJo); /\* Путь к фоновому изображению \*/**

**}**

**</style>**

**</head>**

**<body>**

**<div class="slideshow-container">**

**<div class="mySlides fade">**

**<div class="numbertext">1/2</div>**

**<div class="rectangle" id="mainRec" style="border-radius: 5%">**

**<div id="projetName">Приложение для визуализации**

**алгоритма поиска мостов и точек**

**сочленения в графе</div>**

**<div style="height: 75%; width: 40%; position: absolute; left: 5%; margin-top: 7%">**

**<h1 style="color: white"> Определение "Мосты"</h1>**

**<p style="color: #43bec0; font-size: 20px">Пусть дан неориентированный граф. Мостом называется такое ребро, удаление которого делает граф несвязным (или, точнее, увеличивает число компонент связности). Требуется найти все мосты в заданном графе.**

**Неформально, эта задача ставится следующим образом: требуется найти на карте такие дороги, при удалении которых пропадает путь между какими-либо двумя точками.</p>**

**<h1 style="color: white"> Идея "Мосты"</h1>**

**<p style="color: #43bec0; font-size: 20px">Мы можем заметить тот факт, что ребро между точками v и u является мостом тогда и только тогда, когда из вершины u и её потомков нельзя вернуться в вершину v или подняться выше нее. Осталось лишь понять, как нам это выяснить.</p>**

**<h1 style="color: white"> Определение "Точки сочленения"</h1>**

**<p style="color: #43bec0; font-size: 20px">Точкой сочленения называется вершина, удаление которой делает граф несвязным.</p>**

**<h1 style="color: white"> Идея "Точки сочленения"</h1>**

**<p style="color: #43bec0; font-size: 20px">Мы можем заметить два факта:**

**Рассмотрим ребро между вершинами v и u. Тогда если из вершины u и ее потомков нельзя попасть в какого-либо предка вершины v и притом вершина v не является корнем дерева, то данная вершина и есть точка сочленения**

**Если вершина v - корень дерева, то она является точкой сочленения тогда и только тогда, когда эта точка имеет более одного сына в обходе графа в глубину</p>**

**</div>**

**<div style="height: 75%; width: 40%; position: absolute; left: 55%; margin-top: 7%">**

**<img style="margin-top: 22%; height: 55%; width: 100%" src="/img/img.png">**

**<p style="color: #a052b7; font-size: 20px; position: absolute; top: 75%"> Для построения графа нажмите стрелочку вправо, чтобы перейти на следующую страницу.</p>**

**</div>**

**<div class="text" style="font-size: 25px">Общая информация</div>**

**</div>**

**</div>**

**<div class="mySlides fade">**

**<div class="numbertext">2/2</div>**

**<div class="rectangle" style="background: #0e2425; border-radius: 2%">**

**<div class="canvasCont">**

**<div class="canvas">**

**<!-- <canvas id="viewport" height="540" width="765"></canvas>-->**

**<canvas id="viewport" height="810" width="1147"></canvas>**

**<script src="graphMaker.js"></script>**

**</div>**

**</div>**

**<div class="tools" style="border-radius: 5%">**

**<div style="text-align: center; color: white; padding-top: 10px">Инструменты для построения графа</div>**

**<hr>**

**<div class="toolBlock" style="height: 10%">**

**<div class="pointTool" style="cursor: pointer"></div>**

**<div class="toolText">Node</div>**

**<!-- <input type="text" id="dotInput" placeholder="enter Node number" style="margin-top: 25px; width: 120px; cursor: pointer">-->**

**<button onclick="createDot()" style="width: 128px; cursor: pointer ;margin-top: 25px">CreateDot</button>**

**<button onclick="deleteDot()" style="width: 128px; cursor: pointer">DeleteDot</button>**

**<input type="text" id="dotInput" placeholder="Delete Node number" style="width: 120px; cursor: pointer">**

**</div>**

**<hr>**

**<div class="toolBlock">**

**<div class="edgeTool" style="cursor: pointer"></div>**

**<div class="toolText">Edge</div>**

**<select id="select1dot" style="margin-top: 25px; width: 128px; cursor: pointer"></select>**

**<select id="select2dot" style="width: 128px; cursor: pointer"></select>**

**<button onclick="createEdge()" style="width: 128px; cursor: pointer">CreateEdge</button>**

**<button onclick="deleteEdge()" style="width: 128px; cursor: pointer">DeleteEdge</button>**

**</div>**

**<hr>**

**<div class="toolBlock" style="height: 8%">**

**<div class="toolText" style="text-align: left">graphSolver</div>**

**<button onclick="findBridges()" style="width: 128px; cursor: pointer; margin-top: 25px">FindBridges</button>**

**<button onclick="findAP()" style="width: 128px; cursor: pointer">FindAPs</button>**

**</div>**

**<hr>**

**<div class="toolBlock" style="color: #f2f2f2; font-size: 20px; text-align: center; height: 16%">**

**<div style="text-align: center">graphGeneration</div>**

**<input type="text" id="nInput" placeholder="nodes count" style="margin-top: 6px; width: 120px; cursor: pointer;">**

**<input type="text" id="eInput" placeholder="edges count" style="width: 120px; cursor: pointer">**

**<button onclick="generateGraph()" style="width: 128px; cursor: pointer">genGraph</button>**

**<select id="ownGraphSelect" style="width: 128px; cursor: pointer"></select>**

**<button onclick="generateOwn()" style="width: 128px; cursor: pointer">developer'sGraph</button>**

**</div>**

**<hr>**

**<div class="toolBlock" id="divBlockMap2" style="color: #f2f2f2; font-size: 20px; text-align: center; height: 10%; visibility: hidden">**

**<div style="text-align: center">BridgesSolveMap</div>**

**<button onclick="showSolveMap2(1)" style="width: 128px; cursor: pointer; margin-top: 6px">NextStep</button>**

**<button onclick="showSolveMap2(-1)" style="width: 128px; cursor: pointer">PrevStep</button>**

**<div id="showIt2">0</div>**

**</div>**

**<hr>**

**<div class="toolBlock" id="divBlockMap" style="color: #f2f2f2; font-size: 20px; text-align: center; height: 10%; visibility: hidden">**

**<div style="text-align: center">APsSolveMap</div>**

**<button onclick="showSolveMap(1)" style="width: 128px; cursor: pointer; margin-top: 6px">NextStep</button>**

**<button onclick="showSolveMap(-1)" style="width: 128px; cursor: pointer">PrevStep</button>**

**<div id="showIt">0</div>**

**</div>**

**<hr>**

**<div class="toolBlock" style="font-size: 20px; text-align: center; height: 3%;position: absolute; bottom: 1%">**

**<button id="myBtn">Help</button>**

**<div id="myModal" class="modal">**

**<div class="modal-content">**

**<span class="close">&times;</span>**

**<p class="p1" id="p1">Модальное окно!</p>**

**</div>**

**</div>**

**<script src="help.js"></script>**

**</div>**

**</div>**

**<!-- <div class="text">Форма для графа</div>-->**

**</div>**

**</div>**

**<a class="prev" onclick="plusSlides(-1)">&#10094</a>**

**<a class="next" onclick="plusSlides(1)">&#10095</a>**

**</div>**

**<br>**

**<div class="circles">**

**<span class="dot" onclick="currentSlide(1)"></span>**

**<span class="dot" onclick="currentSlide(2)"></span>**

**</div>**

**<script src="siteMaker.js"></script>**

**</body>**

**</html>**

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |