

第四题plus

3.汉诺塔

```
package disiti;

import java.util.Scanner;

public class Hannuota {

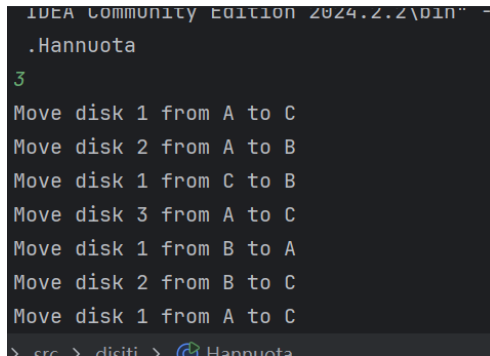
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int numDisks = n; // 设置汉诺塔的圆盘数量
        char source = 'A'; // 初始柱子
        char auxiliary = 'B'; // 辅助柱子
        char target = 'C'; // 目标柱子

        solveHanoiTower(numDisks, source, auxiliary, target);
    }

    public static void solveHanoiTower(int n, char source, char auxiliary, char target) {
        if (n == 1) {
            System.out.println("Move disk 1 from " + source + " to " + target);
            return;
        }
        // 将 n-1 个圆盘从 source 移动到 auxiliary, 使用 target 作为辅助柱子
        solveHanoiTower(n - 1, source, target, auxiliary);

        // 将第 n 个圆盘从 source 移动到 target
        System.out.println("Move disk " + n + " from " + source + " to " + target);

        // 将 n-1 个圆盘从 auxiliary 移动到 target, 使用 source 作为辅助柱子
        solveHanoiTower(n - 1, auxiliary, source, target);
    }
}
```



```
IDEA COMMUNITY EDITION 2024.2.2\BIN" -
.Hannuota
3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
> src > disiti > .Hannuota
```

运用了递归的办法，不断地把整个问题分解成一个更小的问题，直到变成最基础的那个问题。所以我们要注意设计分解什么时候停止（ $n=1$ 时）。

4.归并排序

```
package disiti;

import java.util.Scanner;

public class Guibinpaixu {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        int n=scanner.nextInt();
        int[] nums = new int[n] ;
        for(int i=0;i<n;i++){
            nums[i]=scanner.nextInt();
        }

        mergeSort(nums,0,n-1);

        for (int i=0;i<n;i++) {
            System.out.print(nums[i]+" ");
        }
    }

    private static void mergeSort(int[] nums, int l, int r) {
        if(l>=r) return;
        int mid=l+r>>1;//求中间位置

        mergeSort(nums,l,mid);
        mergeSort(nums,mid+1,r);
        int temp[]=new int [r-l+1];
        int i=l,j=mid+1,k=0;
        while(i<=mid && j<=r){
            if(nums[i]<=nums[j]){
                temp[k++]=nums[i++];
            }
        }
        while(i<=mid){
            temp[k++]=nums[i++];
        }
        while(j<=r){
            temp[k++]=nums[j++];
        }
    }
}
```

```

        else {temp[k++]=nums[j++];}

    }
    while(i<=mid){
        temp[k++]=nums[i++];}
    while(j<=r){
        temp[k++]=nums[j++];
    }
    for(i=1,j=0;i<=r;i++,j++){
        nums[i]=temp[j];
    }

}

}

}

```

也运用了递归的想法

主要步骤如下：分解，排序，合并

时间复杂度：描述算法运行时间与输入数据规模之间关系的量度。归并排序的空间复杂度是 $O(n)$ ，算法的运行时间和输入数据规模成正比。

```

5
2 4 6 7 1
1 2 4 6 7
进程已结束，退出代码为 0
|

```

5.瑞士轮

```

package player;

import java.util.Scanner;

public class Player {
    public static void main(String[] args) {
        Player player=new Player();

        Scanner scanner = new Scanner(System.in);
        System.out.println("请输入N R Q");
        String input = scanner.nextLine();
        String [] temp = input.split(" ");//第一次用这个，将同时输入的数据分开
        int n=Integer.parseInt(temp[0]);
        int r = Integer.parseInt(temp[1]);
        int q = Integer.parseInt(temp[2]);
        int[] no = new int[2*n]; //编号
        int[] s = new int[2*n]; //初始分数
        int[] w = new int[2*n]; //实力值
        System.out.println("请输入对应编号选手初始分数：");
        input = scanner.nextLine();
        temp = input.split(" ");
        for(int i=0;i<temp.length;i++){
            s[i]=Integer.parseInt(temp[i]);
            no[i]=i+1;
        }
        System.out.println("请输入对应编号选手实力值");
        input= scanner.nextLine();
        temp = input.split(" ");
        for(int i = 0; i < w.length;i++){
            w[i] = Integer.parseInt(temp[i]);
        }
        //前期准备已结束，准备开始比赛
        int i=0;
        while(i<r){
            i++;
            System.out.println("第"+i+"轮比赛后");
            player.sgleAfter(no,s,w);
            for(int t = 0 ; t < s.length;t++) {
                System.out.println("第" + (t + 1) + "名：" + no[t] + "，分数：" + s[t]);
            }
        }
        System.out.println("第"+r+"轮后排名第"+q+"的编号是：" +no[q-1]);

    }

    public void sgleAfter(int[] no, int[] s, int[] w){
        for(int i=0;i<s.length-1;i+=2){
            if(w[i]>w[i+1]){
                s[i]++;
            }else{
                s[i+1]++;
            }
        }
        quicksort(no,s,w,0,s.length-1);
    }
    //比赛，对胜者加分
    public void quicksort(int []s,int[]no,int []w,int left,int right){
        if(left<right){
            int pivot=madian(s,no,w,left,right);
            if(right-left>1){
                int i=left;
                int j=right;
                while(true){
                    while(s[i]>pivot){
                        if(s[i]==pivot&&no[i]>no[right]){

```

```

        }
    }
    while(s[--j]<=pivot){
        if(s[j]==pivot&&no[j]<=no[right]){
            break;
        }
    }
    if(i<j){
        swapreference(s,no,w,i,j);
    }else{
        break;
    }
}
swapreference(s,no,w,i,right);
quicksort(s,no,w, left,i-1);
quicksort(s,no,w, i+1, right);
}
}

} //这个方法用于排序
public void median(int []s,int []no,int []w,int left,int right){
    int center=(left+right)/2;
    if (s[left] < s[right]) {
        swapreference(s,no,w,center,left);
    }
    if (s[center] < s[right]) {
        swapreference(s,no,w,right,left);
    }
    if(s[left]<s[center]){
        swapreference(s,no,w,left,center);
    }
    if(left+1>right){
        swapreference(s,no,w,center,right);
    }
    return s[right];
}

//找基准值
public void swapreference(int []s,int[]no,int []w,int i,int j){
    int temp=no[i];
    no[i]=no[j];
    no[j]=temp;

    temp=s[i];
    s[i]=s[j];
    s[j]=temp;

    temp=w[i];
    w[i]=w[j];
    w[j]=temp;
}

} //换顺序，确保三个数组中的元素被同时交换，保证了数据的一致
}
```

```
.Player
请输入N R Q
2 4 2
请输入对应编号选手初始分数:
7 6 6 7
请输入对应编号选手实力值
10 5 20 15
第1轮比赛后
第1名:4,分数: 7
第2名:3,分数: 7
第3名:2,分数: 6
第4名:1,分数: 8
第2轮比赛后
第1名:4,分数: 7
第2名:3,分数: 8
第3名:2,分数: 6
第4名:1,分数: 9
第3轮比赛后
第1名:4,分数: 7
第2名:3,分数: 9
第3名:2,分数: 6
第4名:1,分数: 10
第4轮比赛后
第1名:4,分数: 7
第2名:3,分数: 10
第3名:2,分数: 6
第4名:1,分数: 11
第4轮后排名第2的编号是: 3

进程已结束,退出代码为 0
```

在学归并排序的时候,看了一下快

速排序,好像这个使用频率更高一点

在写程序的时候遇到了2个难点

1.如何处理输入的数据,把同时输入的数据分开

2.排序,方法真的就是一个套一个,排序方法也不熟悉,看网上说快速排序要简单一点,没感觉出来。