# 习题2 表结构

2-1 如果长度为n的线性表采用顺序存储结构存储，则在第i (1≤i≤n+1)个位置插入一个新元素的算法的时间复杂度为( B )。

(A) O(1)

(B) O(n)

(C) O(nlog2n)

(D) O(n2)

2-2 在一个有127个元素的顺序表中插入一个新元素，要求保持顺序表元素的原有(相对)顺序不变，则平均要移动( C )个元素。

(A) 7

(B) 32

(C) 64

(D) 127

2-3 设顺序表的长度为30，平均分成5块，每块6个元素。如果采用分块查找，则其平均查找长度为(    C )。

(A) 5

(B) 5.7

(C) 6.5

(D) 8.2

2-4 已知An×n为稀疏矩阵。试从时间和空间角度比较，采用二维数组和三元组顺序表两种存储结构计算∑aij的优缺点。

二维数组：时间复杂度为O(1),空间复杂度为O（n²）。

三元组顺序表:设非零元素为m个，时间复杂度为O（m），空间复杂度为O（3\*m）

三元组顺序表虽然节省空间，但查询一个数据的时候，时间复杂度较高，所以三元组顺序表只在非零元素很少的情况下使用。

2-5 综合比较顺序表和链表。

顺序表所开辟的空间是连续的，便于查询，但插入和删除需要大量的节点移动，并不方便。

链表每个结点所在空间不一定连续，查询数据需要遍历整个链表，较为耗时。但易于插入和删除元素。

2-6 解释链表的“头指针、头结点和首元素结点”三个概念。

头指针：不存放数据，存放头结点的地址，是链表的开端。

头结点：在首元素结点之前的一个结点，其数据域无意义。

首元素结点：数据域存放第一个元素的数据

2-7 设链表L→a→b→c→d，指针域为\*next。执行下列命令后，(    B )。

p=L->next->next;

L->next->next=NULL;

q=L->next->next;

(A) p→b→c→d，q→a

(B) p→b→c→d，q→NULL

(C) p→c→d，q→a

(D) p→c→d，q→a→b

2-8 描述下列算法的主要功能是(   A )。

① 构造头结点L，取q=L;

② 产生1个结点p;

③ q−>next=p;

④ 输入p−>data的值;

⑤ 取q=p;

⑥ 重复执行②至⑤n次;

⑦ p−>next=NULL;

(A) 通过输入n个数据元素构建链表L

(B) 采用前插法，在链表L中输入n个数据元素

(C) 通过产生n个结点构建链栈L，q为栈顶指针

(D) 在链队列L中输入n个数据元素，q为队尾指针

2-9 设两个循环链表的长度分别为n和m，则将这两个循环链表连接成一个循环链表，最好的时间复杂度为( A )。

(A) O(1)

(B) O(n)

(C) O(m)

(D) O(min(n，m))

2-10 设push和pop分别表示进栈和出栈操作，输入序列为xyz，则经过栈操作(   A )可以输出序列yzx。

(A) push, push, pop, push, pop, pop

(B) push, push, push, pop, pop, pop

(C) push, pop, push, pop, push, pop

(D) push, pop, push, push, pop, pop

2-11 设进栈序列为123，试给出所有可能的出栈序列。

123,132,213,231,321

2-12 如果进栈序列为123456，能否得到出栈序列435612和135426?

不可以得到435612和135426

2-13 简述算法的功能(设数据元素类型为int)：

void proc(LinkQueue \*Q)

{

LinkStack S;

InitStack(S);

while(!EmptyQueue(Q) )

{

DeleteQueue(Q, d);

Push(S,d);

}

while(!EmptyStack(S) )

{

Pop(S, d);

InsertQueue(Q, d);

}

}

将队列中原来的元素，逆序存放

2-14 描述下列递归算法的功能。

int F(int m, int n)

{

if (n>m) return F(n, m);

else if (n==0) return m;

else return F(n, m%n);

}

功能：求m，n的最大公约数

2-15 编写递归算法：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | 0，m=0且n≥0 |
|  | g(m, n)= |  |
|  |  | g(m-1, 2n)+n，m>0且n≥0 |
| int g(int m,int n)  {  if(n<0||m<0)  {  cout<<"Failed"<<endl;  return -1;  }  else  {  if(m==0)  return 0;  if(m>0)  return g(m-1,2\*n)+n;  }  } |  |  |  |

2-16 将下列递归过程改写为非递归过程。

void test(int &s)

{

int x;

scanf ("%d", &x);

if (x==0) s=0;

else

{

test(s);

s+=x;

}

}

非递归程序：

void test(int &s)

{

int x;

s=0;

scanf("%d",&x);

while(x!=0)

{

s+=x;

scanf("%d",&x);

}

}

2-17 按照格式要求给出调用F(3,'A','B','C')的运行结果：

void F(int n, char x, char y, char z)

{

if (n==1) printf("1 %c 🡪 %c\n", x, z);

else

{

F(n-1, x, z, y);

printf("%d %c 🡪 %c\n", n, x, z);

F(n-1, y, x, z);

}

}

1 A->C

2 A->B

1 C->B

3 A->C

1 B->A

2 B->C

1 A->C