1.堆排、冒泡、插入、快速排序中哪一种可能会在最后一趟开始前，所有元素都不在其最终位置上

答：插入排序

2.直接插入排序、冒泡排序、快速排序、堆排序、归并排序中，在最好情况下哪些方法最快？在平均情况下哪些方法最快？在最坏情况下哪些最快？

排序方式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 排序方法 | 最好情况 | 平均情况 | 最坏情况 | 空间复杂度 | 稳定度 |
| 插入排序 |  |  |  |  | 稳定 |
| 冒泡排序 |  |  |  |  | 稳定 |
| 快速排序 |  |  |  |  | 不稳定 |
| 堆排序 |  |  |  |  | 不稳定 |
| 归并排序 |  |  |  |  | 稳定 |

3. 设计快速排序的非递归算法

template<class Item>

void interval(stack<int> &s,int a, int b){

s.push(b);

s.push(a);

}

template <class Item>

void QuickSortBU(Item a[], int l, int r) {

stack<int> s;

int i;

push\_interval(s, l, r);

while (!s.empty()) {

l = s.top(); s.pop();

r = s.top(); s.pop();

if (r <= l) continue;

i = partition(a, l, r);

if (i - l > r - i) {

push\_interval(s, l, i - 1);

push\_interval(s, i + 1, r);

}

else {

push\_interval(s, i + 1, r);

push\_interval(s, l, i - 1);

}

}

}

4. 对已经整理为最小化堆的数组int a[n]，要求删除其中的任一个a[i]后，重新整理为一个最小堆，其中0<i=<n，要求采用一个函数完成调整，时间复杂度为O(logn)

void siftdown(int a[], int i, int n) {

int j;

a[i]=a[n];

t = a[i];

while ((j = 2 \* i + 1) < n) {

if (j<n - 1 && a[j] > a[j + 1]) j++;

if (t>a[j]) { a[i] = a[j]; i = j; }

else break;

}

a[i] = t;

}

5. 在希尔排序、冒泡排序、直接插入排序、简单选择排序、快速排序、归并排序、堆排序和基数排序中，时间复杂度不受数据初始状态影响的是哪些方法？

答：冒泡排序、直接插入排序、简单选择排序、归并排序、堆排序、基数排序

6. n个不同的英文单词，长度均为m，若n=50，m<5，用什么排序时间性能最好？为什么?

答：基数排序。理由：该算法具有线性时间复杂度，对整数与字符串有很高效率。

7. 接近有序的顺序表用希尔、快排、堆排、冒泡哪种最好？请说明理由。

答：希尔排序。接近有序的顺序表里，冒泡排序时间复杂度为，堆排序时间复杂度为，快速排序时间复杂度为，希尔排序时间复杂度为，故应该选择希尔排序时间复杂度最小。