算法阅读题 2024-2025-1

1. 二分查找

二分查找法。按照从小到大的顺序，输入 n 个整数并存入数组 a 中，然后在数组 a 中查找给定的 x。如果数组 a 中的元素与 x 的值相同，输出相应的下标（下标从 0 开始）；如果没有找到，输出“Not Found”。如果输入的 n 个整数没有按照从小到大的顺序排列，或者出现了相同的数，则输出“Invalid Value”。

```
#include <stdio.h>
# define MAXN 10
int main()
{
    int found, i, left, mid, n, right, sorted, x;
    int a[MAXN];

    scanf("%d %d", &n, &x);
    for(i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }

    sorted = 1;
    for(i = 1; i < n; i++){
        if (a[i] <= a[i-1]) 1 分 {
            sorted = 0;  1 分
            break;
        }
    }
    if(sorted == 0){
        printf("Invalid Value\n");
    }else{
        found = 0;
        left = 0; right = n - 1;  1 分
        while(left <= right){
            mid = (left + right) / 2;  1 分
            if (x == a[mid]){
                found = 1;  1 分
                break;
            }else if (x < a[mid]){
                right = mid - 1;  1 分
            }else{
                left = mid + 1;  1 分
            }
        }
        if(found != 0){
            printf("%d\n",mid);
        }
        else{
            printf( "Not Found\n");
        }
    }

    return 0;
}
```

2. 计算二叉树深度。

```
#include<iostream>
using namespace std;

typedef struct BiNode
{
    char data;
    struct BiNode *lchild,*rchild;
}BiTNode,*BiTree;


void CreateBiTree(BiTree &T)
{
    char ch;
    cin >> ch;
    if(ch=='#')  T=NULL;
    else{
        T=new BiTNode;
        T->data=ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }
}

int Depth(BiTree T)
{
    int m,n;
    if( T == NULL  2 分 ) return 0;
    else
    {
        m=Depth(T->lchild)  2 分 ;
        n=Depth(T->rchild)  2 分 ;
        if(m>n) return(m+1);
        else return (n+1);
    }
}

int main()
{
    BiTree tree;
    CreateBiTree(tree);
    cout<<Depth(tree);
    return 0;

}
```

3. 使用栈和队列进行回文判断

```
int main()
{
    DataType ch;
    int flag;
    LinkStack stack_pal = SetNullStack_Link();
    LinkQueue queue_pal = SetNullQueue_Link();
    ch = getchar();
    while (ch != '#')
    {
        Push_link(stack_pal, ch);
        EnQueue_link(queue_pal, ch);
        ch = getchar();
    }
    flag = 1;
    while ( (!IsNullStack_link(stack_pal)) &&(!IsNullQueue_link(queue_pal))   3 分 )
    {
        if ( Top_link(stack_pal) != FrontQueue_link(queue_pal)   2 分 )
        {
            flag = 0;
            break;
        }
        else
        {
                Pop_link(sta  2 分 ;
            DeQueue_link(queue_pal)   2 分 ;
        }
    }
    if ( flag|flag==1   1 分 )
        printf("this is palindromic");
    else
        printf("this is NOT palindromic");
    return 0;
}
```

4. 线索二叉树中序线索化及遍历。

```
#include<iostream>
using namespace std;

typedef struct BiThrNode
{
    char data;
    struct BiThrNode *lchild,*rchild;
    int LTag,RTag;
}BiThrNode,*BiThrTree;

BiThrNode *pre=new BiThrNode;

void CreateBiTree(BiThrTree &T)
{
    char ch;
    cin >> ch;
    if(ch=='#')  T=NULL;
    else
    {
        T=new BiThrNode;
        T->data=ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }
}

void InThreading(BiThrTree p)
{
 if(p)
  {
   InThreading(p->lchild);
   if(!p->lchild)
   {
      p->LTag=1   2 分 ;
      p->lchild=pre   2 分 ;
   }
   else
     p->LTag=0;
   if(!pre->rchild)
    {
      pre->RTag=1   2 分 ;
      pre->rchild=p   2 分 ;
    }
   else
     pre->RTag=0;
    pre=p   2 分 ;
   InThreading(p->rchild);
   }
}
```

```
void InOrderTraverse_Thr(BiThrTree T)
{
   BiThrTree p;
   p=T;
   while(p)
   {
     while(p->LTag==0)
        p=p->lchild  2 分 ;
     cout<<p->data;
     while(p->RTag==1)
     {
        p=p->rchild  2 分 ;
       cout<<p->data;
     }
      p=p->rchild  2 分 ;
   }
}

int main()
{
    pre->RTag=1;
    pre->rchild=NULL;
    BiThrTree tree;
    CreateBiTree(tree);
    InThreading(tree);
    InOrderTraverse_Thr(tree);
    return 0;
}
```

5. 三元组顺序表表示的稀疏矩阵转置

三元组顺序表表示的稀疏矩阵转置Ⅱ。设 a 和 b 为三元组顺序表变量，分别表示矩阵 M 和 T。要求按照 a 中三元组的次序进行转置，并将转置后的三元组置入 b 中恰当的位置。

输入格式:

输入第 1 行为矩阵行数 m、列数 n 及非零元素个数 t。

按行优先顺序依次输入 t 行，每行 3 个数，分别表示非零元素的行标、列标和值。

输出格式:

输出转置后的三元组顺序表结果，每行输出非零元素的行标、列标和值，行标、列标和值之间用空格分隔，共 t 行。

输入样例 1:

3 4 3

0 1 -5

1 0 1

2 2 2

输出样例 1:

0 1 1

1 0 -5

2 2 2

```
#include <stdio.h>
#include <stdlib.h>
#define M 100
struct node{
    int i,j,v;
};

struct tripletable
{
    struct node S[M];
    int m,n,t;
};


struct tripletable * create()
{    int i;
    struct tripletable *head=(struct tripletable *)malloc(sizeof(struct tripletable));
    scanf("%d%d%d",&(head->m),&(head->n),&(head->t));

    for(i=0;i<head->t;i++)
        scanf("%d%d%d",&(head->S[i].i),&(head->S[i].j),&(head->S[i].v));
    return head;

}


void print(struct tripletable * head)
{
    int i;
    for(i=0;i<head->t;i++)
    printf("%d %d %d\n",(head->S[i].i),(head->S[i].j),(head->S[i].v));
}

struct tripletable * trans(struct tripletable *t1)
{
    int i,p,j,q,k;
    int num[100];
    int cpot[100];
    struct tripletable *t2=(struct tripletable *)malloc(sizeof(struct tripletable));
    t2->m=t1->n;t2->n=t1->m;t2->t=t1->t;
    if(t1->t) {
        for(i=0;i<t1->n;i++) num[i]=0;
        for(i=0;i<t1->t;i++) {k= t1->S[i].j  2 分 ;++num[k];}
        cpot[0]=0;
        for(i=1;i<t1->n;i++) cpot[i]= cpot[i-1]+num[i-1]  2 分 ;
        for(p=0;p<t1->t;p++){
            j=t1->S[p].j;  q=cpot[j]  2 分 ;
            t2->S[q].i=t1->S[p].j;t2->S[q].j=t1->S[p].i;
            t2->S[q].v=t1->S[p].v;
             ++cpot[j]  2 分 ;
        }

    }
    return t2;
}

int main()
{
    struct tripletable * head,*t2;
    head=create();
    t2=trans(head);
    print(t2);
    return 0;
}
```

6. 二叉树的层次序遍历

```
#include <stdio.h>
#include <stdlib.h>
#include <queue>
#include <stack>
using namespace std;

struct TreeNode
{
    char data;
    TreeNode* left;
    TreeNode* right;
};

void levelOrder(struct TreeNode* root)
{
    queue<struct TreeNode*> que;
    if (root == NULL)return;
    que.push(root);
    while (!que.empty()) {
        struct TreeNode* tmp = que.front();
        que.pop();
        printf(" %c", tmp->data);
        if (tmp->left)
            que.push(tmp->left)  5 分 ;
        if ( tmp->right  5 分 )
            que.push(tmp->right)  5 分 ;
    }
}
```

7. 希尔排序

```
void ShellInsert(SqList &L,int dk)
{
   int i,j;
   for(i=dk+1;i<=L.length;++i)
      if( L.r[i].key<L.r[i-dk].key  2 分 )
        {
          L.r[0]=L.r[i];
          for(j=i-dk; j>0&& L.r[0].key<L.r[j].key  2 分 ;j-=dk)
                L.r[j+dk]=L.r[j]  2 分 ;
          L.r[j+dk]=L.r[0];
        }
}

void ShellSort(SqList &L,int dt[ ],int t){
  int k;
  for(k=0;k<t;++k)
    ShellInsert(L,dt[k]);
}
```

8. 快速排序

```
#include <stdio.h>
#define MAXSIZE 100

typedef struct
{
 int elem[MAXSIZE];
 int  length;
}SqList;

int Partition(SqList &L,int low,int high)
{
  int pivotkey;
  pivotkey=L.elem[low];
  while(low<high)
  {
      while(low<high && L.elem[high]>=pivotkey)
          high--                2 分;
      L.elem[low]=L.elem[high];
      while(low<high && L.elem[low]<=pivotkey)
          low++;
      L.elem[high]=L.elem[low];
  }
  L.elem[low]=pivotkey;//枢轴放入下标为low的位置
  return  low;
}

void QSort(SqList &L,int low,int high)
{
  int pivotloc;
  if(low<high)
   {
       pivotloc=Partition(L,low,high);
       QSort(L,low,pivotloc-1)     2 分;//对pivotloc以左排序
       QSort(L,pivotloc+1,high);
  }
}

void QuickSort(SqList &L)
{
   QSort(L,0,L.length-1);
}

void Create_Sq(SqList &L)
{
    int i,n;
    L.length=0;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        scanf("%d",&L.elem[i]);
        L.length++;
    }
}
```