13 ГРАФТАРҒА АРНАЛҒАН АЛГОРИТМДЕР

13.1 Графтарды жүріп өту алгоритмдері

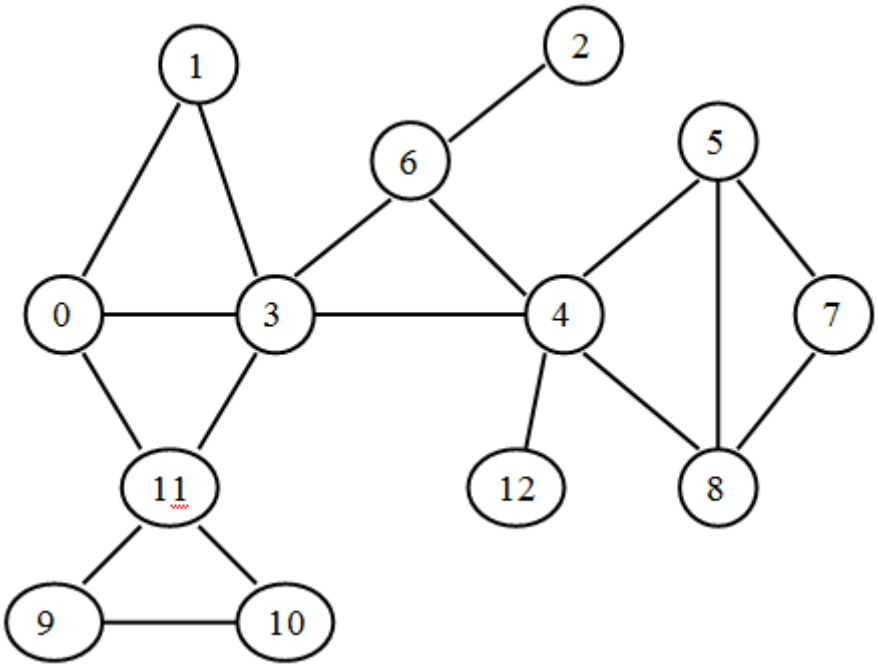
Графтың барлық төбелерін «қарап шығуды» қажет ететін көлік есептерінің тобы бар. Сонымен қатар әрбір төбе бір рет қана «қаралуы» керек.

Графтар теориясында граф төбелерін жүріп өтудің екі негізгі алгоритмі белгілі: графты «тереңдігі» бойынша және «көлденеңі» бойынша жүріп өту.

Бірінші жағдайда алгоритмде стек типіндегі құрылым, ал екінші жағдайда – кезек типіндегі құрылым қолданылады.

«Алгоритмдеу» тұрғысынан бірінші жағдайға көп мән беруге болады, өйткені онда күрделі құрылымдарда қолданылатын «қайтару» алгоритмі жазылады.

Графты «тереңдігі» бойынша жүріп өту алгоритмінің жұмысын 13.1-суретте көрсетілген, күрделілігі орташа мысалда қарастырайық.



13.1-сурет – Күрделілігі орташа граф

Графты «тереңдігі» бойынша жүріп өту алгоритмі графтың «жаңа» төбелері, яғни «қаралмаған» төбелері ұғымын қолданады. Оның жұмысын қарастырайық:

– графтың барлық төбелері «жаңа» деп жарияланады (Дейкстр алгоритмінде «уақытша» термині қолданылады). Графты қарап өтуді белгілі бір төбеден бастаймыз, мысалы 0-ші төбеден оны қарап өткеннен кейін оны «жаңа емес» деп белгілейміз;

– берілген төбемен қосылған бүкіл жаңа сыбайлас тізімнен нөмірі ең төмен төбені таңдаймыз. Қарастырылған мысалда ол 1-төбе;

– оны алғашқы деп жариялаймыз, оны қарап шыққаннан кейін оны «жаңа емес» деп белгілейміз;

– егер қарап өтілген төбенің жаңа сыбайлас төбелері болмаса, онда алдыңғы төбеге қайта ораламыз (стек жұмысы);

– процесс «жаңа» төбелер бар болғанша қайталанады.

13.1-суретте көрсетілген графқа қарастырылған алгоритмді қолданып төбелерді қарап шығу тізбегін аламыз:.

0 – 1 – 3 – 4 – 5 – 7 – 8 – 6 – 2 – 12 – 11 – 9 – 10.

Графты «көлденеңі» бойынша қарастырғанда да графтың «жаңа» төбелері деген ұғымды қолданады. Оның жұмысын қарастырайық:

– графтың барлық төбелері «жаңа» деп жарияланады. Графты жүріп өтуді белгілі бір алғашқы төбеден бастайық, мысалы, 0-төбеден. Ол төбені қарап шыққаннан (төбе нөмерін басамыз) кейін оны «жаңа емес» деп белгілейміз;

– қарап шыққан төбемен байланысты барлық жаңа сыбайлас төбелер тізімі өсу ретімен кезекке тұрғызамыз: 1 – 3 – 11;

– кезектен төбені таңдаймыз (осы мысалды ол 1-төбе), оны қарап өтеміз, «жаңа емес» деп белгілейміз. Қаралып өткен тізіммен байланысы бар сыбайлас төбелердің бүкіл тізімі өсу ретімен кезекке тұрғызамыз: 3 – 11;

– процесс графта төбелер бар болғанша қайталанады.

Графтың қаралған төбелерінің:

0 – 1 – 3 – 11 – 4 – 6 – 9 – 10 – 5 – 8 – 12 – 2 – 7.

13.1-суретінде көрсетілген граф мысалында Графты «тереңдігі» бойынша жүріп өту алгоритмінің бағдарламалақ орындалуын қарастырайық.

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int i, j, k, n,kol;
 static void vkl(Stack vst, int n)
 {
  vst.Push(n);
 }
 static void iskl(Stack vst)
 {
  if (vst == null) Console.WriteLine("Stek bos!");
  else
  n = (int)vst.Pop();
 }
 public static void Main()
 {
  Stack vstek = new Stack();
  //int i, j, k, n;
  bool[] nov = new bool[13];
  int[,] p = new int[13, 13];
int[,] a = new int[13, 13]
{{0,1,1000,1,1000,1000,1000,1000,1000,1000,1000, 1, 1000 },
```

```
{1,0,1000,1,1000,1000,1000,1000,1000,1000,1000,1000, 1000},
{1000,1000,0,1000,1000,1000,1,1000,1000,1000,1000,1000,1000},
{1,1,1000,0,1,1000,1,1000,1000,1000,1000,1,1000},
{1000,1000,1000,1,0,1,1,1000,1,1000,1000,1000, 1},
{1000,1000,1000,1000,1,0,1000,1,1,1000,1000,1000,1000},
{1000,1000,1,1,1,1000, 0,1000,1000,1000,1000,1000,1000},
{1000,1000,1000,1000,1000,1,1000,0,1,1000,1000,1000,1000},
{1000,1000,1000,1000,1,1,1000,1,0,1000,1000,1000,1000},
{1000,1000,1000,1000,1000,1000,1000,1000,1000,0,1,1,1000},
{1000,1000,1000,1000,1000,1000,1000,1000,1000,1,0,1,1000},
{1,1000,1000,1,1000,1000,1000,1000,1000,1,1,0,1000},
{1000,1000,1000,1000,1,1000,1000,1000,1000,1000,1000,1000,0}, };
for (i = 0; i < 13; i++)
{
p[i,0] = i;k=1;
for (j = 0; j < 13; j++)
if ((a[i, j] != 1000) && (a[i, j] != 0))
{
 p[i,k]=j;
 k++;
}
p[i,k]=1000;
}
for (i = 0; i < 13; i++)
{
k = 0;
while (p[i,k] != 1000)
{
 Console.Write(" {0}", p[i,k]);
 k++;
}
Console.WriteLine();
}
// графты жүріп өту
bool b;
// Бастапқы шарттарды беру
for (i = 0; i < 13; i++) nov[i] = true;
vkl(vstek,p[0,0]);
kol = 1;
Console.Write(" {0}", p[0, 0]);
nov[0] = false;
// графты «тереңдігі» бойынша жүріп өту циклі
while (kol != 0)
{ i = (int)vstek.Peek();
if (p[i,0] == 1000) b = false;
else b = !nov[p[i,0]];
```

```
 // графтың жаңа төбесін іздеу
 k = 0;
 while (b == true)
 {
  k++;
  if (p[i,k] == 1000) b = false;
  else
 {
 b = !nov[p[i,k]];
 if (nov[p[i, k]]) { vkl(vstek, p[i, k]); kol++; }
 }
 }
 if (p[i,k] != 1000)
 // егер графтың жаңа төбесі табылса
 {
  i = p[i,k];
  Console.Write(" {0}", i);
  nov[i] = false;
 }
 else
// тізімде жаңа төбе жоқ болса, алдынғы төбеге оралу керек
 {
  iskl(vstek); i = n; kol--;
 }
 }
 Console.WriteLine();
 Console.WriteLine("Enter pernesin basiniz ");
 Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
 0  1  3  11
 1  0  3
 2  6
 3  0  1  4  6  11
 4  3  5  6  8  12
 5  4  7  8
 6  2  3  4
 7  5  8
 8  4  5  7
 9  10  11
 10  9  11
 11  0  3  9  10
 12  4
 0  1  3  4  5  7  8  6  2  12  11  9  10
```

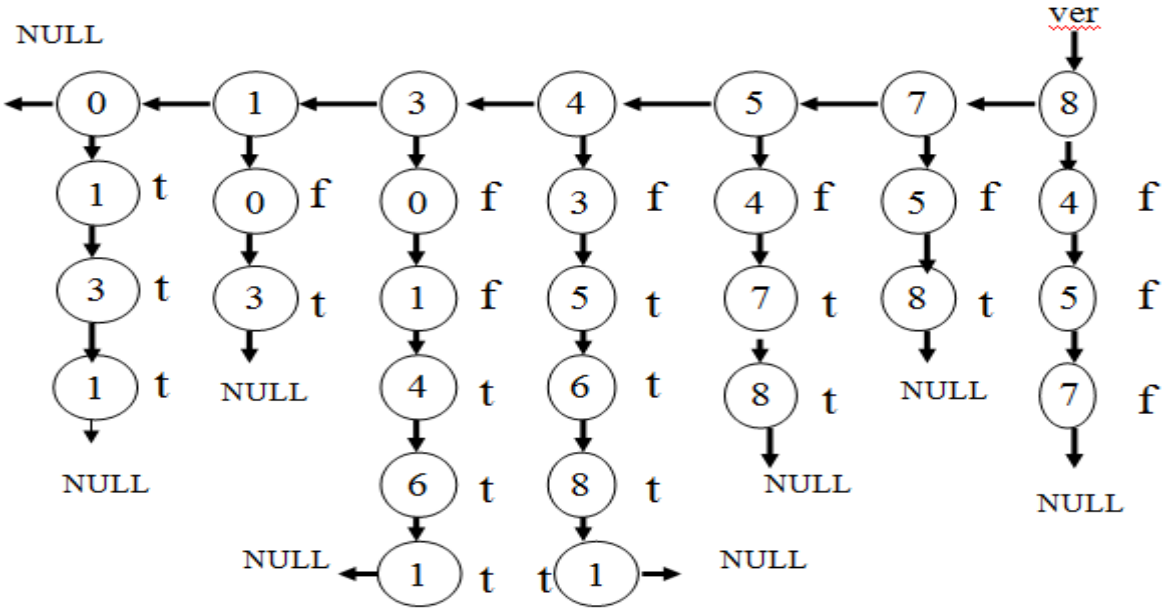
```
Enter pernesin basiniz
```

Бағдарлама басында p[13,13] матрицасы құрылады, онда барлық сыбайлас төбелер тізімі жазылады. Осы тізімдердің мәндерін бақылау үшін олар манитор экранынан шығарылады.

Бағдарламада стек қолданылған, онда біріншіден графтың алғашқы төбе тізімінің тақырыбы жазылады. Графты «тереңдігі» бойынша жүріп өту циклі жұмысының барысында стекке графтың ең кіші жаңа сыбайлас төбелерінің тізім тақырыптары жазылады, олар тақырыбы стек төбесінде орналасқан тізімнен таңдап алынады. Егер осы тізімде жаңа сыбайлас төбе жоқ болса, онда сәйкес тізімнің тақырыбы стектен жойылады, стек төбесі алдыңғы тізімге көшеді.

Графтың сыбайлас төбелер тізімдерінің тақырыптары стекте жоқ болса алгоритм жұмысы аяқталады.

Стектің графты «тереңдігі» бойынша жүріп өту жұмысының бастапқы үзіндісі 13.2-суретінде көрсетілген.



13.2-сурет – Графты «тереңдігі» бойынша жүріп өтудегі стектің жұмысы

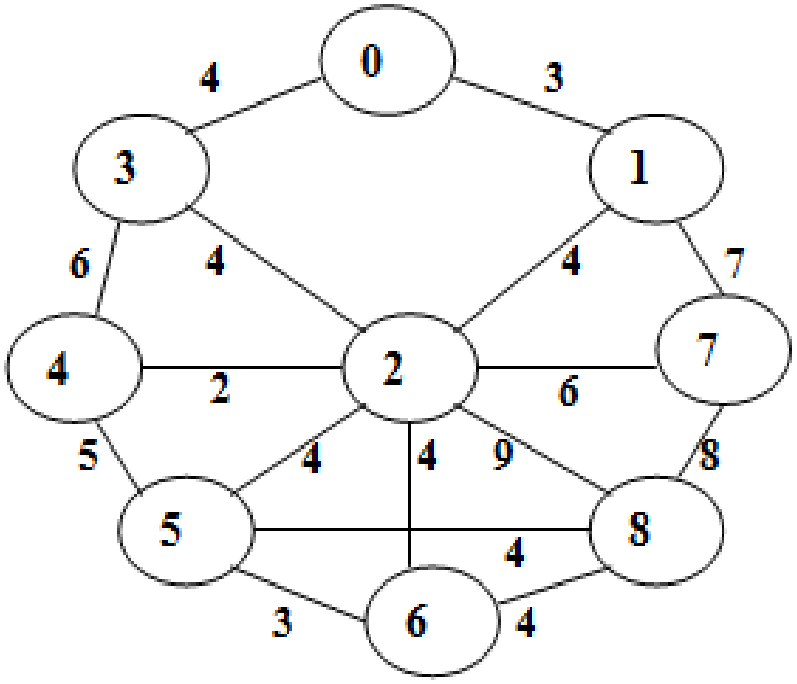
13.2-суретте графтың сыбайлас төбелерінің бірінші тізімінен 1-төбе, екінші тізімінен 3-төбе, үшінші тізімінен - 4-төбе, т.б. таңдап алынғаны көрсетілген. Стектің толуына қарай таңдап алынған төбелердің мәндері true (t) мәнінен false (f) мәніне ауысады. 8-төбені қарап шыққаннан кейін стектен 8, 7, 5 төбелері (оларда жаңа төбелер жоқ) жойылады. 4-төбенің тізімінде жаңа 6-шы нөмерлі төбе бар, оның тізімінің тақырыбы стекке қосылады, т.б.

13.2 Берілген екі төбе арасындағы барлық маршруттарды іздеу алгоритмі

Берілген екі төбе арасындағы барлық маршруттарды іздеу алгоритмі графты «тереңдігі» бойынша жүріп өтуге негізделген. Бірінші айырмашылық – стекті граф төбелерінің нөмірлерін ғана емес, сонымен қатар сыбайлас төбелер тізімін және тізімдегі төбе позициясын сақтау үшін қолдану.

Екінші айырмашылық – граф төбелеріне «жаңа» мәндерді қайтару. Егер граф төбесінің тізімі аяқталса, яғни «жаңа» төбелер жоқ болса, онда стектен төбе жойылады және осы төбені басқа маршруттарда қолдану үшін оған «жаңа»деген мән қайтарылады.

13.1-есеп. Сыбайлас матрицамен (13.1-кесте) берілген 13.3-суреттегі граф үшін диалог режімінде берілетін екі төбе арасындағы барлық маршрутты табатын бағдарламаны құру керек.



13.3-сурет – Бағытталған граф

13.1-кесте – 13.3-суреттегі графтың сыбайлас матрицасы

	0	1	2	3	4	5	6	7	8
0	0	3	1000	4	1000	1000	1000	1000	1000
1	3	0	4	1000	1000	1000	1000	7	1000
2	1000	4	0	5	2	4	4	6	9
3	4	1000	5	0	6	1000	1000	1000	1000
4	1000	1000	2	6	0	5	1000	1000	1000
5	1000	1000	4	1000	5	0	3	1000	4
6	1000	1000	4	1000	1000	3	0	1000	4
7	1000	7	6	1000	1000	1000	1000	0	8
8	1000	1000	9	1000	1000	4	4	8	0

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
```

```
namespace ConsoleApplication1
{
 class Program
 {
 public static int i, j, k, kol;
 public struct uzel
 {
  public int nom;
  public int ki;
  public int kj;
 };
 public static uzel n = new uzel();
 static void vkl(Stack vst, uzel n)
 {
  vst.Push(n);
 }
 static void iskl(Stack vst)
 {
  if (vst == null) Console.WriteLine("Стек бос !");
  else n = (uzel)vst.Pop();
 }
 public static void Main()
 {
  Stack vstek = new Stack();
  string buf;
  int beg, en, x, i1, i2;
  uzel y1 = new uzel();
  bool[] nov = new bool[10];
  int[,] p = new int[9, 9];
  int[] m = new int[10];
  //графтың сыбайлас матрицасы
  int[,] a = new int[9, 9]
 {{ 0, 6,1000, 4,1000,1000,1000,1000,1000},
 { 6, 0, 5,1000,1000,1000,1000, 4,1000},
 {1000, 5, 0, 3, 4,1000, 5, 6,1000},
 { 4,1000, 3, 0, 7,1000,1000,1000,1000},
 {1000,1000, 4, 7, 0, 5,1000,1000,1000},
 {1000,1000,1000,1000, 5, 0, 4,1000, 9},
 {1000,1000, 5,1000,1000, 4, 0,1000, 6},
 {1000, 4, 6,1000,1000,1000,1000, 0, 8},
 {1000,1000,1000,1000,1000, 9, 6, 8, 0}};
  //a матрицасы бойынша сыбайлас төбелер тізімін құру
  for (i = 0; i < 9; i++)
  {
  p[i, 0] = i; k = 1;
  for (j = 0; j < 9; j++)
```

```
   if ((a[i, j] != 1000) && (a[i, j] != 0))
   {
   p[i, k] = j;
   k++;
   }
  p[i, k] = 1000;
  }
  //сыбайлас төбелер тізімін экранға шығару
  for (i = 0; i < 9; i++)
  {
  k = 0;
  while (p[i, k] != 1000)
  {
   Console.Write(" {0}", p[i, k]);
   k++;
  }
  Console.WriteLine();
  }
  Console.Write("Graftin bastapki tobesin engiziniz? ");
  buf = Console.ReadLine();
  beg = Convert.ToInt32(buf);
  Console.Write("Graftin akirgi tobesin engiziniz? ");
  buf = Console.ReadLine();
  en = Convert.ToInt32(buf);
  for (i = 0; i < 9; i++)
  {
  nov[i] = true;
  m[i] = 0;
  }
  nov[9] = false;
  x = 1;  // кезекті маршруттың нөмері
  m[1] = beg; i1 = 2;
  y1.nom = beg; //y1 түйіннінің nom өрісін анықтаймыз
  y1.ki = beg; //сыбайлас төбелер тізімінің нөмерін
сақтаймыз
  y1.kj = 0; // тізімдегі ағымдағы орнын сақтаймыз
  vkl(vstek, y1);
  kol = 1; //стектегі элементтер саны
  nov[beg] = false;
  nov[en] = false;
  i = beg; j = 0;
  // графтағы барлық маршруттарды іздеу циклі
  while (kol != 0)
  {
  do
  {
```

```
   j++;
   if (p[i, j] == en)
   {
   Console.Write(" Жол - {0} -", x); x++;
   for (i2 = 1; i2 < i1; i2++)
    Console.Write(" {0}", m[i2]);
   Console.WriteLine(" {0}", en);
   }
  }
  while ((p[i, j] != 1000) && (!nov[p[i, j]]));
  if (p[i, j] != 1000)
   if (nov[p[i, j]])
   {
   y1.ki = i;
   y1.kj = j;
   i = p[i, j];
   y1.nom = i;
   vkl(vstek, y1);
   j = 0;
   kol++;
   nov[i] = false;
   m[i1] = i;
   i1++;
   };

  if (p[i, j] == 1000)
  {
   kol--;
   if (kol != 0)
   {
   iskl(vstek);
   i = n.ki;
   j = n.kj;
   i1--;
   m[i1] = 0;
   nov[n.nom] = true;
   }
  };
  }
  Console.WriteLine();
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

0 1 3
1 0 2 7
2 1 3 4 6 7
3 0 2 4
4 2 3 5
5 4 6 8
6 2 5 8
7 1 2 8
8 5 6 7
Graftin bastapki tobesin engiziniz? 1
Graftin akirgi tobesin engiziniz? 4
Жол - 1 - 1 0 3 2 4
Жол - 2 - 1 0 3 2 6 5 4
Жол - 3 - 1 0 3 2 6 8 5 4
Жол - 4 - 1 0 3 2 7 8 5 4
Жол - 5 - 1 0 3 2 7 8 6 5 4
Жол - 6 - 1 0 3 4
Жол - 7 - 1 2 3 4
Жол - 8 - 1 2 4
Жол - 9 - 1 2 6 5 4
Жол - 10 - 1 2 6 8 5 4
Жол - 11 - 1 2 7 8 5 4
Жол - 12 - 1 2 7 8 6 5 4
Жол - 13 - 1 7 2 3 4
Жол - 14 - 1 7 2 4
Жол - 15 - 1 7 2 6 5 4
Жол - 16 - 1 7 2 6 8 5 4
Жол - 17 - 1 7 8 5 4
Жол - 18 - 1 7 8 5 6 2 3 4
Жол - 19 - 1 7 8 5 6 2 4
Жол - 20 - 1 7 8 6 2 3 4
Жол - 21 - 1 7 8 6 2 4
Жол - 22 - 1 7 8 6 5 4

Enter pernesin basiniz

Егер екі бірдей төбелер берілсе, онда графтың барлық циклдарын табуға болады, олар берілген төбеде басталып, осы төбеде аяқталады. Мысалы:

0 1 3
1 0 2 7
2 1 3 4 6 7
3 0 2 4
4 2 3 5
5 4 6 8
6 2 5 8
7 1 2 8

8 5 6 7
Graftin bastapki tobesin engiziniz? 6
Graftin akirgi tobesin engiziniz? 6
Жол - 1 - 6 2 1 0 3 4 5 6
Жол - 2 - 6 2 1 0 3 4 5 8 6
Жол - 3 - 6 2 1 7 8 5 6
Жол - 4 - 6 2 1 7 8 6
Жол - 5 - 6 2 3 0 1 7 8 5 6
Жол - 6 - 6 2 3 0 1 7 8 6
Жол - 7 - 6 2 3 4 5 6
Жол - 8 - 6 2 3 4 5 8 6
Жол - 9 - 6 2 4 3 0 1 7 8 5 6
Жол - 10 - 6 2 4 3 0 1 7 8 6
Жол - 11 - 6 2 4 5 6
Жол - 12 - 6 2 4 5 8 6
Жол - 13 - 6 2 6
Жол - 14 - 6 2 7 1 0 3 4 5 6
Жол - 15 - 6 2 7 1 0 3 4 5 8 6
Жол - 16 - 6 2 7 8 5 6
Жол - 17 - 6 2 7 8 6
Жол - 18 - 6 5 4 2 1 7 8 6
Жол - 19 - 6 5 4 2 3 0 1 7 8 6
Жол - 20 - 6 5 4 2 6
Жол - 21 - 6 5 4 2 7 8 6
Жол - 22 - 6 5 4 3 0 1 2 6
Жол - 23 - 6 5 4 3 0 1 2 7 8 6
Жол - 24 - 6 5 4 3 0 1 7 2 6
Жол - 25 - 6 5 4 3 0 1 7 8 6
Жол - 26 - 6 5 4 3 2 1 7 8 6
Жол - 27 - 6 5 4 3 2 6
Жол - 28 - 6 5 4 3 2 7 8 6
Жол - 29 - 6 5 6
Жол - 30 - 6 5 8 6
Жол - 31 - 6 5 8 7 1 0 3 2 6
Жол - 32 - 6 5 8 7 1 0 3 4 2 6
Жол - 33 - 6 5 8 7 1 2 6
Жол - 34 - 6 5 8 7 2 6
Жол - 35 - 6 8 5 4 2 6
Жол - 36 - 6 8 5 4 3 0 1 2 6
Жол - 37 - 6 8 5 4 3 0 1 7 2 6
Жол - 38 - 6 8 5 4 3 2 6
Жол - 39 - 6 8 5 6
Жол - 40 - 6 8 6
Жол - 41 - 6 8 7 1 0 3 2 4 5 6
Жол - 42 - 6 8 7 1 0 3 2 6
Жол - 43 - 6 8 7 1 0 3 4 2 6

```
Жол - 44 - 6 8 7 1 0 3 4 5 6
Жол - 45 - 6 8 7 1 2 3 4 5 6
Жол - 46 - 6 8 7 1 2 4 5 6
Жол - 47 - 6 8 7 1 2 6
Жол - 48 - 6 8 7 2 1 0 3 4 5 6
Жол - 49 - 6 8 7 2 3 4 5 6
Жол - 50 - 6 8 7 2 4 5 6
Жол - 51 - 6 8 7 2 6

Enter pernesin basiniz
```

Әдетте барлық маршруттарды іздестірудің `while` циклінде `do_while` циклі мен екі `if` операторы орналасады.

`do_while` циклінде графты «тереңдігі» бойынша жүріп өту жұмысының алгоритмінің көмегімен графтың жаңа төбесі ізделінеді. Егер кезекті төбе оның «соңғы» төбесіне тең болса, онда стек мәліметтері экранға шығады. Стекте маршрут төбелерінің тізімі бар, олар графтың «бастапқы» мен «ақырғы» төбелері арасында болады.

Егер графтың табылған төбесі «жаңа» болса, онда бірінші `if` операторында алгоритм әрекеттері сипатталады.

Егер қаралған сыбайлас төбелер тізімінде «жаңа» төбе болмаса, онда екінші `if` операторында алгоритмнің әрекеттері сипатталады.

13.3 Өзін-өзі тексеру сұрақтары

1 Өз жұмысына стекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

2 Графты жүріп өту алгоритмінің стегінде не сақталады? Алгоритм өз жұмысында стекті пайдаланады?

3 Өз жұмысында кезекті пайдаланатын графты жүріп өту алгоритмі қалай аталады?

4 Графты жүріп өту алгоритм кезегі нені сақтайды (графты жүріп өту алгоритмі кезекті пайдаланады)?

5 Графты «тереңдігі» бойынша жүріп өту алгоритмінің аяқталу шарты?

6 Графты «көлденеңі» бойынша жүріп өту алгоритмінің аяқталу шарты?

7 Графтың барлық циклдарын іздеу алгоритмінде графтың қаралып кеткен төбелеріне жаңа төбе қасиеті неге «қайтарылады»?

8 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінің стегінде не сақталады?

9 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде `do_while` циклі неге қолданылады?

10 Екі берілген төбе арасында барлық маршруттарды іздеу алгоритмінде неге екі шартты `if операторы` қолданылады?