[**基础查找算法**](http://www.cnblogs.com/chaobao/archive/2012/02/11/2347023.html)

顺序查找

static int SeqSearch(int[] arr, int sValue)

{

    for (int index = 0; index < arr.Length ; index++) //小bug

        if (arr[index] == sValue)

            return index;

    return -1;

}

查找最大和最小

static int FindMin(int[] arr)

{

int min = arr[0];

for(int i = 0; i < arr.Length; i++)

if (arr[i] < min)

min = arr[i];

return min;

}

static int FindMax(int[] arr)

{

int max = arr[0];

for(int i = 0; i < arr.Length; i++)

if (arr[i] > max)

max = arr[i];

return max;

}

自组织加快查找速度

要查找数据会遵循“80-20”规则，查找80%的查找操作，是为了查找20%的数据。

这种概率分布称为帕累托分布。

static int SeqSearch(int sValue)

{

for(int index = 0; index < arr.Length; index++)

if (arr[index] == sValue && index > (arr.Length \* 0.2))

{

swap(index, index-1);

return index;

}

else

if (arr[index] == sValue)

return index;

return -1;

}

static void swap( int item1,  int item2)

{

int temp = arr[item1];

arr[item1] = arr[item2];

arr[item2] = temp;

}

以上查找是需要集合在无序的状态下查找，下面看有序时候的高效查找方法

二叉(二分查找)查找算法

public int binSearch(int value) //回溯版本

{

    int upperBound, lowerBound, mid;

    upperBound = arr.Length - 1;

    lowerBound = 0;

    while (lowerBound <= upperBound)

    {

        mid = (upperBound + lowerBound) / 2;

        if (arr[mid] == value)

            return mid;

        else

            if (value < arr[mid])

                upperBound = mid - 1;

            else

                lowerBound = mid + 1;

    }

    return -1;

}

public int RbinSearch(int value, int lower, int upper)   // 递归版本

{

    if (lower > upper)

        return -1;

    else

    {

        int mid;

        mid = (int)(upper+lower) / 2;

        if (value < arr[mid])

            return RbinSearch(value, lower, mid - 1);

        else if (value == arr[mid])

            return mid;

        else

            return RbinSearch(value, mid + 1, upper);

    }

}

public int Bsearh(int value) // framework 内置方法,实现了二分查找，效率最高

{

return Array.BinarySearch(arr, value);

}

三种方法比较，内置方法最快，回溯版本次之，递归慢。

System.Collections.SortedList 和System.Collections.SortedList<Tkey,Tvalue>

类用于存放键值集合，他们元素存放在线性表中，并且按照键搜索排序，其中 Sotedlist  用另个数组分表存放Key ,Value ，并巧妙地应用二分查找使得性能和Arrlist 相似。

二叉查找树

二叉查找树前面已经介绍过此处略去。

小结 ：

1顺序查找速度最慢，适用于数据量不大的情况。

2二分查找适合在有序表中，效率非常高。

3二叉查找适用求最大，最小，查找效率比较高，并且数据增删成本较低，使用月数据经常增删情况, 但二叉查找树的形态跟数据的输入顺序有关，在最坏情况，他们会退化成链表。