МАССИВТЕРДІ СҰРЫПТАУ АЛГОРИТМДЕРІ

6.1 Алгоритмдер мен массивтер

Көптеген есептерде деректерді массив түрінде көрсету оның шешімін едәуір жеңілдетеді. Кейбір есептер «Жоғарғы математика» курсынан сізге белгілі, мысалы, полиномды есептеу.

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (6)$$

функциясы n-і дәрежелі $P_n(x)$ полином деп аталады. Мұндағы a_k мәні полином коэффициенті деп аталады және $n+1$ элементтері бар бір өлшемді массив түрінде көрсетіле алады.

$P_n(x)$ полиномының коэффициенттер массивін біле тұра полином мәнін кез келген x нүктесінде есептеуге болады. Әдетте x нүктесінде полиномды есептеу есебі кейбір алгоритмдерді жүзеге асырумен байланысты, мысалы, Горнер алгоритмі, кесіндіні тең бөлу әдісі (егер кесіндінің ұштарында полином мәні түрлі таңбада болса), итерация әдісі, Ньютона, Лагранж әдістері және т.б. Барлық осы алгоритмдер полином коэффициентінің массивін өңдеумен байланысты.

Информатиканың өзінің есептері бар, олар деректерді массив түрінде көрсетуді талап етеді. Осы деректерді өңдеудің өз алгоритмдері бар. Деректерді өңдеудің көптеген алгоритмдерінің ішінен сұрыптау (деректерді кейбір сипаты бойынша реттеу) және іздеу (деректерде берілген сипат бойынша элементті анықтау) алгоритмдері ерекшеленеді.

Математикалық есептерді шешу алгоритмі (әдісі) «Жоғарғы математика» курсында қарастырылады, сондықтан оқулықтың осы бөлімінде кейбір деректерді сұрыптау мен іздеу алгоритмдері ғана қарастырылады.

6.2 Массив элементтерін сұрыптау

Біз осы бөлімде массив элементтерін сұрыптаудың тек екі әдісінің алгоритмдерін және бағдарламалақ жүзеге асырылуын қарастырамыз.

6.2.1 Таңдау әдісімен сұрыптау алгоритмі. Массив элементтерін таңдау әдісімен, кему тәртібінде сұрыптау алгоритмі келесі амалдардың орындалуын болжайды.

Массивтің бірінші элементін қарастырамыз, оның мәнін массивтің қалған басқа элементтерімен кезекпен салыстырамыз (2-ден соңғы элементке дейін). Егер мәні үлкен элемент кезіксе, онда ол бірінші элементпен орын ауыстырады. Массивті бірінші рет тексерудің нәтижесінде бірінші элемент ең жоғары мәніне ие болады.

Содан кейін екінші элементті қарастырамыз, оның мәнін массивтің қалған басқа элементтерімен кезекпен салыстырамыз (3-ден соңғы элементке дейін). Массивті екінші рет тексерудің нәтижесінде біз сұрыпталған массивтің екінші элементінің мәнін анықтаймыз. Тексеру массивтегі соңғының алдында тұрған

элементіне дейін осылай қайталанады, соңғы тексеруде массивтің алдынғы элементі массивтің соңғы элементімен ғана салыстырылады.

Алынған массив элементтерінің мәні кему тәртібінде реттеледі.

Массив элементтерінің саны N-ге тең болса, алгоритмде орындалатын салыстыру операцияларының саны (О.С.) келесі түрде есептеледі:

К.О. = (N-1)+(N-2)+(N-3)+ . . . + 2 +1=N·(N-1)/2 (7)

Осы өрнек алгоритмнің есептеу тиімділігін сипаттайды.

6.1-есеп. 11 кездейсоқ бүтін сандардан тұратын, минус 50-ден 50-ге дейінгі аралықта a массивін құр. Оны шығару керек. Массив элементтерін кему тәртібінде сұрыптауды орындау және жаңа массивті шығару.

Массив элементтерін кему тәртібінде сұрыптау алгоритмінің бағдарламалық іске асырылуын қарастырайық.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 { int i, j, b;
  int[] a = new int[11];
  Random rnd = new Random();
  // массивті құру және экранға шығару
  Console.Write("Massivti syriptaganga dein: ");
  for (i = 0; i <= 10; i++)
  {
  a[i] = rnd.Next() % 101 - 50;
  Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
  //таңдау әдісі арқылы массив элементтерін сұрыптау
  for (i = 0; i <= 9; i++)
  for (j = i + 1; j <= 10; j++)
   if (a[i] < a[j])
   { b = a[i]; a[i] = a[j]; a[j] = b; }
  //сұрыптаудан кейін массивті экранға шығару
  Console.Write("Massivti syriptagannan kein: ");
  for (i = 0; i <= 10; i++)
  Console.Write(" {0}", a[i]);
  Console.WriteLine();
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
    Massivti syriptaganga dein: 11 16 47 9 5 14 41 18 -2 -31
49
    Massivti syriptagannan kein: 49 47 41 18 16 14 11 9 5 -2
-31
```

6.2.2 «Көпіршікті» әдіспен сұрыптау алгоритмі. Массив элементтерін өсу тәртібінде «көпіршікті» әдіспен сұрыптау алгоритмі келесі әрекеттердің орындалуын қарастырады.

Массивтің бірінші элементін екіншісімен салыстырамыз және егер біріншісі үлкен болса, онда олар орындарын алмастырады. Одан кейін массивтің екінші элементі үшіншісімен салыстырылады, егер екінші элемент үлкен болса, онда олар орындарын алмастырады және т.с.

Массивті бірінші рет «қарап шығудың» нәтижесінде массив элементінің ең үлкен мәні ең соңына жазылады.

Массивтің бірінші элементін қайтадан аламыз және оны екінші элементпен салыстырамыз – барлық процесті массивтің соңғы тұрған элементтің алдындағы элементке дейін қайталаймыз - оның мәнін құрамыз.

Соңғы салыстыру операциясына дейін осылай қайталанады.

Сонымен массив элементерінің мәні өсу ретімен сұрыпталады.

«Көпіршікті» және таңдау әдістері алгоритмдерінің есептеу тиімділіктері бірдей.

Мысал ретінде алдынғы 6.1-есебін қарастырайық.

Массив элементтерін кему тәртібінде «Көпіршікті» әдіспен сұрыптау алгоритмінің бағдарламалық жүзеге асырылуын қарастырайық.

Бағдарламаның коды:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i,j,b;
  int[] a = new int[11];
  Random rnd = new Random();
  // массивті құру және экранға шығару
  Console.Write("Massivti syriptaganga dein: ");
  for (i = 0; i <= 10; i++)
  {
  a[i] = rnd.Next()%101 - 50;
  Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
  //  «көпіршікті»  әдісі  арқылы  массив  элементтерін
сұрыптау
  for (i=0; i<=9; i++)
```

```
    for (j=0; j<=9-i; j++)
     if (a[j]<a[j+1])
     { b=a[j];a[j]=a[j+1];a[j+1]=b;}
    // сұрыптаудан кейін массивті экранға шығару
    Console.Write("Massivti syriptagannan kein: ");
    for (i=0; i<=10; i++)
    Console.Write(" {0}", a[i]);
    Console.WriteLine();
    Console.ReadLine();
   }
   }
}
```

Бағдарлама жұмысы:
Massivti syriptaganga dein: -26 -19 32 35 -19 -44 -35 36 10 -49 -5
Massivti syriptagannan kein: 36 35 32 10 -5 -19 -19 -26 -35 -44 -49

Әлбетте әрбір қарастырылған алгоритмді массив элементтерін өсу немесе кему бойынша сұрыптау үшін қолдануға болады.

Егер басқа сұрыптау алгоритмдерімен танысқыңыз келсе, Д.Кнуттың «Искусство программирования для ЭВМ» кітабын ұсынамыз, «сұрыптау мен іздеу» сұрақтарына арналған 3-том, көлемі 843 бет.

6.3 Массивте мәні бойынша элементтерді іздестіру

Д.Кнут «Искусство программирования для ЭВМ» кітабының жартысын («сұрыптау мен іздеу» сұрақтарына 3-том арналған) кейбір бір типті элементтер жиынтығында (массивте, файлда, т.б.) мән бойынша элементтерді іздестіру сұрақтарына арналған. Бұл бағдарламалауды оқу барысында іздеу алгоритмін білу қаншалықты маңызды екенін көрсетеді.

Түрлі іздеу әдістерінің ішінен олардың тек үшеуінің алгоритмдері мен бағдарламалық жүзеге асырылуын қарастырамыз – массивте элемент мәні бойынша екілік, блоктық және тізбектеп іздеу әдістері. Араластыру арқылы іздеу әдісін алгоритм деңгейінде ғана қарастырамыз.

Бізге массив элементтерінің мәндері емес іздеу алгоритмі ғана қажет, сондықтан бүтін сандардан тұратын әлементтер үшін барлық алгоритмдерді қарастырамыз. Біз «іздестіру массивінде», яғни кездейсоқ түрде құрылатын бүтін сандар массивінде ізделетін бүтін санды - «ізделетін кілтті» ұсынамыз.

6.3.1 Тізбектеп іздеу алгоритмі. Элементтің тізбектеп іздеу алгоритмі ізделетін кілтті массивтің барлық элементімен, яғни бірінші элементінен бастап соңғы элементіне дейін тізбекті түрде салыстыруға негізделген. Егер массивтің соңғы элементі қаралып қойса, онда іздеу аяқталады.

Осы алгоритмді көрсету үшін оны кітаптың керекті бетін іздеу алгоритмімен салыстырады, мысалы, үш жүзінші бет. Әрбір бет бірінші беттен үш жүзінші бетке дейін тізбекті түрде парақталады.

Осы алгоритмнің кемшілігі – массив элементтерінің саны көп болған жағдайда іздеуге айтарлықтай көп уақыт жұмсалады. Іздеу уақыты N санына пропорционал, мұндағы N - массив элементтерінің саны.

Алгоритм артықшылығы - массивте элементтерді кез келген тәртіпте және тәртіпсіз түрде орналастыруға рұқсат беру.

6.2-есеп. Іздестіру массиві 0-ден 99-дейінгі аралықтағы 20 кездейсоқ бүтін сандарынан тұрады. Диалог режімінде кез келген бүтін сан – ізделінетін кілт беріледі. Осы сан (индекстері қандай) іздестіру массивінде неше рет кездесетінін табу керек. Тізбектеп іздеу алгоритмін қолдану.

Алгоритм белгілі болғандықтан бағдарлама кодын құруға кірісеміз.

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 {
  int i,k,n;
  int[] a = new int[20];
  int[] p = new int[20];
  Random rnd = new Random();
  string buf;
  // массивті құру және экранға шығару
  Console.WriteLine("Izdestiry massivi: ");
  for (i=0;i<10;i++)
  Console.Write(" {0}", i);
  for (i=10;i<20;i++)
  Console.Write(" {0}", i);
  Console.WriteLine();
  for (i = 0; i < 20; i++)
  {
  a[i] = rnd.Next()%100;
  if (a[i]>9) Console.Write(" {0}", a[i]);
  else Console.Write(" {0}", a[i]);
  }
  Console.WriteLine();
  Console.WriteLine("Izdey kiltin engiziniz ");
  buf = Console.ReadLine();
  k = Convert.ToInt32(buf);
  n=0;
 // іздеу кілтіне сәйкес массив элементтерін іздестіру
  for (i=0;i<20;i++)
  if (k == a[i]) {p[n]=i; n++;}
  if (n==0)
Console.WriteLine("Izdey kiltine saikes element jok!!");
  else
```

```
       {
              Console.WriteLine("Izdey kiltine saikes keletin
elementter sani = {0}",n);
              Console.WriteLine("Tabilgan indexter: ");
       for (i=0;i<n;i++)
       Console.Write(" {0}",p[i]);
       Console.WriteLine();
       }
         Console.ReadLine();
      }
      }
     }
```

Бағдарлама жұмысы:

```
Izdestiry massivi:
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
 43 68 85 72 14 44 45 55 57 30 94 83 52 72 77 92 0 91 29
97
Izdey kiltin engiziniz
91
Izdey kiltine saikes keletin elementter sani = 1
Tabilgan indexter:
 17
```

6.3.2 Блоктық іздеу алгоритмі. Блоктық іздеу алгоритмі іздестіру массивінің элементтерін реттелген түрде орналасыруды талап етеді, мысалы, түйінді элементтерді өсу немесе кему ретімен, алфавиттік тәртіпте, т.б.

Іздестіруді іздестіру массивінің түйінді элементі арқылы ғана орындауға болады, мұндағы түйінді элемент іздестіру массивін реттейді.

Іздестіру массивінің элементтері мәндері бойынша өсу ретімен орналасқан деп жорамалдайық.

Бүкіл іздестіру массиві шартты түрде блоктарға бөлінеді, мысалы, бір блокта жүз элементтен. Сұратудың ізделетін кілті ретті түрде бірінші блоктан бастап әрбір блоктың соңғы элементімен салыстырылады.

Егер сұратудың ізделетін кілті кезекті массив блогының соңғы элементінен үлкен болса, онда келесі блоктың соңғы элементіне көшу керек, әйтпесе соңғы салыстыру орындалған блокта тізбекті іздеуді орындау керек.

0-ден 99-ға дейінгі аралықта жүз кездейсоқ құрылған бүтін сандарға арналған алгоритмді орындайтын бағдарламаның коды мынандай:

```
using System;
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
```

```
      {
       int i,j,b,k,n,f,m;
       int[] a = new int[101];
       int[] p = new int[21];
       Random rnd = new Random();
       string buf;
       f = 0; m = 0;
       // іздестіру массивіу құру және экранға шығару
       Console.WriteLine("Izdestiry massivi: ");
       for (i = 0; i < 100; i++)
       {
       a[i] = rnd.Next() % 100;
       if (a[i] < 10) Console.Write(" {0}", a[i]);
       else Console.Write(" {0}", a[i]);
       if ((i + 1) % 20 == 0) Console.WriteLine();
       }
       Console.WriteLine();
       // массивті сұрыптау
       for (i=0;i<99;i++)
       for (j=i+1;j<100;j++)
        if (a[i]>a[j])
        {b=a[i];a[i]=a[j];a[j]=b;}
     Console.WriteLine("Sandardi      syriptalgannan      kein
izdestiry massivi :");
       for (i = 0; i < 100; i++)
       {
       if (a[i] < 10) Console.Write(" {0}", a[i]);
       else Console.Write(" {0}", a[i]);
       if ((i + 1) % 20 == 0) Console.WriteLine();
       }
       Console.WriteLine();
       Console.WriteLine("Izdey kiltin engiziniz");
       buf = Console.ReadLine();
       k = Convert.ToInt32(buf);
       Console.WriteLine("Blok olshemin engiziniz");
       buf = Console.ReadLine();
       b = Convert.ToInt32(buf);
       // Блоктық іздеу алгоритмі
       for (i=b-1;i<100;i=i+b)
       if(k<=a[i])
       {
       j=i-b+1;n=i;
       while (j<=n || k==a[j])
       {
        if (k==a[j])
        {
```

```
        f=1;i=100;
        Console.WriteLine("Nomer:{0}", j);
        }
        j++;
       }
       }
       if (f==0)
     Console.Write("Izdey kiltine saikes element jok!");
       Console.ReadLine();
      }
      }
     }
```

```
     Бағдарлама жұмысы:
     Izdestiry massivi:
      26 3 28 95 59 6 49 15 46 21 26 61 91 72 37 60 14 34 36
88
      48 65 91 14 48 47 46 52 45 75 18 43 29 15 55 56 35 64
37 56
      56 73 8 57 10 68 22 64 45 78 54 41 6 42 0 8 12 14 47 12
      7 44 70 18 50 30 32 23 64 81 72 50 97 27 79 75 57 81 51
80
      72 82 7 14 21 77 84 87 15 47 44 68 17 25 49 12 50 37 54
80

     Sandardi syriptalgannan kein izdestiry massivi :
      0 3 6 6 7 7 8 8 10 12 12 12 14 14 14 14 15 15 15 17
      18 18 21 21 22 23 25 26 26 27 28 29 30 32 34 35 36 37
37 37
      41 42 43 44 44 45 45 46 46 47 47 47 48 48 49 49 50 50
50 51
      52 54 54 55 56 56 56 57 57 59 60 61 64 64 64 65 68 68
70 72
      72 72 73 75 75 77 78 79 80 80 81 81 82 84 87 88 91 91
95 97

     Izdey kiltin engiziniz
     44
     Blok olshemin engiziniz
     10
     Nomer:43
     Nomer:44
```

Бұл алгоритмнің артықшылығы алдыңғы алгоритмге қарағанда іздестіру уақытының аз жұмсалуы болып табылады. Іздестіру уақыты іздестіру массивіндегі блоктар қосындысына және бір блоктағы элементтер санына пропорционал. Блок

өлшемін өзгерте отырып белгілі бір деңгейде іздестіру жылдамдағын реттеуге болады.

Алгоритмнің кемшіліктері: іздестіру уақытының көп жұмсалуы, іздестіру массивін дайындау, яғни түйінді элемент бойынша сұрыптау.

6.3.3 Екілік іздеу алгоритмі. Екілік іздеу алгоритмі іздестіру массивінің элементтерін реттелген тәртіпте орналасуын талап етеді.

Іздестіру массивінің элементтері өсу ретімен орналасқан болсын. Сұратудың ізделетін кілті іздестіру массивінің орта элементімен салыстырылады.

Егер сұратудың ізделетін кілті іздестіру массивінің орта элементінен кіші болса, онда массивтің ортасы мен соңына дейінгі аралықтағы барлық элементтер кейінгі іздестіруде қолданылмайды, ал іздестіру массиві болып басынан ортасына дейінгі аралықта орналасқан барлық элементтер есептеледі.

Егер сұратудың ізделетін кілті іздестіру массивінің орта элементінен үлкен болса, онда массивтің басынан ортасына дейінгі аралықта орналасқан барлық элементтері кейінгі іздестіруде қолданылмайды, ал іздестіру массиві болып ортаншыдан кейін тұрған элементтен соңғы элементке дейін орналасқан барлық элементтер есептеледі.

Сонымен салыстыру операциясынан кейін іздестіру массивінің жартысы кейінгі іздестіруден шығарылды, сондықтан осы алгоритмді жартылай бөлу әдісі деп атайды .

Ары қарай бөлу процессі жаңа іздестіру массивімен қайталанады. Сұратуға сай элемент табылса немесе іздестіру массиві сұратуға сәйкес емес бір элементімен берілсе, онда іздестіру аяқталады.

Алпыс кездейсоқ, өсу тәртібінде құрылған бүтін сандарға арналған екілік іздеу алгоритмін орындайтын бағдарламаның коды мынандай:

```
using System;
namespace ConsoleApplication1
{
 class Program
 {
 static void Main()
 { int i, n, k, x, f, m, c, p;
  f = 0; m = 0; c = 0;
  int[] a = new int[61];
  Random rnd = new Random();
  string buf;
  // іздестіру массивін құру және экранға шығару
  Console.WriteLine("Izdestiry massivi: ");
  a[0] = rnd.Next() % 2 + 1;
  for (i = 1; i < 60; i++)
  {
  a[i] = a[i - 1] + rnd.Next() % 2 + 1;
  if (a[i-1] < 10) Console.Write(" {0}", a[i-1]);
  else Console.Write(" {0}", a[i-1]);
  if ((i) % 20 == 0) Console.WriteLine();
  }
```

```
  if (a[i-1] < 10) Console.Write(" {0}", a[i-1]);
  else Console.Write(" {0}", a[i-1]);
  Console.WriteLine();
  Console.WriteLine("Izdey kiltin engiziniz");
  buf = Console.ReadLine();
  p = Convert.ToInt32(buf);
  // екілік іздеу алгоритмі
  n = 0; k = 59;
  while (n <= k)
  {
  x = (n + k) / 2;
  if (p == a[x]) { f = 1; m = x; n = k + 10; }
  if (p <= a[x]) k = x - 1; else n = x + 1;
  c++;
  }
  // Іздеудің нәтижесін шығару
 if (f == 0)
Console.Write("Izdey kiltine saikes element jok!");
 else Console.WriteLine("Nomer:{0}", m+1);
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы, бірінші рет іске қосылуы:
Izdestiry massivi:
1 2 4 6 8 10 12 14 15 16 18 20 22 23 24 26 28 29 31 33
34 35 36 38 39 40 42 43 45 46 48 49 50 51 53 54 55 57 58 60
62 64 65 67 68 69 70 72 73 75 77 79 81 83 85 86 87 89 90 92
Izdey kiltin engiziniz
67
Nomer:44

Бағдарлама жұмысы, екінші рет іске қосылуы:
Izdestiry massivi:
1 2 4 5 7 8 10 11 13 15 17 18 20 21 22 23 24 25 27 29
31 33 34 36 38 39 41 43 45 46 48 49 50 51 53 55 57 58 59 60
61 62 64 66 67 68 69 70 72 73 75 77 78 79 81 82 84 86 88 89
Izdey kiltin engiziniz
63
Izdey kiltine saikes element jok!

Басқа алгоритмдерден қарағанда қарастырылған алгоритмнің артықшылығы - іздестіру уақытының көп жұмсалмауында. Іздестіру уақыты іздестіру массиві элементтерінің санының екілік логарифміне пропорционал. Егер іздестіру массиві 1000 элементтен тұрса, онда ең көп дегенде 10 салыстыру операцияларын қолдана

отырып іздестірудің нәтижесін алуға болады. Егер іздестіру массиві 65000 элементтен тұрса, онда барлығы 16 салыстыру операциялары керек болады. Осы алгоритмнің артықшылығы іздестіру массивінің элементтерінің саны үлкен болғанда ерекше байқалады.

Алгоритмнің кемшілігі – массивті реттеу талабы. Егер іздестіру массивінде бірдей элементтер болса, онда алгоритмді «жетілдіру» керек. Мысалы, тауарлардың атаулары немесе қызметкерлердің тегі.

6.3.4 Кілтті адреске түрлендіретін іздеу алгоритмдері хештеу (хеширование) деп аталады. Іздестіру операцияларын орындау тұрғысынан деректерді ұйымдстырудың ең жақсы әдісі деректерді массив түрінде ұйымдастыру болып табылады, ал іздеу кілті ретінде индекс мәні қолданылады. Осы жағдайда керекті элементті табу үшін деректер массивіне бір рет жүгіну жеткілікті.

Әрине осындай жүйелер «таза» түрінде сирек кездеседі. Әдетте олар шағын жүйелер, мысалы, студент нөмірі немесе оқытушы коды бойынша керекті жазбаны табуға болады.

Деректер массивінің индексін табу үшін кілттерге арнайы хеш-функция (HF) арқылы кейбір түрлендірулерді орындау керек.

Хеш-функция ізделетін элементтің кілтін 0 мен n-1 аралығындағы сандық мәнге (индексе) түрлендіреді. Әлбетте элементтер мәнін сақтау үшін N ұяшығы (хеш-кесте) бар массив керек.

Егер мүмкін болатын кілттер n-ге тең немесе одан кіші болса, онда жүйе адреске баламалы кілтті жүйеге түрленеді.

Егер мүмкін болатын кілттер n-нен үлкен болса, онда хеш-функцияны қолдану керек. Ол 0 мен n-1 аралығында кілттерді біркелкі «таратуды» қамтамасыз етеді.

Әлбетте түрлі кілттерге бірдей индекстер тағайындалуы мүмкін, өйткені хеш-функция «көпшілігі біреуге» қағидасымен жұмыс істейді. Осындай жағдайды қақтығыс деп атайды.

Берілген мәнде ықтимал қақтығыстарды шектеуге мүмкіндік беретін хеш-функцияларды ұйымдастырудың түрлі әдістері бар.

Хеш-функцияны дайындау кезінде әдетте бөлу әдісін қолданады.

Ол бастапқы кезеңде кілтті бүтін санға түрлендіреді, ал одан кейін осы сан 0 мен n-1 аралығына қосылады.

Жолдық кілттер үшін сандық мәнді алуға арналған белгілі алгоритмнің бірі былай орындалады: әрбір жолдың байтына алдынғы мән мен белгілі бір тиянақталған көбейткіштің (хэш) көбейтіндісі қосылады.

Тәжірибе түрінде анықталған, 31 мен 37 мәндері хеш-функциясының ASCII жолдары үшін тиімді жиын болып келеді. Атап өту керек, бір деректерге арналған жақсы жұмыс істеп тұрған «хеш-функция» басқа түрге арналғандарға «жаман» жұмыс істеуі мүмкін. Мысалы, бір топ адамдардың (мысалы, студенттердің) туған жылдарына байланысты кілттер бойынша хеш-функциясын пайдалану өте қиын. Коллизияны алудың нұсқасының бірі әр хеш-кесте торына арналған тізімдік құрылымын пайдалану. Хеш-кесте мен жай тізімдерді пайдалануды көптеген авторлар информатиканың ең негізгі жаңалығы деп санайды.

Мысалы, «автомобильдер» анықтама жүйесін құру барысында автомобильдер нөмірінің сандық бөлігі бойынша хеш-функциясын пайдалануға болады.

Сандық бөлігі бойынша сәйкес келетін автомобилдерді (олар айырмашылығы әріптік бөлігінде) жай тізімге ұйымдастыру.

Осындай жүйеде іздеу екі кезеңде орындалады: біріншіде – нөмірдің сандық бөлігі бойынша, ал екіншіде нөмірдің әріптік бөлігінің тізімдегі реттілік іздеуінде.

6.4 Өзін-өзі тексеру сұрақтары

1 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
for (I = 0; I < 10; I++)
for (j = I + 1; j <= 10;j++)
if (a[I] > a[j])
{b = a[I]; a[I] = a[j]; a[j] = b;} ?
```

2 Бағдарламаның келесі үзіндісінде сұрыптаудың қандай алгоритмі қолданылған:

```
 for (i=0; i<=9; i++)
 for (j=0; j<=9-i; j++)
 if (a[j]<a[j+1])
 { b=a[j];a[j]=a[j+1];a[j+1]=b;} ?
```

3 «Таңдау» әдісін қолданатын сұрыптау алгоритмінің есептеу тиімділігі қандай болады?

4 Ең үлкен орташа іздеу уақыты қандай іздеу алгоритмінде жұмсалады?

5 Массивте элементтерді блоктық іздеу алгоритмі туралы мағлұмат беру.

6 Блоктық іздеу алгоритмінде орташа іздеу уақытын қалай азайтуға болады?

7 Кілт бойынша өсу ретімен сұрыпталған массив жазбаларын екілік іздеу алгоритмі туралы мағлұмат беру.

8 Іздестіру массивінің «кілті» дегеніміз не?

9 Жазба өрісі. Іздестіру массиві жазба өрісі арқылы реттелген.

10 Хеш-функцияның маңызы қандай?

11 Араластыру кезінде қақтығысты болдырмаудың қандай нұсқасы жиі қолданылады?