1. 编程题
2. **求解第k小的值**

**提示：**

**p=partition(a,l,h)**

**If (p==k-1) //此时p位置对应的值就是第K小的值**

**核心代码参考：**

**int partition(int a[], int low, int high) {**

**int pivot = a[low],temp;**

**int last\_small = low;**

**for (int i = low+1; i <= high; i++) {**

**if (a[i] < pivot) {**

**last\_small++;**

**swap(a, i, last\_small);**

**}**

**}**

**swap(a, low, last\_small);**

**return last\_small;**

**}**

**int findkth(int a[], int low, int high, int k) {**

**if (low <= high) {**

**int pivot = partition(a, low, high);**

**if (k == pivot+1)**

**return a[k-1];**

**else if (k > pivot+1 )**

**return findkth(a, pivot + 1, high, k);**

**else**

**return findkth(a, low, pivot - 1, k);**

**}**

**}**

1. **链表下的插入排序**

**核心参考代码**

**template <typename DataType>**

**void LinkList<DataType>::insertion\_sort() {**

**Node<DataType>\* temp = NULL, \* first\_unsorted, \* last\_sorted, \* p;**

**if (first->next == NULL)**

**return;**

**first->data = INT16\_MIN;**

**first\_unsorted = first->next;**

**last\_sorted = first;**

**while (first\_unsorted != NULL) {**

**if (first\_unsorted->data >= last\_sorted->data) {**

**last\_sorted = last\_sorted->next;**

**first\_unsorted = first\_unsorted->next;**

**}**

**else {**

**p = first;**

**while (p->next->data <= first\_unsorted->data)**

**p = p->next;**

**last\_sorted->next = first\_unsorted->next;**

**first\_unsorted->next = p->next;**

**p->next = first\_unsorted;**

**first\_unsorted = last\_sorted->next;**

**}**

**}**

**}**

**二 填空题**

**在对一组记录（54，38，96，23，15，72，60，45，83）进行直接插入排序时，当把第7个记录60插入到有序表时，为寻找插入位置需要比较 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 次。**

**答案: 3**

**排到60，前面都应该有序，可以快速把前面的序列写出来，然后进行比较次数**

**15，23，38，54，72，96，| 60，45，83**

**三 简答题**

1. **快速排序最多最少比较次数**

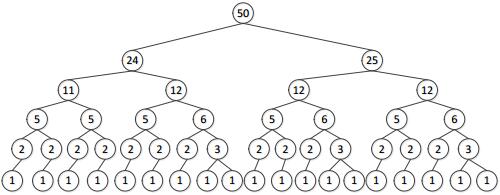
**对50个整数进行快速排序，需进行的关键码之间的比较次数可能达到的最大值和最小值分别是多少？给出分析求解过程。**

**参考答案：**

**最大比较次数：在关键字有序的情况下，为**

**49+48+...+1=1225**

**最小比较次数：判定树如下图所示（树中结点的所示数字为此子序列所含关键字个数）**

****

**最少比较次数：49+47+43+35+19=193.**

**也可以用类似下面公式推导**

**F(n)=f(ceil[(n-1)/2)]+f(n-1-ceil((n-1)/2))+1**

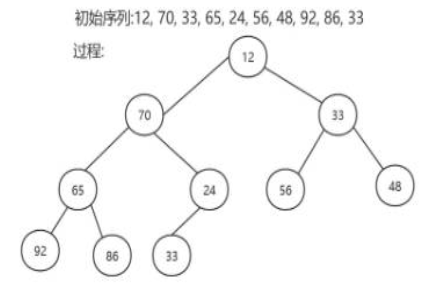
**F(50)=f(24)+f(25)+1**

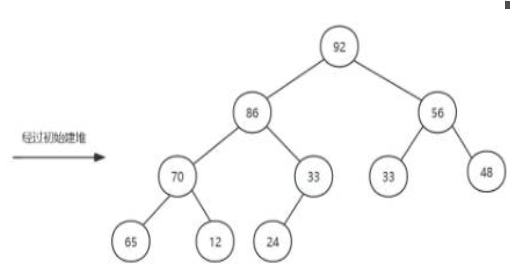
**=f(11)+f(12)+f(12)+f(12)+3**

**=f(5)+f(5)+（f(5)+f(6)）\*3+7**

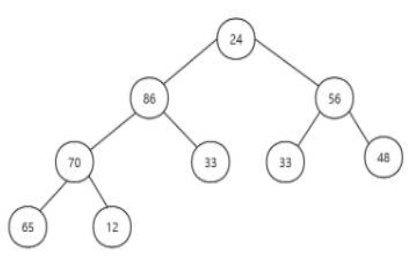
**=……..**

1. **堆排序获取3个最大值**

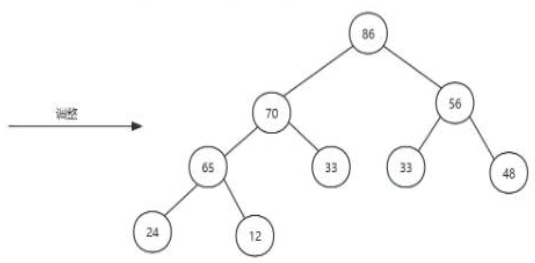
****

****

**第一个取出92之后将最后一个元素24放置堆顶**

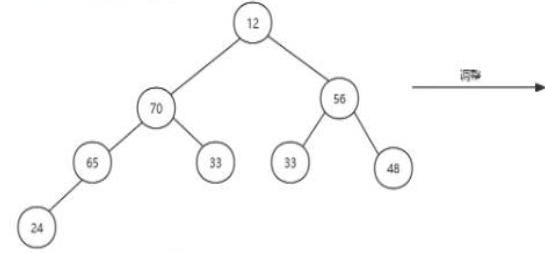
****

**进一步调整**

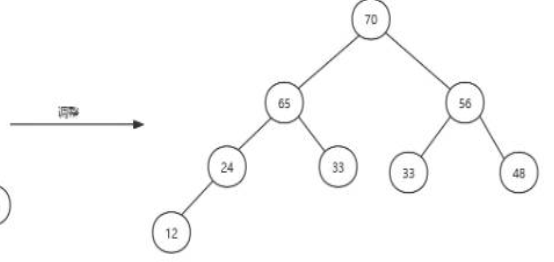
****

第一个取出92

**第2个取出堆顶86，用12替代后得到**

****

**进一步调整获得**

****

**第3个取出70**

1. **链表下的递归归并排序**

第2个取出86

**void LinkList<DataType> ::merge\_sort(); //归并排序主调函数**

**void recursive\_merge\_sort(Node<DataType> \* &sub\_list);//递归归并排序  
Node<DataType> \*divide\_from(Node<DataType> \*sub\_list);//分割链表，返回后半部分的头指针  
Node<DataType> \*merge(Node<DataType> \*pa, Node<DataType> \*pb);//合并两个有序链表**

**核心参考代码：**

**template<typename DataType>  
void LinkList<DataType> ::merge\_sort(){  
    recursive\_merge\_sort(head);  
}**

**template<typename DataType>  
void LinkList<DataType> ::recursive\_merge\_sort(Node<DataType> \* &sub\_list){  
    if(!sub\_list || !sub\_list->next)return;//链表为空或只有一个节点，无需排序  
    Node<DataType>\* second\_half = divide\_from(sub\_list);// 将链表均分成两半  
    recursive\_merge\_sort(sub\_list);  
    recursive\_merge\_sort(second\_half);  
    sub\_list = merge(sub\_list,second\_half);// 合并两部分为有序链表  
}**

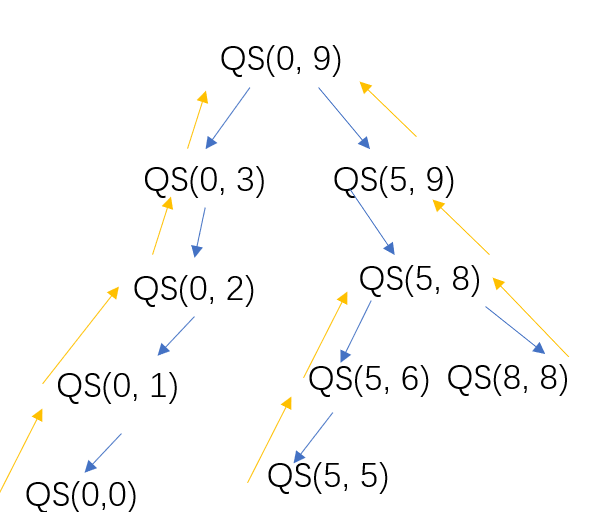
**template<typename DataType>  
Node<DataType> \*LinkList<DataType> ::divide\_from(Node<DataType> \*sub\_list){  
    if(!sub\_list || !sub\_list->next)return nullptr;  
    Node<DataType>\* slow = sub\_list;  
    Node<DataType>\* fast = sub\_list->next;  
    while (fast && fast->next){//找到链表的中间位置  
        slow = slow->next;//快指针每次走两步，慢指针每次走一步，当快指针到达链表末尾时，慢指针恰好指向链表的中间节点。  
        fast = fast->next->next;  
    }  
    Node<DataType>\* second\_half = slow->next;  
    slow->next = nullptr;// 分割链表  
    return second\_half;  
}**

**template<typename DataType>  
Node<DataType> \* LinkList<DataType>::merge(Node<DataType> \*pa, Node<DataType> \*pb){  
    if(!pa)return pb;  
    if(!pb)return pa;  
    Node<DataType>\* dummy = new Node<DataType>(0);// 虚拟头节点  
    Node<DataType>\* current = dummy;  
    while(pa && pb){//按顺序将较小的节点添加到合并链表中  
        if(pa->data <= pb->data){  
            current->next = pa;  
            pa = pa->next;  
        }else{  
            current->next = pb;  
            pb = pb->next;  
        }  
        current = current->next;  
    }  
    current->next = pa? pa:pb;// 将剩余节点加入  
    Node<DataType>\* result = dummy->next;  
    delete dummy;// 删除虚拟头节点  
    return result;  
}**

1. **快速排序的递归调用树**

**注：单向快速排序，一般以最后一个元素为轴值，设置两个类似指针向量，i和j，i和j一直向右前进，i指向小于轴值的区域边界。**

****

****

**四． 多选题**

**用某种排序方法对线性表（25，84，21，47，15，27，68，35，20）进行排序时，元素序列的若干中间状态（按先后出现次序排列，但不一定是连续的）如下：  
（1） 25，84，21，47，15，27，68，35，20  
（2） 20，15，21，25，47，27，68，35，84  
（3） 15，20，21，25，35，27，47，68，84  
（4） 15，20，21，25，27，35，47，68，84  
则所采用的排序方法是： \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**A. 选择排序 B. 希尔排序 C. 归并排序 D. 快速排序**

**正确答案: D**