Алгоритмы поиска

Алгоритмы поиска, как и алгоритмы сортировки, являются основными алгоритмами обработки данных прикладных задач.

**Алгоритмы поиска:**

–линейный (последовательный) поиск;

–бинарный поиск;

–интерполяционный поиск.

***Последовательный поиск***

Если исходный массив не упорядочен, то единственно разумным способом является последовательный перебор всех элементов массива и сравнение их с заданным значением.

Классический алгоритм поиска элемента q в массиве а[n]:

**1 шаг**: установить начальный индекс равный 1 (j=1)

**2 шаг**: проверить условие q=a[j], если оно выполняется, то сообщить, что искомое значение находится в массиве на j-о м месте и прервать работу. В противном случае продолжить работу;

**3 шаг**: увеличить индекс на 1;

**4 шаг**: проверить условие j<n+1, если выполняется, то вернуться к шагу 2, в противном случае выдать сообщение, что данное значение q в массиве не содержится

**int ssearch (int q, int**a**[ ], int n)**

**{ int j;**

**for (j=0; j<n; j++)**

**if (q= =**a**[j])  return j;**

**return -1;**

**}**

Бинарный поиск (двоичный)

Метод половинного деления. Применяется только для **предварительно упорядоченных массивов.**

Даны целое число х и массив а[n], отсортированный в порядке неубывания. Идея бинарного метода состоит в том, чтобы проверить, является ли х средним элементом массива. Если да, то ответ получен, если нет, то возможны два случая:

а)  х < среднего значения, тогда из рассмотрения исключаются все элементы массива, расположенных в нем правее среднего;

б) х > среднего значения, тогда из рассмотрения исключается левая половина массива.

Средний элемент в том и другом случае в дальнейшем не рассматривается. На каждом шаге отсекается та часть массива, где заведомо не может быть обнаружен элемент х.

**int binar (int q, int**a**[ ], int n)**

**{ int l, r, m;**

**l=0; r=n-1;**

**for (; l<=r;)**

**{ m=(l+r)/2;**

**if (q<**a**[m]) r=m-1;**

**else if (q>**a**[m]) l=m+1;**

**else return m;**

**}**

**return -1;**

**}**

Интерполяционный поиск

Если k находится между ke и kr, то номер очередного элемента для  сравнения определяется формулой:

**m=l+(r-l)\*(k-ke)/(kr-ke)**

**Пример:**Два упорядоченных массива объединить в один, тоже упорядоченный.

**#include<iostream.h>**

**#define n 5**

**void main()**

**{ int a[n], b[n], c[2\*n], I, j, k;**

**for (i=0;i<n; i++)**

**cin>>a[i];**

**for(j=0; j<n; j++)**

**cin>>b[j];**

**i=0; j=0; k=0;**

**do { if (a[i]<b[j]) c[k++]=a[i++];**

**else if (a[i]>b[j]) c[k++]=b[j++];**

**else { c[k++]=a[i++];**

**c[k++]=b[j++]**

**}**

**}**

**while ((i<n) && (i<j));**

**while (i<n)  c[k++]=a[i++];**

**while(i<n)   c[k++]=b[j++];**

**for(i=0; i<2\*n; i++)**

**cout<<c[i]<<’\t’;**

**cout<<’\n’;**

**}**

**Пример задачи с сортировкой (Решение задачи можно посмотреть, скачав файл "Задача-12"):**

Дан одномерный массив. Найти количество различных чисел в этом массиве. Использовать функцию сортировки.

**#include<iostream.h>**  
**#include<math.h>**  
**#include<conio.h>**  
**#define n 10**  
**void sort(int mas[n])**  
**{int i,j,c;**  
**for (j=0;j<n;j++)**  
**{for (i=n-1;i>=1;i--)**  
**if (mas[i]>mas[i-1]) {c=mas[i-1];**  
**mas[i-1]=mas[i];**  
**mas[i]=c;**  
**}**  
**}**  
**}**  
**main()**  
**{int i, kol=0, m[n];**  
**clrscr();**  
**cout<<"Vvedite elementy massiva"<<'\n';**  
**for (i=0; i<n; i++)**  
**cin>>m[i];**  
**cout<<"Ishodniy massiv:"<<'\n';**  
**for(i=0;i<n;i++)**  
**cout<<m[i]<<" ";**  
**cout<<'\n';**

**sort (m);**  
**cout<<"Otsortirovanniy massiv:"<<'\n';**  
**for (i=0; i<n; i++)**  
**cout<<m[i]<<" ";**  
**cout<<'\n';**  
**for (i=0; i<n; i++)**  
**if (m[i]!=m[i+1]) kol++;**  
**cout<<"Kolichestvo razlicnih chisel= "<<kol;**  
**return 0;**  
**}**

**Пример задачи с интерполяционным поиском(Решение задачи можно посмотреть, скачав файл "Задача-13"):**

Два упорядоченных массива объединить в один, тоже упорядоченный.

**#include<iostream.h>**  
**#include<conio.h>**  
**#define n 5**

**interpolationSearch(int a[], int key, int n1)**  
**{ int left = 0;**  
**int right = n1 - 1;**  
**int mid;**  
**while ((a[left] < key) && (key < a[right]))**  
**{**  
**mid = left + (key - a[left]) \* (right - left) / (a[right] - a[left]);**  
**if (a[mid] < key) left = mid + 1;**  
**else if (a[mid] > key) right = mid - 1;**  
**else return mid;**  
**}**  
**if (a[left] == key) return left;**  
**else if (a[right] == key) return right;**  
**else return -1;**  
**}**  
**void main()**  
**{ int a[n], b, i, j, k;**  
**clrscr();**  
**cout<<"vvedite otsortirovanniy po vozrastaniyu massiv iz "<<n<<" elementov"<<endl;**  
**for (i=0;i<n; i++)**  
**cin>>a[i];**  
**cout<<"vvedite chislo dlya poiska: " ;**  
**cin>>k;**  
**j=interpolationSearch(a,k,n);**  
**if (j==-1) cout<<"chislo "<<k<<" v massive otsutstvuet"<<endl;**  
**else cout<<"chislo "<<k<<" nahoditcya v position "<<j<<endl;**  
**}**

Источники:

1. https://ucheb2015.ucoz.net/index/metody\_sortirovki\_i\_poiska\_dannykh/0-43