|  |
| --- |
|  |
| 1. 每天必须做的事情，非常重要的事情（需要时间来积累的东西。）   数据结构和算法。排序什么的，每天最少看他几遍，休息一下的时候，就开始看看。每一个都要很快的会写。需要很熟练。  设计模式，这个只有熟稔于心，才能真的用的到自己的代码设计中。  每天收集前一天的面经。  这样整个面试就还差一个项目，能够说出深入了解什么，知道什么，做过什么，方法论。就无敌了。  2.每天就这样做，就是面试的王道和主要矛盾。熟练是最重要的技能，达到熟练的举重若轻就是不断成功的做好。把简单的事情做熟练。就nice。每天的成功书写，和一次不断突破的效果我想绝对是不同的。 |
| 3，个人品牌经营：就是每天把自己的代码提交上去，这个绝对是捷径。还有把自己读书笔记提交上去，非常重要，也是对自己的一个交代，总结是一种再学习。  4.复习：面经上提到过的，但是不能清晰说出来的知识。  5.广度：面经上提到过的，但是还是不清楚的。 |
|  |
| 新知识，就是机器学习了。这段时间的重点。  参加一次比赛。还有kmp串的匹配算法，很重要。 |

|  |
| --- |
| 每天练习排序的代码，这样1，能够训练自己把想法很容易的用计算机实现，2，也是以后用来解决其他问题的范例。3，能够训练快速写代码，每次应该是让自己顺着自己的思路写，这样能够，让自己快速让思维和代码对应，这个是很重要，也是别人很看重的能力。4，快速写代码，调整代码质量。就想交换一样。形成这样的块。4.有这些基础能力，相对来说，就是**缩短和简化自己以后写其他代码的时间和思考的强度**，这个很**经典**。  坚持，很重要哦。 |
| 下午，完成了堆排序的代码和复习。复习了8大排序，明天再把桶排序搞定，就每天复习一下就可以了。接下来就是红黑树，还有字符串匹配，很经典 |
| 今天的突破，跟随了androw W。吴恩达的视频，看到了第五章。英语听课毫无压力。 |
| 学会用Octave了，绘制了第一张图（还保存下来了，非常的方便。），学会了计算矩阵等。 |
|  |

|  |
| --- |
| 签到8.29， |
| //归并排序--二路归并。  //其实有思路，然后去做很简单。关键是先把思路完全熟悉了。这样去写，然后知道其中应该注意的地方，信手拈来。在其他地方可以当一个小技巧，随便使用。  //所有的方法都应该做成util的样子。  **public** **static** **void** mergeSort(**int**[] a){  **int**[] swap = **new** **int**[a.length];  **int** k = 1;  **while**(k<a.length){  *merge*(a, swap, k);  //将元素从零时数组swap中放回到数组a中。  **for** (**int** i = 0; i < a.length; i++)  a[i] = swap[i];  //调整每个子数组的长度。  k \*= 2;  }  }  **public** **static** **void** merge(**int**[] a , **int**[] swap ,**int** k){  **int** m= 0,l1 =0,u1,l2,u2;  **int** i,j;  **while**(l1+k <= a.length - 1){  l2 = l1+k;//第二个有序子数组的下界low  u1 = l2 -1;//第一个有序子数组的上界up  u2 = (l2 + k -1 <= a.length -1)? l2+k-1 : a.length -1;//1第二个有序子数组的上界low  //2从两个子数组找小的元素，插入到我们的新数组中，直到某一个子数组已经找完，那么另外一个没有找完的子数组剩下的元素就比已经找完的这个要大。把剩下的全部遍历完就可以了。  **for**(i = l1,j = l2;i<=u1&&j<=u2;m++){  **if**(a[i]<=a[j]){//稳定  swap[m] = a[i++];  }**else**{  swap[m] = a[j++];  }  }  **while**(i<=u1) swap[m++] = a[i++];//如果上一个for循环，是j>u2跳出的会执行这段代码（说明子数组2已经归并完成），将子数组1中剩余的元素放入到数组swap中。  **while**(j<=u2) swap[m++] = a[j++];//如果上一个for循环，是1>u1跳出的会执行这段代码（说明子数组1已经归并完成），将子数组2中剩余的元素放入到数组swap中。  l1 = u2 +1;//3第三个有序数组的上界限。非常经典。  }  //4将原始数组中，只够一组的数据元素，顺序的放到数组swap中。  **for**(i = l1;i<a.length;i++,m++) swap[m] = a[i];  } |
|  |

|  |
| --- |
|  |
| 8.28的面经，要写哦。 |
|  |

|  |
| --- |
|  |
| **堆排序** |
| /\*\*  \* 堆排序实现：经典。  \*/  **public** **static** **void** heapSort(**int**[] a){  //1.把最大堆的堆顶元素和当前最大堆的最后一个元素交换  //2.最大堆元素个数减1  //3.调整根节点，使之继续满足最大堆的定义。  **int** temp;  *initCreateHeap*(a);  **for** (**int** i = a.length - 1; i > 0; i--) {  temp = a[0];  a[0] = a[i];  a[i] = temp;  *createHeap*(a, i,0);//这里i全部是堆的元素个数。  }  }  **public** **static** **void** initCreateHeap(**int**[] a){  //左右孩子都是符合最大堆的约定。  //原始状态第一个可能不符合最大堆定义的是第一个非叶子节点。从这个位置出发，依次创建最大堆。  **for**(**int** i = (a.length-1)/2;i >= 0;i--){  *createHeap*(a,a.length, i);  }  }  **public** **static** **void** createHeap(**int**[] a,**int** n , **int** h){  **int** i = h ,j = 2\*h + 1;  **int** flag = 0;  **int** temp = a[h];  //找出左右孩子中较大者，重复向下筛选。  **while**(j<n && flag != 1){  **if**(h < n-1 && a[j] < a[j+1]) j++;//找到左右孩子中较大者。  **if**(temp > a[j]){  flag = 1;//如果比左右孩子中最大者都大，那么就标记结束筛选条件，跳出循环。  }**else** {  a[i] = a[j];//这里就是a[i],不能是a[h]。把子树中的最大者放到i位置。因为这样处理之后，如果直接把temp放入a[j],可能子树就不符合最大堆了。所以还要找到合适的位置放入才行。  i = j;  j = 2\*i +1;//让子树执行同样的操作，循环。  }  }  a[i] = temp;//把等待放入的元素，放入到它应该放入的位置中。  } |
|  |
|  |

Achievement：

