1. 在[二叉排序树](https://so.csdn.net/so/search?q=%E4%BA%8C%E5%8F%89%E6%8E%92%E5%BA%8F%E6%A0%91&spm=1001.2101.3001.7020" \t "_blank)中，它每个结点的值均大于其左子树上所有结点的值，小于其右子树上所有结点的值。而在堆中，根节点小于（或者大于）左右子树结点的值，但是左右字树之间的大小顺序并不确定。功能上看，二叉排序树主要用于查找操作，堆主要用于排序操作。
2. 最好是使用选择排序的方法，这样便于选择出前k个元素的排列，时间复杂度为O(n^2)

3.

a. 基本设计思想：只需要找到中位数，以中位数为界，将小于中位数的部分归为一个集合，大于等于中位数包括中位数在内的归为另外一个集合。按照快速排序的思想，每次找到枢轴所在位置，将集合一分为二。设枢轴所在最终位置为i,若 i= n/2，则已找到；若i < n/2，则在枢轴后半部分找；若i > n/2，则在前半部分找。

b. 算法描述：

#include <iostream>

using namespace std;

#define MaxSize 20000

/\*一趟划分\*/

int partition(int R[], int low, int high)

{

int temp = R[low]; //以R[low]为基准

while (low<high) //从两端交替向中间扫描

{

while (low < high&&R[high] >= temp) //逆向查找首个小于枢轴的元素

{

high--;

}

R[low] = R[high]; //将小于枢轴的元素交换到低端

while (low < high&&R[low] <= temp)//正向查找首个大于枢轴的元素

{

low++;

}

R[high] = R[low]; //将大于枢轴的元素交换到高端

}

R[low] = temp; //枢轴元素置于正确的位置

return low; //返回枢轴元素的位置

}

/\*查找n/2，数组以n/2排序，将序列分成[0...n/2)和[n/2...n)\*/

void Divide(int arr[], int n)

{

int low = 0, high = n - 1;

int mid = -1; //[0...mid) [mid...n)

while (low < high)

{

mid = partition(arr, low, high);

if (mid == n / 2) //循环终止条件，当mid位于数组中央时

break;

else if (mid < n / 2) //在右半部分查找

{

low = mid + 1;

}

else //在左半部分查找

{

high = mid - 1;

}

}

}

/\*输出数组a[start...end]\*/

void OutputArray(int a[], int start, int end)

{

int i;

for (i = start; i <= end - 1; i++)

{

printf("%d ", a[i]);

}

printf("%d\n", a[i]);

}

int main()

{

int n = 0;

int a[MaxSize];

cout << "请输入由正整数构成的集合元素总数： ";

cin >> n;

cout << "请依次输入集合元素： ";

for (int i = 0; i < n; i++)

{

cin >> a[i];

}

Divide(a, n);

cout << "\n划分子集A1：";

OutputArray(a, 0, n / 2 - 1);

cout << " A2：";

OutputArray(a, n / 2, n - 1);

}

c. 时间复杂度和空间复杂度：

时间复杂度：遍历数组的时间复杂度为O(n)

空间复杂度：空间复杂度为O(1)