МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Выполнила: студентка 2 курса 5 группы

специальности ПОИТ Вовна Я. Р.

**Отчёт**

По дисциплине “Математическое программирование”

На тему “Алгоритмы на графах”

Минск

2024

**Лабораторная работа 6. Алгоритмы на графах.**

**Цель работы:** Освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов. Разобрать алгоритм Прима и алгоритм Крускала

**Ход Работы**

1. **Представление графа в виде матрицы смежности, матрицы**

**инцидентности, списка смежных вершин.**

Вариант 4



e10

e8

e7

e9

e6

e5

e4

e3

e2

e1

Матрица смежности:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | v0 | v1 | v2 | v3 | v4 | v5 | v6 |
| v0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| v1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| v2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| v3 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| v4 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| v5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| v6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | e1 | e2 | e3 | e4 | e5 | e6 | e7 | e8 | e9 | e10 |
| v0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| v1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| v2 | -1 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| v3 | 0 | -1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| v4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 1 |
| v5 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | 0 | 0 |
| v6 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | -1 | 0 | 0 |

Список смешных вершин:

V0: V2, V3

V1: –

V2: V5

V3: V2, V4, V5, V6

V4: V1, V6

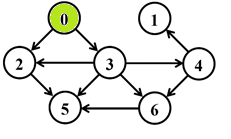
V5: –

V6: V5

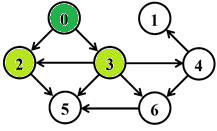
1. **Алгоритмы поиска в ширину и глубину, топологическая сортировка**

**Поиск в ширину (BFS)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **0** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **G** | **W** | **W** | **W** | **W** | **W** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **I** | **I** | **I** | **I** | **I** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **N** | **N** | **N** | **N** | **N** |

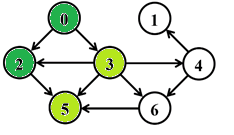
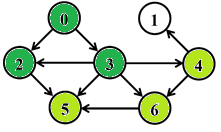


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **2** | **3** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **W** | **G** | **G** | **W** | **W** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **I** | **1** | **1** | **I** | **I** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **N** | **N** | **N** |

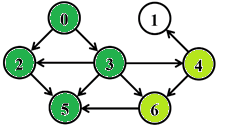


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **3** | **5** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **W** | **B** | **G** | **W** | **G** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **I** | **1** | **1** | **I** | **2** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **N** | **2** | **N** |

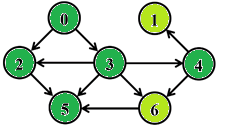
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **5** | **4** | **6** |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **W** | **B** | **B** | **G** | **G** | **G** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **I** | **1** | **1** | **2** | **2** | **2** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **3** | **2** | **3** |



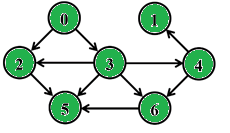
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **4** | **6** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **W** | **B** | **B** | **G** | **B** | **G** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **I** | **1** | **1** | **2** | **2** | **2** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **3** | **2** | **3** |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | **6** | **1** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **G** | **B** | **B** | **B** | **B** | **G** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **3** | **1** | **1** | **2** | **2** | **2** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **3** |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **0** | **2** | **3** | **5** | **4** | **6** | **1** |
|  |  |  |  |  |  |  |  |
| **C** | **B** | **B** | **B** | **B** | **B** | **B** | **B** |
|  |  |  |  |  |  |  |  |
| **D** | **0** | **3** | **1** | **1** | **2** | **2** | **2** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **3** |



5

2

0

15

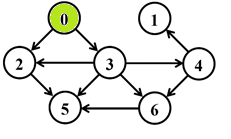
6

4

3

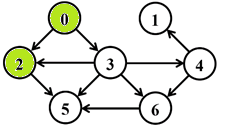
**Поиск в глубину (DFS)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **W** | **W** | **W** | **W** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **I** | **I** | **I** | **I** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **N** | **N** | **N** | **N** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |



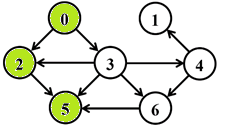
t = 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **G** | **W** | **W** | **W** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **I** | **I** | **I** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **N** | **N** | **N** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

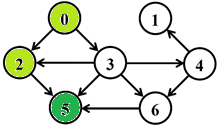


t = 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **G** | **W** | **W** | **G** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **I** | **I** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **N** | **N** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

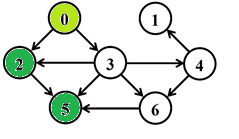


t = 3



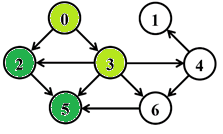
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **G** | **W** | **W** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **I** | **I** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **N** | **N** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **0** | **0** | **0** | **4** | **0** |

t = 4



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **B** | **W** | **W** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **I** | **I** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **N** | **N** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **5** | **0** | **0** | **4** | **0** |

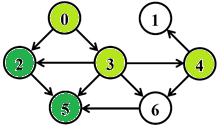
t = 5



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **B** | **G** | **W** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **6** | **I** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **N** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **5** | **0** | **0** | **4** | **0** |

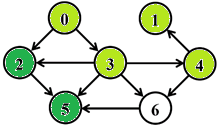
t = 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **W** | **B** | **G** | **G** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **I** | **2** | **6** | **7** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **N** | **0** | **0** | **N** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **5** | **0** | **0** | **4** | **0** |

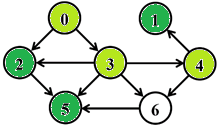


t = 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **G** | **B** | **G** | **G** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **0** | **5** | **0** | **0** | **4** | **0** |



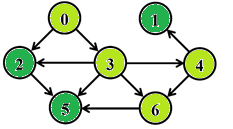
t = 8



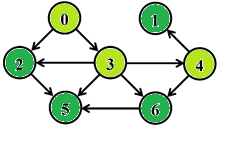
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **B** | **B** | **G** | **G** | **B** | **W** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **I** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **N** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **9** | **5** | **0** | **0** | **4** | **0** |

t = 9

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **B** | **B** | **G** | **G** | **B** | **G** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **10** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **4** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **9** | **5** | **0** | **0** | **4** | **0** |



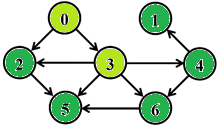
t = 10



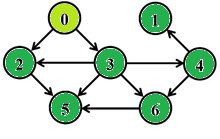
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **B** | **B** | **G** | **G** | **B** | **B** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **10** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **4** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **9** | **5** | **0** | **0** | **4** | **11** |

t = 11

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **B** | **B** | **G** | **B** | **B** | **B** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **10** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **4** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **9** | **5** | **0** | **12** | **4** | **11** |

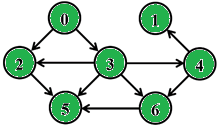


t = 12



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **G** | **B** | **B** | **B** | **B** | **B** | **B** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **10** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **4** |
|  |  |  |  |  |  |  |  |
| **F** | **0** | **9** | **5** | **13** | **12** | **4** | **11** |

t = 13



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **С** | **B** | **B** | **B** | **B** | **B** | **B** | **B** |
|  |  |  |  |  |  |  |  |
| **D** | **1** | **8** | **2** | **6** | **7** | **3** | **10** |
|  |  |  |  |  |  |  |  |
| **P** | **N** | **4** | **0** | **0** | **3** | **2** | **4** |
|  |  |  |  |  |  |  |  |
| **F** | **14** | **9** | **5** | **13** | **12** | **4** | **11** |

t = 14

A: 5-> 2 -> 1 -> 6 -> 4 -> 3 -> 0

5

2

0

15

4

3

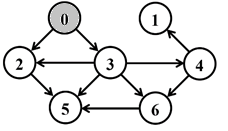
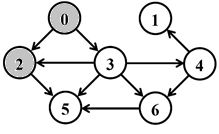
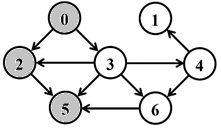
6

**Топологическая сортировка**

1/

1/

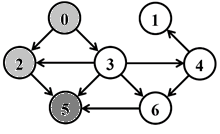
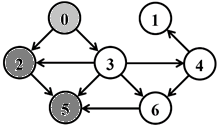
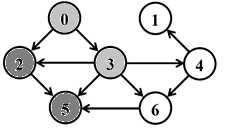
1/

**  **

3/

2/

2/

3/4

2/5

1/

2/5

1/

3/4

3/4

1/

2/

2 5

2 5

5

8/9

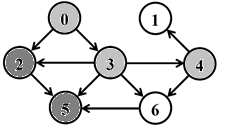
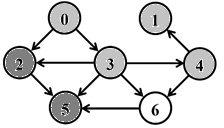
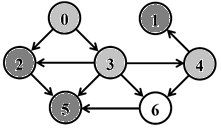
1/

8/

1/

3/4

1/

7/

6/

3/4

7/

2/5

7/

2/5

6/

6/

3/4

2/5

1 2 5

2 5

2 5

3/4

1/

7/8

6/

7/12

10/11

1/

8/9

1/

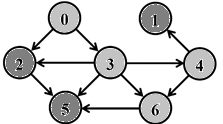
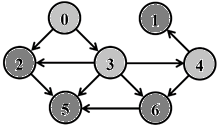
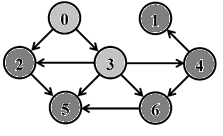
10/11

7/

6/

7/8

2/5

2/5

3/4

3/4

7/

6/

2/5

10/

4 6 1 2 5

6 1 2 5

1 2 5

7/12

6/13

7/8

1/14

2/5

3/4

10/11

7/12

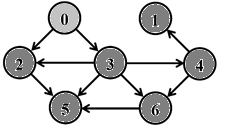
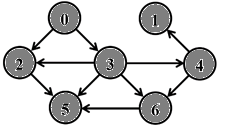
6/13

7/8

1/

2/5

3/4

10/11

0 3 4 6 1 2 5

3 4 6 1 2 5

1. **Разработка функций для преобразования способов представления,**

**функции для обхода вершин графа (поиск в ширину)**

Листинг:

AMatrix::AMatrix(int n)

{

this->n\_vertex = n;

this->mr = new int[this->n\_vertex \* this->n\_vertex];

for (int i = 0; i < n \* n; i++)mr[i] = 0;

};

AMatrix::AMatrix(int n, int mr[])

{

this->n\_vertex = n;

this->mr = mr;

};

AMatrix::AMatrix(const AMatrix& am)

{

this->n\_vertex = am.n\_vertex;

this->mr = new int[this->n\_vertex \* this->n\_vertex];

for (int i = 0; i < this->n\_vertex; i++)

for (int j = 0; j < this->n\_vertex; j++)

this->set(i, j, am.get(i, j));

};

AMatrix::AMatrix(const AList& al)

{

this->n\_vertex = al.n\_vertex;

this->mr = new int[this->n\_vertex \* this->n\_vertex];

for (int k = 0; k < this->n\_vertex \* this->n\_vertex; k++)mr[k] = 0;

for (int i = 0; i < this->n\_vertex; i++)

for (int j = 0; j < al.size(i); j++) this->set(i, al.get(i, j), 1);

};

void AMatrix::set(int i, int j, int r) { this->mr[i \* this->n\_vertex + j] = r; };

int AMatrix::get(int i, int j)const

{

return this->mr[i \* this->n\_vertex + j];

};

void AList::create(int n)

{

this->mr = new std::list<int>[this->n\_vertex = n];

};

AList::AList(int n) { create(n); }

AList::AList(const AMatrix& am)

{

this->create(am.n\_vertex);

for (int i = 0; i < this->n\_vertex; i++)

for (int j = 0; j < this->n\_vertex; j++)

if (am.get(i, j) != 0) this->add(i, j);

};

AList::AList(const AList& al)

{

this->create(al.n\_vertex);

for (int i = 0; i < this->n\_vertex; i++)

for (int j = 0; j < al.size(i); j++) this->add(i, al.get(i, j));

};

AList::AList(int n, int mr[])

{

this->create(n);

for (int i = 0; i < this->n\_vertex; i++)

for (int j = 0; j < this->n\_vertex; j++)

if (mr[i \* this->n\_vertex + j] != 0) this->add(i, j);

};

void AList::add(int i, int j) { this->mr[i].push\_back(j); };

int AList::size(int i) const { return (int)this->mr[i].size(); };

int AList::get(int i, int j)const

{

std::list<int>::iterator rc = this->mr[i].begin();

for (int k = 0; k < j; k++) rc++;

return (int)\*rc;

};

void BFS::init(const graph::AList& al, int s)

{

this->al = &al;

this->c = new Color[this->al->n\_vertex];

this->d = new int[this->al->n\_vertex];

this->p = new int[this->al->n\_vertex];

for (int i = 0; i < this->al->n\_vertex; i++)

{

this->c[i] = WHITE;

this->d[i] = INF;

this->p[i] = NIL;

};

this->c[s] = GRAY;

this->q.push(s);

};

BFS::BFS(const graph::AList& al, int s) { this->init(al, s); };

BFS::BFS(const graph::AMatrix& am, int s)

{

this->init(\*(new graph::AList(am)), s);

};

int BFS::get()

{

int rc = NIL, v = NIL;

if (!this->q.empty())

{

rc = this->q.front();

for (int j = 0; j < this->al->size(rc); j++)

if (this->c[v = this->al->get(rc, j)] == WHITE)

{

this->c[v] = GRAY;

this->d[v] = this->d[rc] + 1;

this->p[v] = rc;

this->q.push(v);

};

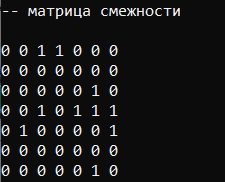
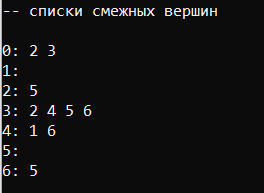
this->q.pop();

this->c[rc] = BLACK;

};

return rc;

}

* 1. **Разработать функцию DFS для обхода в глубину и топологической**

**сортировки**

Листинг:

void DFS::init(const graph::AList& al)

{

this->al = &al;

this->c = new Color[this->al->n\_vertex];

this->d = new int[this->al->n\_vertex];

this->f = new int[this->al->n\_vertex];

this->p = new int[this->al->n\_vertex];

this->t = 0;

for (int i = 0; i < this->al->n\_vertex; i++)

{

this->c[i] = WHITE;

this->d[i] = this->f[i] = 0;

this->p[i] = NIL;

};

for (int i = 0; i < this->al->n\_vertex; i++)

if (this->c[i] == WHITE)

{

this->visit(i);

this->topological\_sort.push\_back(i);

}

};

DFS::DFS(const graph::AList& al) { this->init(al); };

DFS::DFS(const graph::AMatrix& am)

{

this->init(\*(new graph::AList(am)));

};

void DFS::visit(int u)

{

int v = NIL;

this->c[u] = GRAY;

this->d[u] = ++(this->t);

for (int j = 0; j < this->al->size(u); j++)

if (this->c[v = this->al->get(u, j)] == WHITE)

{

this->p[v] = u;

this->visit(v);

this->topological\_sort.push\_back(v);

}

this->c[u] = BLACK;

this->f[u] = ++(this->t);

};

int DFS::get(int i)

{

int j = 0, min1 = INF, min2 = NINF, ntx = NIL;

for (int j = 0; j <= i; j++) // iая статистика

{

for (int k = 0; k < this->al->n\_vertex; k++)

if (this->f[k] < min1 && this->f[k] > min2)

{

min1 = this->f[k]; ntx = k;

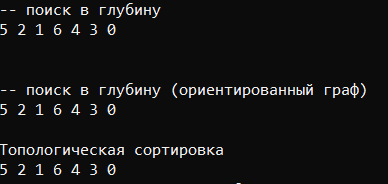
};

min2 = min1; min1 = INF;

};

return ntx;

};



1. **Алгоритм Прима**



1

9

11

6

1

3

10

6

9

2

1

9

11

6

1

3

10

6

9

2

1

9

11

6

1

3

10

6

9

2

1

9

11

6

1

3

10

6

9

2

2

1

9

11

6

1

3

10

6

9

1

9

11

6

1

3

10

6

9

2

1

9

11

6

1

3

10

6

9

2

2

1

9

11

6

1

3

10

6

9

1

9

11

6

1

3

10

6

9

2

1

6

1

3

6

2

1. **Алгоритм Крускала**

2

1

9

11

6

1

3

10

6

9

6

3

1

6

2

1

6

3

1

2

1

3

2

1

1

2

1

1

1

1

1

**Вывод:** в результате выполнения лабораторной работы были изучены основные алгоритмы на графах: поиск в ширину, поиск в глубину, топологическая сортировка, алгоритм Прима и алгоритм Крускала.