二叉树给出前序/后序结果还原

前序：注意两个字母前面跟符号的开始画 求值从右边开始扫描 数字入栈 符号两个数字出栈

后序：注意两个字母后面跟符号的开始画 求值从左边开始扫描 数字入栈 符号两个数字出栈

前序后序能不能确定一棵树 前序存在紧紧相连的XY 后序YX 就不行 不然就可以

最小生成树

Prim算法 在接触的边里找权值最小的边，不会成环就加进来 O（n^2) 稠密图

Kruskal算法 整个图里选小边 依次判断有用没用 O（elog2e) 稀疏图

拓扑排序 若有向边<vi,vj> 则vi必须在vj前面 不能有回路 可以检测回路

全都不同的数 出栈顺序 二叉搜索树种类 C2n取n除以n+1 或者fn=fi+f（n-1-i)求和 catalan公式

最短路径

Dijkstra算法 一个一个加顶点进来 列表 类似离散数学

Floyd算法 一个一个考虑顶点

identifier density 标志符密度 有几种哈希值/size

loading density 存了几种哈希值/size

radix sort 基数排序 Least Significant Digit 低位优先（先看个位排）Most（先看高位）比如个位都一样那就原顺序不变

```
#include <stdio.h>

void InsertionSort(int a[], int total)
{
    int i, j;
    for (i = 1; i < total; i++)
    {
        if (a[i] < a[i - 1])
        {
            int temp = a[i];
            for (j = i; j > 0; j--)
            {
                if (a[j - 1] > temp)
                    a[j] = a[j - 1];
                else
                    break;
            }
            a[j] = temp;
        }
    }
```

```
}

int Partition(int a[], int low, int high)
{
    int key = a[low];
    while (low < high)
    {
        while (low < high && a[high] >= key)
        {
            --high;
        }
        a[low] = a[high];
        while (low < high && a[low] <= key)
        {
            ++low;
        }
        a[high] = a[low];
    }
    a[low] = key;
    return low;
}

void QuickSort(int a[], int low, int high)
{
    int pivotloc;
    if (low < high)
    {
        pivotloc = Partition(a, low, high);
        QuickSort(a, low, pivotloc - 1);
        QuickSort(a, pivotloc + 1, high);
    }
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void maxHeap(int a[], int start, int end)
{
    int dad = start;
    int son = dad * 2 + 1;
    while (son <= end)
    {
        if (son + 1 <= end && a[son] < a[son + 1])
            son++;
        if (a[dad] > a[son])
            return;
        else
        {
            swap(&a[son], &a[dad]);
            dad = son;
            son = dad * 2 + 1;
```

```
        }
    }
}
void HeapSort(int a[], int n)
{
    int i;
    for (i = n / 2 - 1; i >= 0; i--)
    {
        maxHeap(a, i, n - 1);
    }
    for (i = n - 1; i > 0; i--)
    {
        swap(&a[0], &a[i]);
        maxHeap(a, 0, i - 1);
    }
}

int main()
{
    int a[10] = {2, 3, 5, 1, 6, 9, 7, 8, 4, 0};
    InsertionSort(a, 10);
    int i;
    printf("InsertionSort:");
    for (i = 0; i <= 9; i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
    QuickSort(a, 0, 9);
    int i;
    printf("QuickSort:");
    for (i = 0; i <= 9; i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
    HeapSort(a, 10);
    int i;
    printf("HeapSort:");
    for (i = 0; i <= 9; i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```