快速排序：

public static void quickSort(int[] a){

sort(a,0,a.length-1);

}

public static int partition(int a[],int low,int high){

int i,j,t;

i = low;

j = high;

while(i<j){

while(i<j && a[i] < a[low]){

i++;

}

while(i<j && a[j]>=a[low]){

j--;

}

t = a[i];

a[i] = a[j];

a[j] = t;

} // 最后一次情况下，j 在 i的前面，这时候表示已经遍历完了整个数组，将首项与j 互换位置，定位完成。

a[j] = a[low];

return j;

}

找到首项的位置

public static void sort(int[] a, int low, int high){

if(high <= low){

return;

}

int j = partition(a,low,high);

sort(a,low,j-1);

sort(a,j+1,high);

}

递归，给每一项定位

归并排序以及快速排序的improvement：

快速排序：

排序最差情况（降序）优化: 随机化数组 shuffle

Knuth Shuffle:

public static void shuffle(int[] a){

for(int i = 0;i<a.length;i++){

int j = (int)(Math.random()\*(i+1));

int t = a[i];

a[i] = a[j];

a[j] = t;

}

}

随机数的产生方法：

若产生一个M-N的随机数：(int) Math.random()\*(N-M+1)+M

Math.random 产生一个[0,1)的随机double型

归并排序优化：

Bottom-up mergesort

Pass through array, merging subarrays of size 1

Repeat for subarrays of size 2,4,8,16

优势：不用递归，只用循环。节省运行时间以及存储空间

public static void sort(int[] a){

int N = a.length;

for(int sz = 1; sz<N; sz=sz+sz){

for(int lo = 0; lo<N-sz; lo+=sz+sz){

merge(a,lo,lo+sz-1,Math.min(lo+sz+sz-1,N-1));

}

}

}

Import java 的排序包：

在java IDE 中，一般可以直接使用已经有的java包，用import方法

import java.util.Arrays;

Arrays.sort(array);

Arrays 包即采用的是归并排序和快速排序。对于数字，采用快速排序，对象则采用归并排序。对象采用Comparable方法比较，具体内容自行查找。

二维数组：

二维数组：

二维数组的初始化方式：

int[][] m = {{1,2,3},  
        {2,3,4}};

int m[][];  
m = new int[4][4];

使用这种方法，初始化出的第二维的长度都是相同的，如果需要初始化第二维长度不一样的二维数组，则可以使用如下的格式：  
               int n[][];  
               n = new int[2][]; //只初始化第一维的长度  
               //分别初始化后续的元素  
               n[0] = new int[4];  
               n[1] = new int[3];

             这里的语法就体现了数组的数组概念，在初始化第一维的长度时，其实就是把数组n看成了一个一维数组，初始化其长度为2，则数组n中包含的2个元素分别是n[0]和n[1]，而这两个元素分别是一个一维数组。后面使用一维数组动态初始化的语法分别初始化n[0]和n[1]。

作业：

1. 6+6+2

2. 1213121

4. 5+4+2+1+3

斐波那契

public static int fibonacci(int input){

if (input == 0||input == 1){

return input;

}else{

return fibonacci(input - 1) + fibonacci(input - 2);

}

}

习题：

String 类，public类，所有都可以使用。

String 类的方法，对String对象进行操作。 s.Method() 的方法。

String类的方法：

String length(); // return String’s length

The index of the String’s first character is 0, last character is a null string”” which has index of length

String substring(int startindex); //return a string starts with the character of startindex and ends with the last character of this string, if startindex > length, throw StringIndexOutOfBoundsException

String substring(int startindex, int endindex); // return a string starts with the character of startindex and ends with the character of endindex-1 if

int indexOf(String str); // return the index of the first occurrence of str within this string. If is not a substring, return -1

public ColorGrid(String s, int numRows, int numCols){

myRows = numRows;

myCols = numCols;

index = 0;

**myPixels = new String[myRows][myCols];**

for(int i = 0; i<myRows;i++){

for(int j = 0; j<myCols;j++){

myPixels[i][j] = s.substring(index,index+1);

Index++;

}

}

}