# 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 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。

**答：1.从时间上来看，对稀疏矩阵A求和时，由于a[i][j]是随机存取的，读取速度更快；但用三元组顺序表时，因为其中存储的都是非零元素，所以需要从头到尾遍历整个三元组顺序表才能找到行列值相等的非零元素，时间上比采用二维数组更长。**

**2.从空间上来看，二维数组需要存储系数矩阵中所有零和非零元素，而三元组顺序表仅需存储非零元素以及它们对应的下标，因此三元组顺序表占用的空间比二维数组要少很多。**

2-4 综合比较顺序表和链表。

**答：1.从时间上来看，顺序表是一种随机存储结构，对表中的任意节点都可以以O(1)的时间复杂度直接存取，而链表中的特定节点则需要从头指针开始遍历整个链表才能找到。因此，若线性表的操作主要是查找，基本不做插入或删除操作时，可以采用顺序表；但如若需要频繁地进行插入或删除操作，则需要采用链表。因为链表的任意位置上进行插入删除操作只需要修改指针。**

**2.从空间上来看，顺序表的存储空间是静态分配的，在编写程序时需要对它的大小进行直接声明。如若线性表的长度会发生较大的变化，则顺序表不会是合适用来存储数据的选择，因为可能造成空间的浪费或者溢出问题。而链表的存储空间是动态分配的，只要计算机内存中有空闲，就不会导致空间溢出和浪费。因此，如果线性表的长度会发生较大变化时，采用链表的存储结构更为合适。但由于链表单个节点不仅仅存放数据，还要存放指向下一个节点的指针，因此链表的空间利用率并没有顺序表那么高。所以当线性表的变化不大时，可以事先确定其大小，适当地开大一点，然后用来存取数据，这样可以节约存储空间。**

2-5 解释链表的“头指针、头结点和首元素结点”三个概念。

**答：1.头指针：头指针是指向链表中第一个结点的指针。若链表存在头结点，则头指针指向头结点；若链表没有头结点，则头指针指向首元素结点。**

1. **头结点：头结点是链表中首元素结点之前的一个节点，头结点的指针域指向首元素结点，头结点的数据域可以不存储信息，也可以存储附加信息，如链表长度等。**
2. **首元素结点：首元素结点是链表中存储第一个数据元素的结点。**

2-6 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

2-7 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

2-8 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))

2-9 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

2-10 设进栈序列为123，试给出所有可能的出栈序列。

**答：(1).321 (2).231 (3).213 (4).132 (5).123**

2-11 如果进栈序列为123456，能否得到出栈序列435612和135426?

**答：不能得到435612，可以得到135426。**

2-12 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);

}

}

**答：利用堆栈做辅助，将队列Q中的数据元素逆置。**

2-13 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);

}

**答：辗转相除法求m与n的最大公约数。**

2-14 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 (m == 0 && n >= 0)

        return 0;

    else if (m > 0 && n >= 0)

        return g (m-1, 2\*n) + n;

    return 0;

}

2-15 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;

}

}

void add (int &s) {

    int x;

    s = 0;

    scanf ("%d", &x);

    while (x != 0) {

        s += x;

        scanf ("%d", &x);

    }

}

2-16 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);

}

}

运行结果如下：

**1 A->C**

**2 A->B**

**1 C->B**

**3 A->C**

**1 B->A**

**2 B->C**

**1 A->B**