

- 作业截止每**周三22:00**，晚交不参与红包
- 提交到stem111@qq.com
- 标题格式：**cs100第X周作业-姓名**，如**cs100第1周作业-张三**)
- 非编程题统一写成一个docx附件
- 编程题提交cpp文件附件，文件名会在题目里注明
- 有任何问题/吐槽可以写在正文里😏

作业回顾：非编程题

1.想一想，你为什么来学编程/学算法

从小对电脑感兴趣，喜欢编程

竞赛，升学，找工作，走上人生巅峰

想要编写游戏，操作机器人，...

觉得会编程很厉害

与同学约好一起学习编程

[真爱难不倒技术宅:美数学家利用大数据找到女朋友_爱活网 Evolife.cn](http://www.evolife.cn)



2014年1月24日 - 可以说没有什么问题能够真正难倒他们——即使是用数学方法找到一名女朋友,...者,于是他决定利用自己收集的**大数据**进行更加精确的配对,从而让自己找到真爱...

www.evolife.cn/html/20... - 百度快照 - 96%好评

为了赚钱

锻炼自己强化思维能力，找乐子，拓展知识面

2.去查一查，你能找到多少种现存的编程语言？

有同学去查了Wikipedia，找到700多种👍

所以有时候找对工具很重要

作业回顾：问题总结

- **新语言感觉怕怕？**

- 对照样例程序一行行输入一遍（脱敏）
- 语言是死的，就那点语法。输至多10个程序就熟了
- 遇到问题是好事，解决问题才能有进步

我好怕怕哦！



- **看代码的问题**

- 没有def，带参数声明的都是函数，程序主要是函数和变量组成的
- 由{}勾勒出程序的层次结构，**从外层向内层**看，不要像读书那样一行行看

- **写代码的问题**

- 每个程序都需要有main函数（程序开始运行的地方）
- 后面的函数可以调用前面的，反之不行。所以main函数要放在最后
- 不要忘记预编译代码，下面两个基本是每个程序都要的
 - #include <iostream>
 - using namespace std;
- **提交作业前运行一下，至少样例数据要通过**

作业回顾：编程题

3.min substr.cpp

4.equation.cpp

5.max substr better.cpp



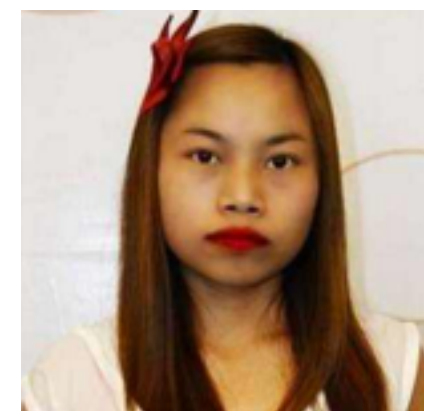
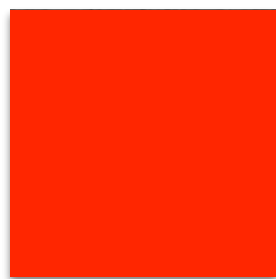
Welcome to Xcode

CS100 算法入门

排序

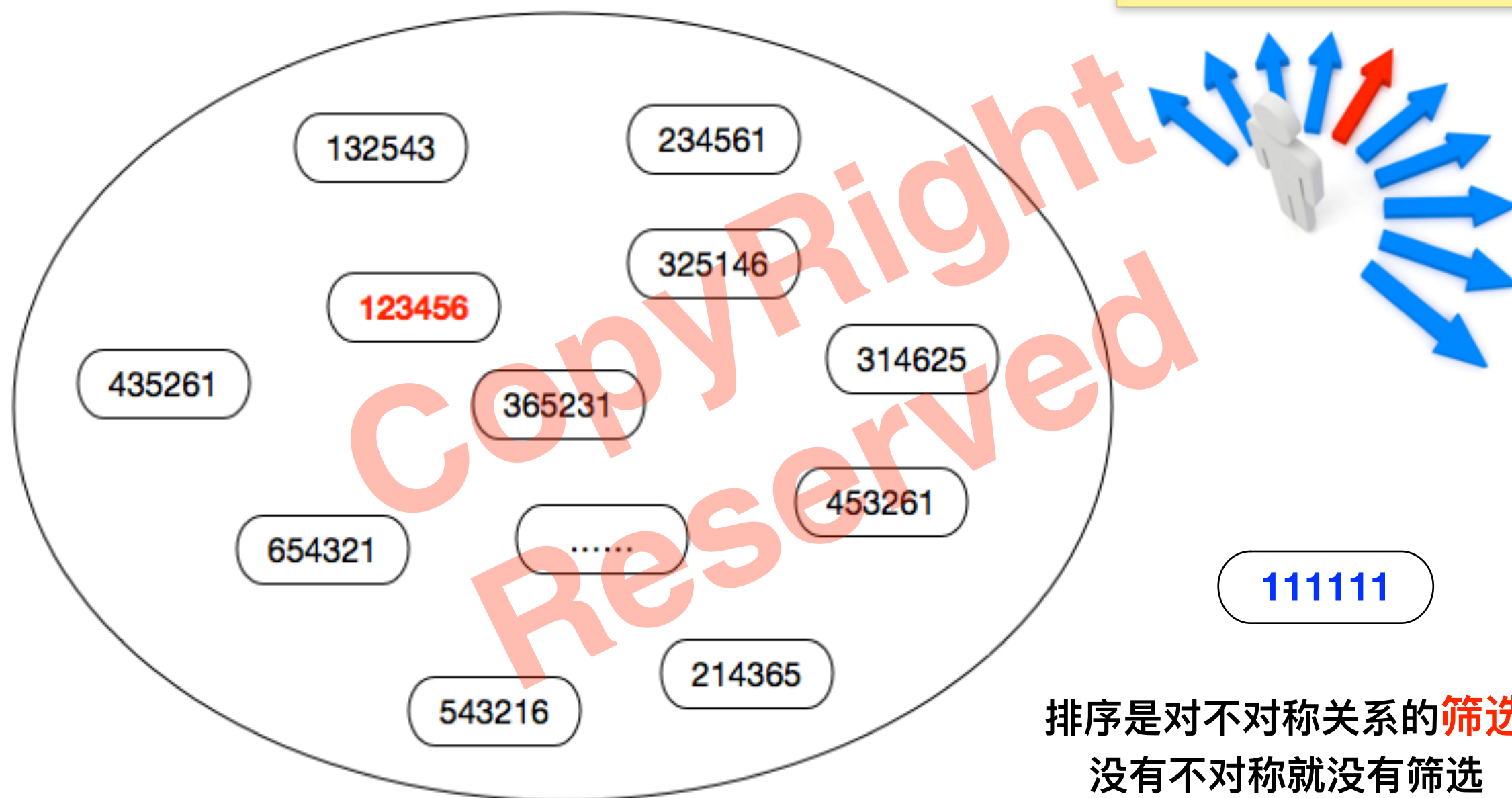
先讲一点哲学：序

序是一种**不对称关系**的描述



先讲一点哲学：排序

按第几位置标记，等价问题转换



史上最牛排序算法

输入数组a

```
while (a不是有序的) {  
    随机打乱a的元素顺序  
}
```

输出a

→ 你这是什么语言？还带中文的？

这叫做“伪代码”。是用来表达程序思路和交流用的。回想一下**语言只是工具**，有时候程序甚至**不需要真的写代码出来运行**

其实还真有场合适合使用这种算法的，你猜得到么😊

给你厉害的
你咋不上天呢



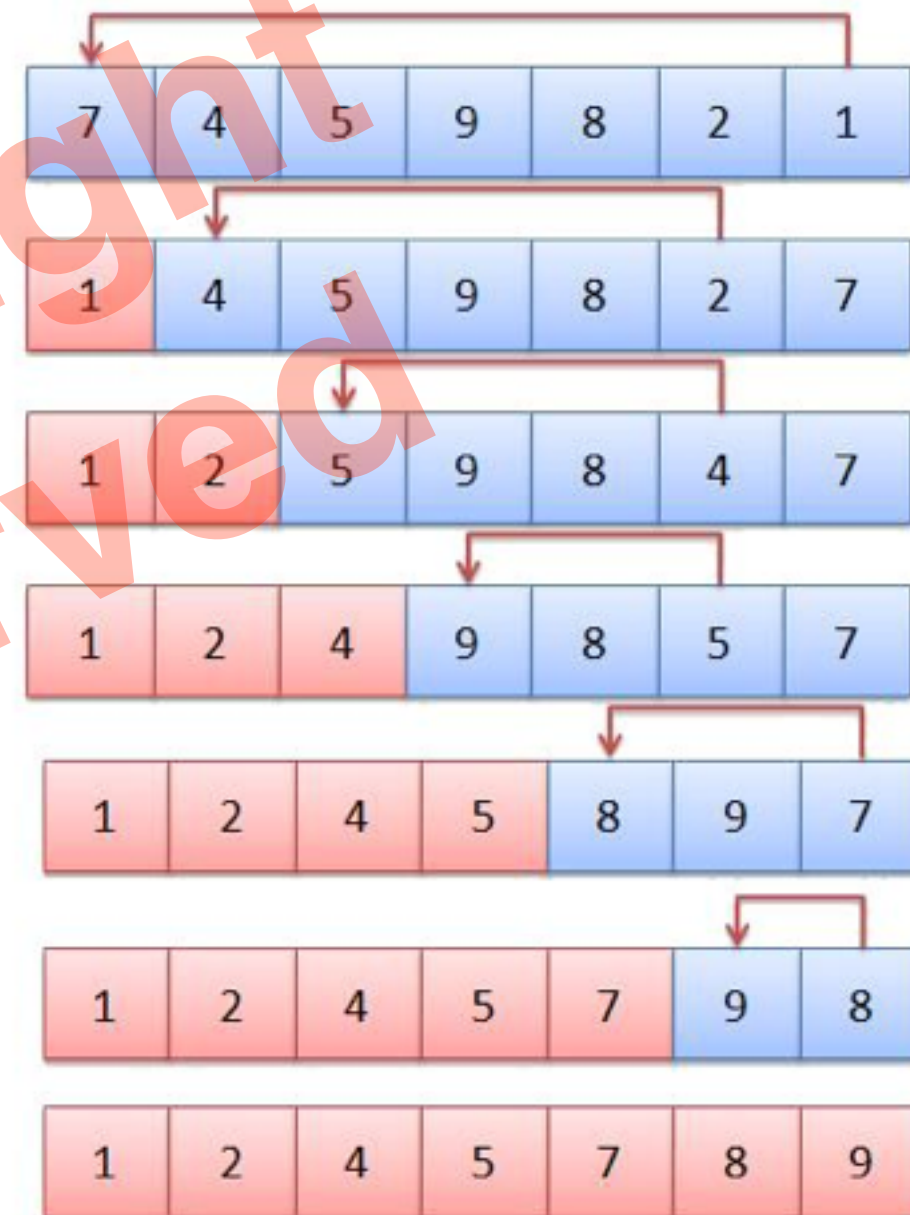
选择排序：最直观的排序算法

输入数组a，长度n

```
for (int i=0;i<n-1;i++) {  
    从a[i:n]中选出最小的元素a[j]  
    交换a[i]与a[j]  
}  
输出a
```

```
14 void sortBySelect(int data[],int n){  
15     for (int i=0;i<n-1;i++){  
16         int least=i;  
17         for (int j=i+1;j<n;j++){ // 循环选择i..n-1中最小的元素  
18             if (data[j] < data[least]){  
19                 least=j;  
20             }  
21         }  
22         // 交换元素i和元素least  
23         int tmp=data[i];  
24         data[i]=data[least];  
25         data[least]=tmp;  
26     }  
27 }
```

恭喜！你已经会写排序了
然鹅感觉有点low？
下面来一种比较高大上的



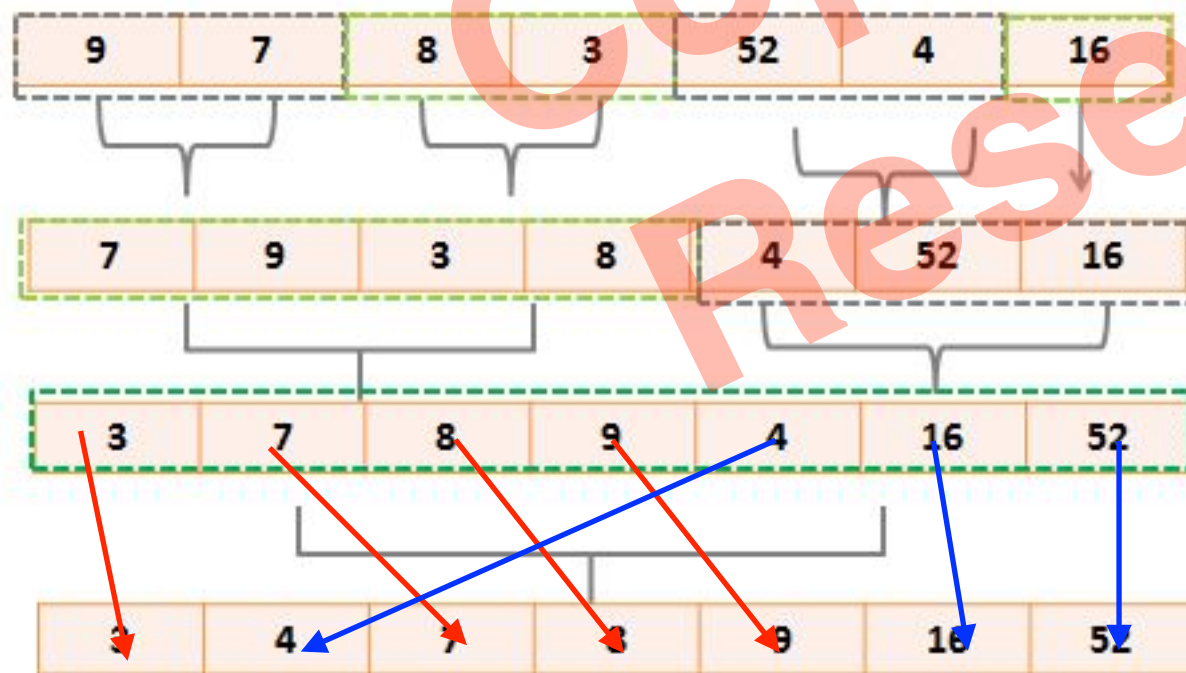
演示一下

归并排序：分治思想

```
void sort(a[left:right]){  
    mid=(left+right)/2  
    递归排序子数组 sort(a[left:mid])  
    递归排序子数组 sort(a[mid+1:right])  
    合并左右两个已排序的子序列  
}
```

分治思想的精髓：

当你在归并的时候，子问题是已经解决好(排好序)的
相当于平白多出一个有利条件
回去好好参悟一下这句话🤔下次还会细讲



悟

归并排序：解答

```
29 void sortByMerge(int data[],int head,int tail){
30     if (head==tail) { // 递归边界
31         return;
32     }
33     int mid=(head+tail)/2;
34     // cerr<<"head="<<head<<" mid="<<mid<<" tail="<<tail<<"\n";
35     sortByMerge(data,head,mid); // 排序左子列
36     sortByMerge(data,mid+1,tail); // 排序右子列
37
38     int left=head, right=mid+1, i=0; // 左右两个游标
39     int tmp[tail-head+1]; // 存放临时结果
40     while (left<=mid || right<=tail) { // 归并过程
41         // 把左右游标中较小的一个放入tmp的当前位置
42         if (right>tail || (left<=mid && data[left]<data[right])){
43             tmp[i++]=data[left++];
44         } else{
45             tmp[i++]=data[right++];
46         }
47     }
48     // 将tmp (排好序的) 回填到原数组的head到tail位置
49     for (int j=head;j<=tail;j++){
50         data[j]=tmp[j-head];
51     }
52 }
```

→ 注意：我为了节省篇幅省掉了include、using和main函数。你在写的时候不要忘记加上

恭喜！你已经会写一种高级排序了
有没感觉高大上？



复杂度：算法优劣的鉴定

史上最牛排序	$O(n!)$
选择排序	$O(n^2)$
插入排序	$O(n^2)$
冒泡排序	$O(n^2)$
希尔(Shell)排序	$O(n^{1.3})$
快速排序	$O(n \log n)$
归并排序	$O(n \log n)$
堆排序	$O(n \log n)$
基数排序	$O(dn)$

→ 这个大O是什么鬼？这称为算法的（渐近）复杂度
 n 表示数据规模， O 表示 n 的一个函数的渐近趋势（装逼用）
直观上，可以理解成当 n 增大时数值变大速度最快的一项

→ 很抽象？看几个例子你就懂了

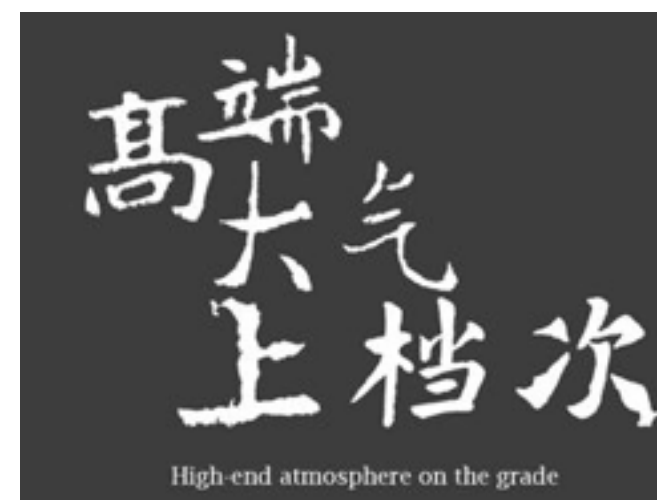
$$3n^2 + n + 1 = O(n^2)$$

$$n \cdot \log n + 3n = O(n \log n)$$

$$2^n + 100n^2 + 1000n = O(2^n)$$

→ 大O符号在数学上有严格定义（等你学高等数学的时候可能会学到😄）。精准复杂度分析是一门专门的学术分支，但我们只会做一些简单的复杂度估算

纳尼？你没觉得哪里高大上？
我来告诉你怎么评判算法的高大上



感受一下不同复杂度下的算法规模

		n=1	n=10	n=100	n=1000	n=10000	
O(1)	散列查找	1	1	1	1	1	有效 算法
O(logn)	二分查找	1	4	7	10	14	
O(n)	递推	1	10	100	1000	10000	
O(nlogn)	归并排序	1	40	700	10000	140000	
O(n²)	选择排序	1	100	10000	1E+06	1E+08	
O(n³)	两两最短路 (Floyd算法)	1	1000	1E+06	1E+09	1E+12	
O(2ⁿ)	分数0-1背包	2	1024	1E+30	1E+301	1E+3010	
O(n!)	回溯法(无剪枝)	1	3628800	1E+158	1E+2568	1E+35659	



猜算法

$O(1)$	∞	数学方法
$O(\log n)$	$>10^9$	分治法
$O(n)$	10^6-10^9	递推, 贪心法
$O(n \log n)$	10^6-10^8	分治法, 树
$O(n^2)$	10^4-10^5	动态规划, 图论, 递推
$O(n^3)$	10^2-10^3	动态规划, 图论
$O(2^n)/O(n!)$	<20	搜索, 回溯法

复杂度分析有什么用?

有时下可以帮你大致猜出用什么算法解题

因为以当前计算机的计算能力, 在竞赛可接受时间内运行完一个程序,
可以执行的计算次数大概在 10^8-10^{10} 左右



基于比较的排序复杂度下限

排序究竟能做到多快？

1. n 个数字的排列顺序共 $n!$ 种 (称为状态空间)

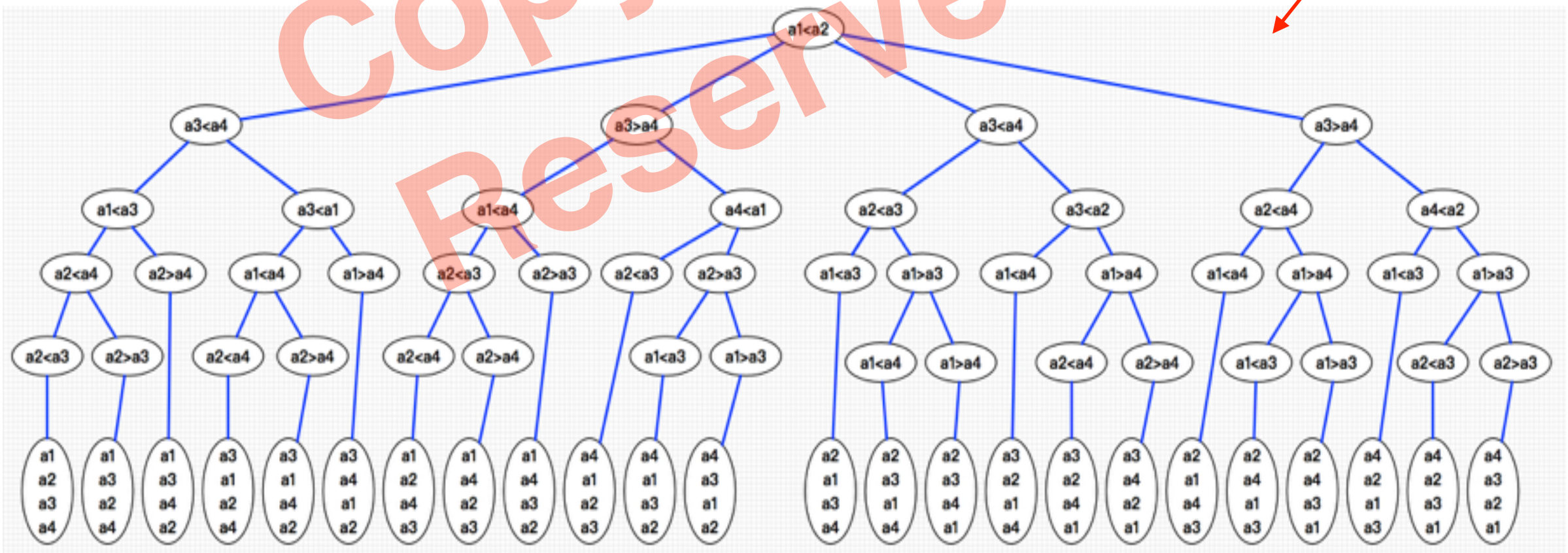
2. 每次比较会产生 2 个结果, x 次比较 2^x 种

3. 要选出正确的顺序, 必须满足 $2^x > n!$

$\Rightarrow x > \log(n!) = O(n \log n)$

→ 最后这步称为斯特林定律 (Stirling's Law), 你
现在不需要知道怎么证明 😊

解答树 (以后会详细介绍什么是树 🌲)



此处插播硬广告：vector类型

```
14 #include<vector>    // 使用容器类型需要加香油include
15 using namespace std;
16
17 void printVector(vector<int> v){
18     cout<<"v=[ ";
19     for(vector<int>::iterator it=v.begin();it!=v.end();it++){
20         cout<<*it<<" ";
21     }
22     cout<<"]"<<endl;
23 }
24
25 void vectorDemo(){
26     vector<int> v;
27     printVector(v);
28     cout<<"size="<<v.size()<<endl<<endl;    // 输出尺寸
29
30     v.push_back(1); v.push_back(2); v.push_back(3); // 向后追加
31     printVector(v);
32     cout<<"v[1]="<<v.at(1)<<endl;    // 单个元素
33     cout<<"size="<<v.size()<<endl<<endl;
34
35     v.pop_back();    // 删除尾部元素
36     printVector(v);
37     cout<<"size="<<v.size()<<endl<<endl;
38
39     v.clear();    // 清空列表
40     printVector(v);
41     cout<<"size="<<v.size()<<endl<<endl;
42 }
```

→ 容器指多个同类数据组成的复合数据类型，比如数组也可以理解成一种容器

→ vector是C++提供的标准容器之一，你目前可以理解为**长度可变的数组**，但vector功能比数组更齐全，使用起来更方便

```
v=[ ]
size=0

v=[ 1 2 3 ]
v[1]=2
size=3

v=[ 1 2 ]
size=2

v=[ ]
size=0
Program ended with exit code: 0
```

 百度 | 移动DSP投放服务

madv.baidu.com

基数排序：基于值域的排序（选学）

```
输入数组a，长度n
准备10个vector(称为桶)，对应0-9十个数字
for (d从个位到最高位) {
    for (i=0; i<n;i++){
        a[i]的第d位数为x，则a[i]入队到第x个桶
    }
    for (x=0; x<10;x++){
        桶x顺序出队并写入a中
    }
}
输出a
```

挑战一下更有难度的排序算法！

基数排序仅限正整数排序
如不超过d位数，则复杂度为O(dn)
因为需要预先知道数据的取值范围 (0-10^d)，所以称为
基于值域的排序

053	0		111	0	005	005	0	005,014,053,063	005
542	1	111	542	1	111,512,014,616,217	111	1	111,134,154	014
005	2	542,742,512,452	742	2		512	2	217	053
063	3	053,063	512	3	134	014	3		063
014	4	014,154,134	452	4	542,742,748	616	4	452	111
217	5	005	053	5	452,053,154	217	5	512,542	134
154	6	616	063	6	063	134	6	616	154
748	7	217	014	7		542	7	742,748	217
616	8	748	154	8		742	8		452
742	9		134	9		748	9		512
111			005			452			542
412			616			053			616
452			217			154			742
134			748			063			748

第d次操作后，数组的后d位为有序状态

基数排序：实现

```
15 using namespace std;
16 vector<int> buckets[10]; // 10个桶 (队列)
18
19 void baseSort(int data[],int n){
20
21     int max=data[0]; // 找最大元素作为循环边界
22     for (int i=1;i<n;i++){
23         max=data[i]>max?data[i]:max;
24     }
25
26     for (int base=1;base<=max;base*=10) { // 从低到高循环每一位
27         for(int i=0;i<n;i++) {
28             int index=(data[i]/base) % 10; // 提取一位
29             buckets[index].push_back(data[i]); // 顺序插入对应桶
30         }
31
32         for (int i=0,k=0;k<10;k++) {
33             vector<int> vec=buckets[k]; // 依次提取每个桶
34             for(int j=0;j<vec.size();j++) {
35                 data[i++]=vec.at(j);
36             }
37             buckets[k].clear(); // 清空桶, 准备下一次
38         }
39     }
40 }
```

→ 后面的<int>是什么？这种语法称为**模板**。你理解为这是**整数组成的**可变数组就行了

→ 为什么用vector不用数组？因为桶里装几个数是变化的，用vector可以节省空间

→ 基数排序适用的场合：

如果n个数互不相同，最大的数至少是n，而n的位数至少是 $O(\log n)$ （参见作业第3题）。于是 $O(dn)$ 就是 $O(n \log n)$

所以只有当排序的数组有**大量重复**时，基数排序才会优于归并排序



作业

1. 举三个生活中用到排序的例子



2. (选做) 你觉得史上最牛排序法在什么场合适合应用?
(只要说的有道理就有红包, 不一定要猜到我想的答案)

3. 写出以下表达式对应的渐进复杂度

1) $7n+n^2+6$ 2) $2n\log n+3n^3-100$ 3) $2^n+n^{100}+3n$

4. 输入两组整数 (数量 <1000000), 输出其中公共的部分

样例输入:

10
13 -21 11 -31 32 22 -12 33 23 -99
25 13 -12 37 36 99 11 23 33 45

样例输出:

-12 11 13 23 33

(提示: 排序你们肯定能想到, 排序以后怎么办? 参考一下归并排序的合并过程)

NOTE: 提交代码前运行一下, 至少把样例跑对



5. (选做) \log 究竟是什么? 去学习一下什么是**对数**? 对数有什么特性?

如果觉得学会了, 试试这道题:

输入正整数 n , 输出 $n!$ 有几位 ($n \leq 10000$, $n! = 1*2*3*...*n$)

样例输入:

10

样例输出:

7 (注: $10! = 3628800$, 共7位)

(提示: 1.如果你会高精度运算, 可以强行把结果算出来。但这不是最好的方法

2.C++使用对数需要引入数学函数库 `#include <math.h>`)

扩展阅读：引用传参

```
12 #include <iostream>
13
14 using namespace std;
15
16 void swap1(int a,int b,int arr[]) { // 形式参数/值参
17     int tmp=a; a=b; b=tmp;
18     tmp=arr[0]; arr[0]=arr[1]; arr[1]=tmp;
19     cout<<"in swap1:\ta="<<a<<" b="<<b<<" arr=["<<arr[0]<<","<<arr[1]<<"]<<endl;
20 }
21
22 void swap2(int &a,int &b,int arr[]) { // 引用参数/变量参
23     int tmp=a; a=b; b=tmp;
24     tmp=arr[0]; arr[0]=arr[1]; arr[1]=tmp;
25     cout<<"in swap2:\ta="<<a<<" b="<<b<<" arr=["<<arr[0]<<","<<arr[1]<<"]<<endl;
26 }
27
28 void paramDemo(){
29     int a=1,b=2;
30     int arr[2]={3,4};
31     cout<<"outside:\ta="<<a<<" b="<<b<<" arr=["<<arr[0]<<","<<arr[1]<<"]<<endl;
32
33     swap1(a,b,arr);
34     cout<<"outside:\ta="<<a<<" b="<<b<<" arr=["<<arr[0]<<","<<arr[1]<<"]<<endl;
35     swap2(a,b,arr);
36     cout<<"outside:\ta="<<a<<" b="<<b<<" arr=["<<arr[0]<<","<<arr[1]<<"]<<endl;
37 }
38
```

```
outside:    a=1 b=2 arr=[3,4]
in swap1:   a=2 b=1 arr=[4,3]
outside:    a=1 b=2 arr=[4,3]
in swap2:   a=2 b=1 arr=[3,4]
outside:    a=2 b=1 arr=[3,4]
Program ended with exit code: 0
```

→ 学过Pascal的知道Pascal有值参和变量参两种传参方式

→ c++的基本类型传参默认是值参形式，也就是函数内改变参数的值，不会影响到函数外

→ 如希望函数内参数的改变传递到外面，可以在参数声明前加&符号，称为引用传参

→ 为什么数组不需要引用传参？这涉及指针相关知识（引用本质上也是指针）。后面会讲到指针，现在你只需要记下来就行了

扩展阅读：模板

```
93  template<class T>
94  T add(T a,T b){
95      return a+b;
96  }
97
98  void templdateDemo(){
99      std::cout<<add(1,2)<<std::endl;
100     std::cout<<add(1.1,2.2)<<std::endl;
101 }
102
```

3
3.3

→ **vector**类型后有一个尖括号，这个语法称为模板（Template，有些语言称为泛型，generic type）

→ 模板表示一种抽象的类型，用以简化对不同类型的相同操作。例如整数可以比大小，实数也可以比大小，使用泛型你就不需要写两个不同的比大小函数