ГРАФТАР

12.1 Графтар теориясының негізгі анықтамалары

Графтар теориясын білуді қажет ететін белгілі бір есептердің класы болады, мысалы, көлік есептері, желілерде ағындарды тиімдеу есептері, иерархиялық бұтақтар тәрізді құрылымдарда іздестестіру есептері, т.б.

Бұтақтар мен графтар бір біріне ұқсас болып келеді. Мысалы, бағдарлама саласындағы белгілі теорияшыл Д.Кнут графтарды бұтақтар бөлімінде түсіндіріп кеткен. Паскаль тілін жасаушы Н.Вирт “Алгоритмы и структуры данных” кітабында графты мүлдем қарастырмайды, бірақ күрделілігі әртүрлі бұтақтарды қарастырады. Ресей авторлары күрделі құрылымдарды сипаттағанда біріншіден графтарды қарастырады (12.1-суретті қара).

Сонымен қатар граф қос (V,E) жиынтық ретінде анықталады.

мұнда V – төбелердің шекті жиыны;

E – төбелерді байланыстыратын қабырғалардың шекті жиыны.

S=<U,W> жазбасы мынаны сипаттайды: S қабырғасы U және W төбелерін байланыстырады.

Бір қабырғамен байланысатын төбелер сыбайлас төбелер деп аталады.

S қабырғасы және U төбесі (және S қабырғасы мен W төбесі) түйісті (инцидентті) деп аталады.

Бір төбеге түйісті (инцидентті) қабырғалар сыбайлас қабырғалар деп аталады.

Төбенің дәрежесі оған түйісті (инцидентті) қабырғалардың санына тең.

U және K төбелерін байланыстыратын жол – $W_0,W_1, \ldots, W_n$ (n>0), төбелерінің тізбегі, мұнда W_0=U, W_n=K, кез келген i ($0 \le i \le n-1$) үшін W_i және W_{i+1} төбелері қабырғалармен байланысқан.

$W_0, \ldots, W_n$ жолының ұзындығы оның қабырғаларының санына – n тең.

Егер жолдың барлық төбелері әртүрлі болса, онда ондай жол қарапайым жол деп аталады.

Егер $W_0=W_n$ болса, онда ондай жол тұйық жол деп аталады.

Барлық қабырғалары әр түрлі тұйық жолды цикл деп атайды.

Барлық төбелері әр түрлі тұйық жолды контур деп атайды.

Екі төбе арасындағы қашықтық – осы төбелерді байланыстыратын ең қысқа жолдың ұзындығы.

Егер кез келген төбелер жұбын байланыстыратын жол бар болса, онда ондай граф байланысты граф деп аталады.

Егер кез келген екі төбелер қабырғалар шекті байланыстырылса, онда ондай граф толық граф деп аталады.

Бағытталған граф немесе орграфтың қарапайым графтан айырмашылығы - төбелер, қабырғалармен емес, доғалармен (доға дегеніміз - бағыты берілген қабырға) байланысуы.

Орграф бойымен орын ауыстыруды тек доға бағытымен ғана орындауға болады.

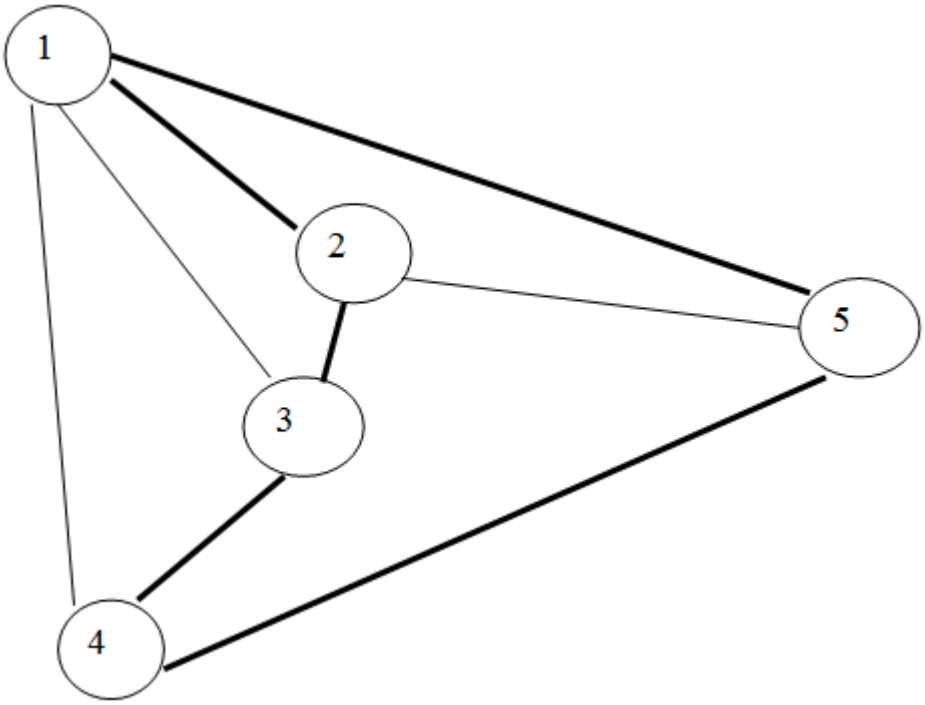
Графтың кез келген төбесінен белгілі төбеге дейін жол бар болса, онда осы төбе орграфтың құйылысы деп аталады.

Бір төбеден графтың кез келген төбесіне дейін жол бар болса, онда ондай төбені орграфтың көзі деп атайды.

U төбесінің графтың қалған төбелеріне дейінгі ең үлкен қашықтық U төбесінің эксцентриситеті деп аталады.

Төбелердің арасындағы ең үлкен эксцентриситетті графтың диаметрі деп атайды. Төбелердің арасындағы ең кіші эксцентриситетті графтың радиусы деп атайды.

Эксцентриситеті графтың радиусына тең төбені графтың медианасы немесе оның центрлік төбесі деп атайды.



12.1-суреті – Графтың көрінісі

Егер барлық төбелер (бірақ барлық қабырғалардың кіруі міндетті емес) циклге кіретін болса және осы цикл графта бар болатын болса, онда осы граф гамильтонды граф деп аталады.

Егер графтың барлық қабырғалары циклге бір рет қана кіретін болса және осы цикл графта бар болатын болса, онда осы граф Эйлер графы деп аталады.

Графтың барлық қабырғаларының бойымен тек бір рет қана жүріп өтетін жолды Эйлер жолы деп атайды.

Графты сурет түріндегі қабылданған форма бойынша көрсетуге болады, онда төбелерге шеңберлер, ал қабырғаларға осы шеңберлерді байланыстыратын сызықтар сәйкес келеді.

12.1-суретінде Гамильт циклі жалпақ сызықпен белгіленген.

Бұтақтар тәрізді құрылымның көптеген анықтамалары бар.

Байланысқан граф арқылы бұтақтардың екі анықтамасын келтірейік.

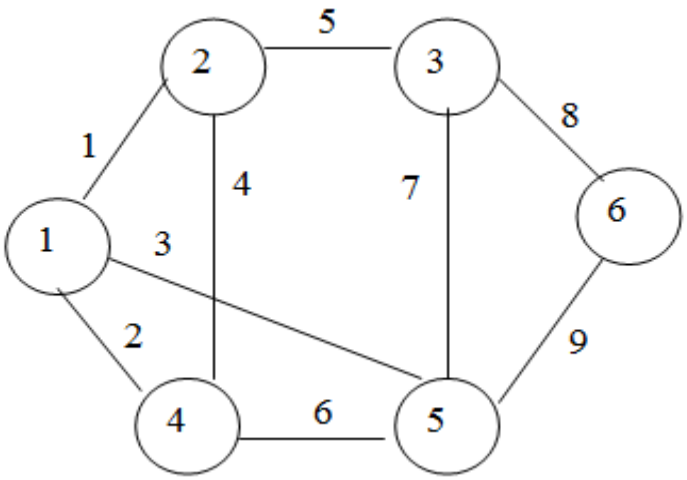
Бұтақтар дегеніміз – доғаларының саны төбелерінің санынан бірге кем болатын байланыстырылған граф.

Бұтақтар дегеніміз – циклі жоқ байланыстырылған граф.

«Графтық» есептерді шығару барысында алгоритмнің күрделі болуы оның «машиналық» көрсетіміне байланысты, яғни «графтық» есептің типіне байланысты ЭЕМ жадысында графты көрсетудің белгілі бір формасы қолданылады.

12.2 Компьютер жадысында графты көретудің тәсілдері

12.2.1 Графты түйістілік матрицасы (матрица инцидентностей) түрінде көрсету. 12.2-суретте көрсетілген қарапайым бағытталмаған граф берілсін.



12.2-сурет – Қарапайым бағытталмаған граф

Графтар теориясы бойынша ЭЕМ жадысында графты көрсетудің классикалық әдісі ретінде түйістілік матрицасы қолданылады. Осындай матрицада жолдардың саны төбелер санына, ал бағаналар саны граф қабырғаларының санына сәйкес болады. Егер төбе мен қабырға түйістілі болса, онда бағана мен жол қиылысында 1 жазылады, әйтпесе 0 жазылады.

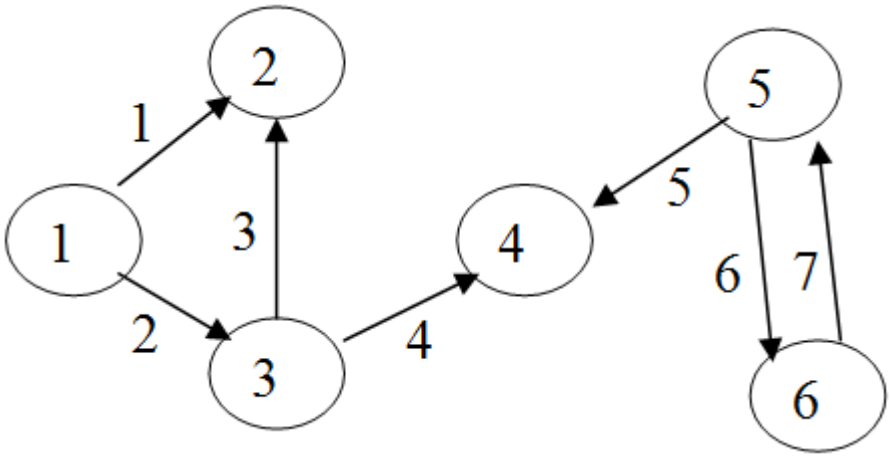
12.2-суретте көрсетілген қарапайым бағытталмаған граф үшін 12.1-кестесінде көрсетілген түйістілік матрицасын құруға болады.

12.1-кесте – Түйістілік матрицасы

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

Егер доға сәйкес төбеден «шықса», онда бағытталған графта түйістілік матрицасындағы доға бағыты 1-ге тең, егер доға сәйкес төбеге «кіретін» болса, онда минус 1-ге тең.

12.3-суретте көрсетілген бағытталған граф үшін түйістілік матрицасын қарастырайық



12.3-сурет – Бағытталған граф

Бағытталған графтың түйістілік матрицасы 12.2-кестеде көрсетілген.

12.2-кесте – Бағытталған графтың түйістілік матрицасы

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

Айта кететін жәйт, түйістілік матрицалары ЭЕМ жадысында графтарды сақтаудың қолайсыз тәсілдердің бірі болып есептеледі, өйткені оның мәндері анық мәліметті бермейді. Мысалы, матрицаның өзінде 1-ші мен 2-ші төбелерінің байланысқанын анықтау мүмкін емес.

12.2.2 Графты сыбайлас матрица арқылы көрсету. Сыбайлас матрицада жолдар мен бағаналар саны төбелер санына тең.

Қиылыста осы төбелердің байланысын сипаттайтын мән орналасады, мысалы, егер қабырға болса, онда 1-ге, егер қабырға жоқ болса, онда 0-ге тең.

Бағытталған графтар үшін сыбайлас матрицада жолдар мен бағаналар саны төбелер санына тең.

Қиылыста осы төбелердің байланысын сипаттайтын мән орналасады, мысалы, егер сәйкес төбелер арасында байланыс бар болса, онда 1-ге, егер байланыс жоқ болса, онда 0-ге тең.

12.2-суретте көрсетілген граф үшін сыбайлас матрица 12.3-кестеде көрсетілген.

12.3-кесте – Сыбайлас матрица

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	1	0	0
3	0	1	0	0	1	1
4	1	1	0	0	1	0
5	1	0	1	1	0	1
6	0	0	1	0	1	0

12.3-суретте көрсетілген бағытталған граф үшін сыбайлас матрица 12.4-кестеге сәйкес келеді.

12.4-кесте – Бағытталған графтың сыбайлас матрицасы

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	0	0
5	0	0	0	1	0	1
6	0	0	0	0	1	0

12.2.3 Графты жұптар тізімі түрінде көрсету. Графты төбелердің сыбайлас жұптарының тізімі түрінде көрсету. Мысалы, жоғарыда ұсынылған графтарды төбелердің сыбайлас жұптарының тізімімен көрсетуге болады.

1-граф	2-граф
1) 1 – 2	1 – 2
2) 1 – 4	1 – 3
3) 1 – 5	3 – 2
4) 2 – 3	3 – 4
5) 2 – 4	5 – 4
6) 3 – 5	5 – 6
7) 3 – 6	6 – 5
8) 4 – 5	1– 2
9) 5 – 6	1– 2

Осы тізімдерді ЭЕМ жадында екі өлшемді массив түрінде көрсетуге болады. ЭЕМ жадында графтарды осы жолмен сақтау ең тиімді болып есептеледі, бірақ жұмыс жасауға үнемі қолайлы бола бермейді.

12.2.4 Сыбайлас төбелердің тізімі түрінде графты көрсету. Графтарға арналған кейбір есептерде граф төбелерін сыбайлас төбелердің тізімдік құрылымы түрінде көрсетуді талап етеді. Мысалы, стек, кезек, қарапайым тізім. Осы жағдайда әдетте сыбайлас төбелер тізімдерінің тақырыптары бойынша массив құрылады. Мысалы, 12.2-суретте көрсетілген графты тақырыптар массиві, графтың көршілес төбелерінің тізімі түрінде, төменде көрсетілгендей жазуға болады:

M[1]->1->2->4->5->NULL;
M[2]->2->1->3->4->NULL;
M[3]->3->2->5->6->NULL;
M[4]->4->1->2->5->NULL;
M[5]->5->3->4->6->NULL;
M[6]->6->3->5->NULL;

12.3-суретте көрсетілген графты тақырыптар массиві, графтың көршілес төбелерінің тізімі түрінде, төменде көрсетілгендей жазуға болады:

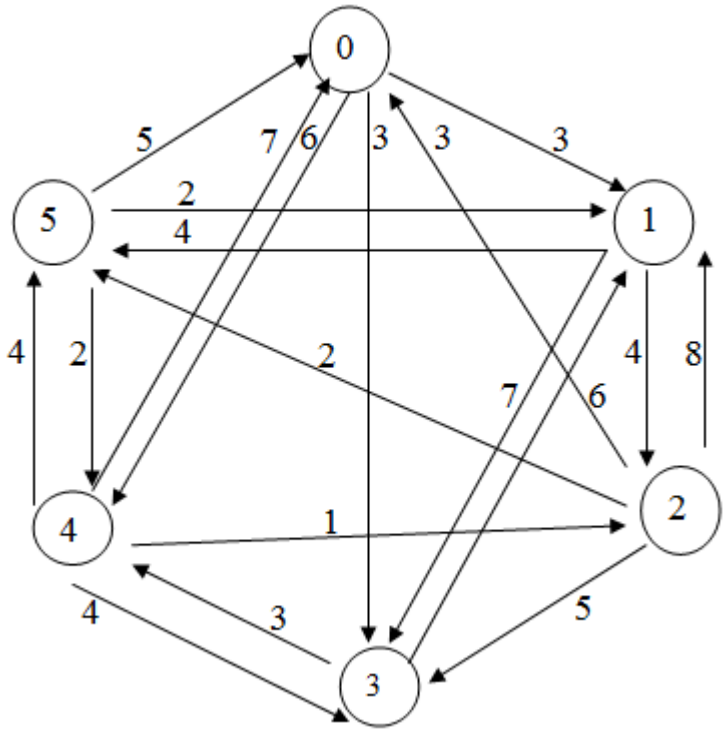
M[1]->1->2->3->NULL;
M[2]->2->NULL;
M[3]->3->2->4->NULL;
M[4]->4->NULL;
M[5]->5->4->6->NULL;
M[6]->6->5->NULL;

Графты көрсетудің осы формасы түрлі графты жүріп өту алгоритмдерін орындау үшін қолайлы.

12.3 Флойд алгоритмі

Көлік туралы есептердің көптеген түрлері бар, есепте графтың әр түрлі төбелерінің арасындағы арақашықтықты анықтау немесе оны қолдану керек және осы арақашықтық ең қысқа арақашықтық болуы керек.

Графтар теориясында граф төбелерінің арасындағы ең қысқа арақашықтықты анықтайтын көптеген алгоритмдер белгілі, бірақ оның барлығы Флойд ұсынған алгоритмді негізге алады. Осы алгоритмнің кең қолданылуы төбелер арасындағы ең қысқа арақашықтықты табу идеясында, бірақ есептеу бойынша Флойд алгоритмінің тиімділігі өте төмен. Флойд алгоритмін қарастырайық. 6 төбесі бар бағытталған граф бар болсын, 12.4-суретте көрсетілген.



12.4-сурет – Бағытталған граф

Осы графтың сыбайлас матрицасы 12.5-кестесінде көрсетілген:

12.5-кесте –12.4-суреттегі графтың сыбайлас матрицасы

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

Флойд алгоритмінің идеясы бойынша әрбір `a[i,j]` – төбелер жұбына `k (0 – 5)` төбелерінің барлық мүмкін нұсқалары мына түрде тексеріледі: `a[i,j]>a[i,k]+a[k,j].`

Егер `a[i,j]` жолы үлкен болса, онда сыбайлас матрицада `i` және `j` төбелерінің арасындағы жолдың мәні `a[i,k]+a[k,j]` жолына тең жаңа мәнмен ауыстырылады. Сонымен, ең қысқа қашықтықты анықтайтын матрица дайын болды.

Мысалы, сыбайлас матрицада 2-шіден 1-г дейін жол 8-ге тең. Бірақ, егер біз бірінші 0-ге, ал содан кейін 1-ге орын ауыстыратын болсақ, онда жол 6-ға тең болады: `a[2,0]+a[0,1]= 3+3=6`. Барлық нұсқаларды тексере отырып, біз мынаны байқадық: `2-5-1 жолы қысқарық, a[2,5]+a[5,1]=2+2=4`.

Сонымен, 2-ші мен 1-ші төбелердің арасындағы ең қысқа жол табылды.

Бағдарлама коды:

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

```
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int[,] a = new int[6, 6]
  {{0,3,1000,3,6,1000},
  {1000,0,4,7,1000,4},
    {3,8,0,5,1000,2},
  {1000,6,1000,0,3,1000},
    {7,1000,1,4,0,4},
  {5,2,1000,1000,2,0}};
  int i, j, w, n, k;
  Console.WriteLine("Matrizani shigary: ");
  for (i = 0; i < 6; i++)
  {
  for (j = 0; j < 6; j++)
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.WriteLine();
  for (k = 0; k < 6; k++)
  for (i = 0; i < 6; i++)
   for (j = 0; j < 6; j++)
   if (a[i, j] > a[i, k] + a[k, j])
    a[i, j] = a[i, k] + a[k, j];
  Console.WriteLine("Zhana matrizani shigary: ");
  for (i = 0; i < 6; i++)
  {
  for (j = 0; j < 6; j++)
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысының нәтижесі:

```
Matrizani shigary:
 0 3 1000 3 6 1000
 1000 0 4 7 1000 4
 3 8 0 5 1000 2
 1000 6 1000 0 3 1000
 7 1000 1 4 0 4
 5 2 1000 1000 2 0
```

```
Zhana matrizani shigary:
 0  3  7  3  6  7
 7  0  4  7  6  4
 3  4  0  5  4  2
 7  6  4  0  3  6
 4  5  1  4  0  3
 5  2  3  6  2  0
```

Бағдарламаның (алгоритмнің) кемшіліктері:

– ең қысқы арақышықтықтың маршруты анықталмайды;

– алгоритмнің есептеу тиімділігі n-нің 3-ші дәрежесіне пропорционал (мұнда n – граф төбелерінің саны), ол n-нің үлкен мәндерінде тиімсіз болып келеді.

12.4 Дейкстр алгоритмі

Көптеген авторлардың пікірінше Дейкстр алгоритмі графта берілген екі төбе арасындағы ең қысқа маршрутты табу үшін ең тиімді алгоритм болып есептеледі.

Бағытталмаған граф берілген болсын, ол 12.5-суретінде көрсетілген. Дейкстр алгоритмінде үш массивті қолдану керек, олардың өлшемі граф төбелерінің санына сәйкес болады.

Бірінші массив - «тұрақты» төбелер массиві графтың берілген төбесінен барлық қалған төбелеріне дейінгі ең қысқа маршруттарды сақтайды. Массивке бірінші болып бірінші төбенің нөмері жазылады (осы төбелер арқылы графтың басқа төбелеріне үшін ең қысқа ара қашықтық анықталады). Массивтің атауы таңдап алынған төбелердің атауларына сәйкес болады. Дейкстр алгоритмі бойынша графтың барлық төбелері «уақытша» болып жарияланады, ал таңдап алынған төбелер «тұрақты» деп аталады. Бірінші төбе «тұрақты» болып жарияланады.

Екінші массив графтың берілген төбесінен барлық қалған төбелеріне дейінгі ең қысқа арақашықтықты сақтайды. Бірінші болып оған таңдап алынған төбе нөмеріне сәйкес сыбайлас матрицасының жолы көшіріліп жазылады. Графтың сыбайлас матрицасы 12.6-кестеде көрсетілген.

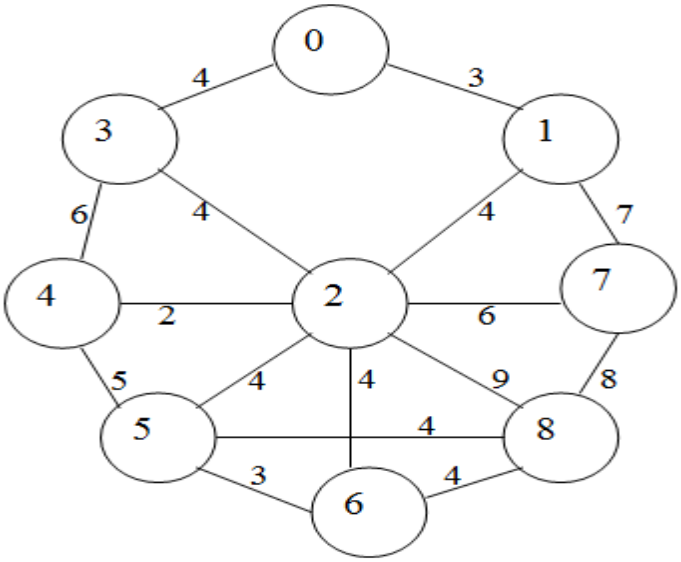
Үшінші логикалық типтегі массивте графтың таңдап алынған_(тұрақты) төбелер жазылады. Алғашында графтың «тұрақты» төбесі ретінде бірінші төбе ғана бола алады.

Одан кейін «уақытша» төбе таңдалады, бірінші (тұрақты) төбеден осы төбеге дейінгі аралық ең қысқа болуы керек. Осы төбе `k` төбесі болып жарияланады.

Графтың «тұрақты» төбесінен барлық «уақытша» төбеге дейінгі барлық маршруттар тікелей және `k` төбесі арқылы тексеріледі. ең қысқа аралықтар екінші массивке жазылады. Ал, егер ең қысқа арақашықтықтың маршруты `k` төбесі арқылы өтетін болса, онда бірінші массивке сәйкес позицияда k жазылады.

12.6-кесте – 12.5-суреттегі графтың сыбайлас матрицасы

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



12.5–сурет – Қарапайым бағытталмаған граф

Ары қарай `k` төбесі «тұрақты» болып жарияланады (ол үшінші массивте орындалады). Графтың келесі төбесін іздеу алгоритмі жаңа «тұрақты» төбеге қатысты қайталанады.

Бірінші төбенің нөмері 0-ге тең деп жорамалдайық. Онда `post[…]`–бірінші массив нөлдерден тұрады. Екінші массивте, яғни ең қысқа қышықтықтар массивінде сыбайлас матрицаның нөлінші жолының мәндері жазылады:

0 1 2 3 4 5 6 7 8

d[min] – 0 3 1000 4 1000 1000 1000 1000 1000

Үшінші масссив, яғни таңдап алынған (выбранный) төбелер массивінің мәндері `true` болады, тек нөлінші элементте ғана `t[0]=false`;.

0-ші төбеден басқа бір төбеге дейінгі ең қысқа аралықты таңдаймыз. Біздің жағдайымызда ол 1-ші төбе болады, оған дейінгі қашықтық 3-ке тең болады. Таңдап алынған төбе `k` деп аталсын. Флойд алгоритмін қолданып, алғашқы төбеден “`k`” төбесі арқылы өтетін қалған басқа төбелерге дейінгі барлық ең қысқа маршруттарды табамыз.

```
for (j=0;j<9;j++)
 if ((t[j]==true)&&(d[j]>d[k]+a[k,j]))
```

```
      {
  d[j]=d[k]+a[k,j];
  post[j]=k;
   }
```

нәтижесінде d және post массивтерінің жаңа мәнін аламыз:

```
         0 1 2 3 4 5 6 7 8
     d[min] -    0 3 7 4 1000 1000 1000 10 1000
0 1 2 3 4 5 6 7 8
    post[…] - 0 0 1 0 0 0 0 1 0
```

Бұл 0-2 төбелерінің арасындағы ең қысқа арақашықтық 7-ге тең болатынын білдіреді.

Таңдап алынған `k` төбесін `t[k]=false` «тұрақтысымен» жариялаймыз және осы төбеге қатысты процесс қайталанып отырады.

Бағдарлама коды:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int[,] a = new int[9, 9]
 {{ 0, 3,1000, 4,1000,1000,1000,1000,1000},
 { 3, 0, 4,1000,1000,1000,1000, 7,1000},
 {1000, 4, 0, 5, 2, 4, 4, 6, 9},
 { 4,1000, 5, 0, 6,1000,1000,1000,1000},
 {1000,1000, 2, 6, 0, 5,1000,1000,1000},
 {1000,1000, 4,1000, 5, 0, 3,1000, 4},
 {1000,1000, 4,1000,1000, 3, 0,1000, 4},
 {1000, 7, 6,1000,1000,1000,1000, 0, 8},
 {1000,1000, 9,1000,1000, 4, 4, 8, 0}};
 public static int[] d = new int[10];
 public static int[] post = new int[10];
 public static bool[] t = new bool[10];
 public static int i, j, p, k, minras;
 public static void poisk()
 {
  //алғышарттар
  minras = 0;
  for (i = 0; i < 9; i++)
  {
  post[i] = 0;
  t[i] = true;
  d[i] = a[0, i];
  }
```

```
  t[0] = false;
  post[0] = 0;
  for (i = 0; i < 8; i++)
  {
  // k төбесін іздеу
  minras = 1000;
  for (j = 0; j < 9; j++)
   if ((t[j] == true) && (minras > d[j]))
   {
   minras = d[j]; k = j;
   }
  // k төбесі аркылы маршруттарды іздеу және минимальді
қашықтықты табу
  t[k] = false;
  for (j = 0; j < 9; j++)
   if ((t[j] == true) && (d[j] > d[k] + a[k, j]))
   {
   d[j] = d[k] + a[k, j];
   post[j] = k;
   }
  }
 }
 public static void print()
 {
  // сыбайлас матрицаны шығару
  Console.WriteLine("Sibailas matrisani shigary: ");
  for (i = 0; i < 9; i++)
  {
  for (j = 0; j < 9; j++)
   Console.Write("\t" + a[i, j]);
  Console.WriteLine();
  }
  Console.WriteLine();
  // минимальді қашықтықтарды сақтайтын массивті шығару
  for (i = 0; i < 9; i++) Console.Write("\t" + i);
  Console.WriteLine();
  for (i = 0; i < 9; i++) Console.Write("\t" + d[i]);
  Console.WriteLine();
  //маршруттар массивін шығару
  for (i = 0; i < 9; i++) Console.Write("\t" + post[i]);
  Console.WriteLine();
  // графтың 0-і төбесінен бастап 8-і төбесіне дейін
маршрутты шығару
  p = 8;

  Console.Write("Songi tobe = 8 ");
```

```
  do
  {
  p = post[p];
  Console.Write("\t" + p);
  }
  while (p != 0);
  Console.WriteLine();
  Console.WriteLine("Bastapki tobe = 0 ");
 }
 static void Main(string[] args)
 {
  poisk();
  print();
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Sibailas matrisani shigary:
0 3 1000 4 1000 1000 1000 1000 1000
3 0 4 1000 1000 1000 1000 7 1000
1000 4 0 5 2 4 4 6 9
4 1000 5 0 6 1000 1000 1000 1000
1000 1000 2 6 0 5 1000 1000 1000
1000 1000 4 1000 5 0 3 1000 4
1000 1000 4 1000 1000 3 0 1000 4
1000 7 6 1000 1000 1000 1000 0 8
1000 1000 9 1000 1000 4 4 8 0

0 1 2 3 4 5 6 7 8
0 3 7 4 9 11 11 10 15
0 0 1 0 2 2 2 1 5
Songi tobe = 8 5 2 1 0
Bastapki tobe = 0
```

Бағдарлама жұмысының нәтижесі – екі массивті шығару, яғни графтың берілген төбесінен қалған төбелеріне дейінгі ең қысқа аралықтар мен ең қысқа маршруттардың массивтері. Мысал ретінде графтың 0-ші төбесінен 8-ші төбесіне дейінгі ең қысқа маршрут қарастырылады 8-ші мен 0-ші төбелерінің арасындағы ең қысқа аралық 15-ке тең. Орын ауыстыру маршруты: 8 – 5 – 2 – 1 – 0.

12.5 Өзін–өзі тексеру сұрақтары

1 Қабырға арқылы байланысатын төбелерді қалай атайды?

2 Графтың жолы туралы ұғым.

3 Графтың қандай жолы қарапайым жол деп аталады?

4 Егер кез келген төбеден басқа төбеге жол бар болса, онда ондай граф қалай аталады?

5 Циклдері жоқ, байланыстаралған граф қалай аталады?

6 Графта оның барлық төбелерін өзіне қосатын цикл бар болса (бірақ барлық қабырғалары міндетті емес), онда ондай графты қалай атайды?

7 Сыбайлас матрицада бағана мен жол қиылысында не жазылады?

8 Егер берілген төбеден графтың кез келген төбесіне жол бар болса, онда осы төбе қалай аталады?

9 Флойд алгоритмі нені анықтайды?

10 Дейсктр алгоритмі нені анықтайды?