哈尔滨工业大学（深圳）2021年春《数据结构》

第一次作业 线性结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学号 |  | 姓名 |  | 成绩 |  |

1、简答题

1.1 简述线性链表头指针,头结点,首元结点(第一个结点) 三个概念的区别。

1.2 在什么情况下用顺序表比链表好?

1.3 简述队列和栈这两种数据类型的相同点和差异处

1.4 设有如下图所示的火车车轨，入口到出口之间有n条轨道，列车的行进方向均为从左至右，列车可驶入任意一条轨道。现有编号为1～9的9列列车，驶入的次序依次是8,4,2,5,3,9,1,6,7。若期望驶出的次序依次为1至9，则n至少是多少？



1.5现有队列Q与栈S，初始时队列Q中的元素依次是1,2,3,4,5,6（1在队头），栈S为空。若仅允许下列3种操作：

① 出队并输出出队元素；

② 出队并将出队元素入栈；

③ 出栈并输出出栈元素。

请分析是否能得到1,2,5,6,4,3和3,4,5,6,1,2两个输出序列,为什么？

1.6假设按低下标优先存储整数数组A(-3:8,3:5,-4:0,0:7)时，第一个元素的字节存储地址是100，每个整数占4个字节。问：A(0,4,-2,5)的存储地址是什么？请简要说明计算方法。

2、数据结构设计及算法描述

给出一个停车场需求如下：

设停车场是一个可以停放n辆汽车的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列（大门在最南端，最先到达的第一辆车停放在车场的最北端），若车场内已停满n辆车，那么后来的车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车要离开时，在它之后进入的车辆必须先退出车场为它让路，待该辆车开出大门外，其他车辆再按原次序进入车场，每辆停放在车场的车在它离开停车场时必须按它停留的时间长短交纳费用。

试为停车场管理系统实现设计所需的数据结构；完成该管理系统需要哪些基本操作？文字描述算法基本思想。

3、算法设计

针对本部分的每一道题，要求：

1. 采用C或C++语言设计数据结构；
2. 给出算法的基本设计思想；
3. 根据设计思想，采用C或C++语言描述算法，关键之处给出注释；
4. 说明你所设计算法的时间复杂度和空间复杂度。

可用类语言描述，给出伪码，无需上级调试。

3.1己知单向链表L是一个递增有序表,试写一高效算法,删除表中值大于min且小于max的 结点(若表中有这样的结点),同时释放被删结点的空间,这里min和max 是两个给定的参数。

3.2 一个长度为L（L≥1）的升序序列S，处在第L/2个位置的数称为S的中位数。例如，若序列S1=(11, 13, 15, 17, 19)，则S1的中位数是15。两个序列的中位数是含它们所有元素的升序序列的中位数。例如，若S2=(2, 4, 6, 8, 20)，则S1和S2的中位数是11。现有两个等长升序序列A和B，试设计一个在时间和空间两方面都尽可能高效的算法，找出两个序列A和B的中位数。

3.3 用单向链表保存*m*个整数，结点的结构为：(data, next)，且|data|≤*n*（*n*为正整数）。现要求设计一个时间复杂度尽可能高效的算法，对于链表中data的绝对值相等的结点，仅保留第一次出现的结点而删除其余绝对值相等的结点。例如，若给定的单链表head如下：

-15

-15

21

head

-7

15 ∧

则删除结点后的head为：

21

head

-7 ∧

-15

3.4设有一个双向链表,每个结点中除有 pred、data和next这3个域外,还有一个访问频度域freq,在链表被启用之前,其值均初始化为零。每当在链表进行一次LocateNode(L, x)运算时,令元素值为x的结点中freq域的值加1,并调整表中结点的次序,使其按访问频度的递减序排列,以便使频繁访问的结点总是靠近表头。试写一符合上述要求的 LocateNode 运算的算法。

3.5线性表中元素存放在数组 A(1..n)中,元素是整型数。分别写出非递归和递归算法求出数组 A 中的最大和最小元素，分析时间和空间复杂度。

3.1

思想：遍历L，当p指向L中小于max且大于min的数时，删除p，当p指向L中小于等于min或大于等于max的数时，进行跳过。

时间复杂度为O(n)

typedef struct Node{

int data;

struct Node \*next;

}SNODE;/\*struct Node\*/

void Delete(SNODE \*head,int max,int min)

{

SNODE \*p,\*q;

p=head;

q=head->next;

while(p!=NULL)

{

if(p->data<=min||p->data>=max)

{

q=p;

p=p->next;

}

else

{

q->next=p->next;

free(p);

p=q->next;

}

}/\*while\*/

}/\*Delete\*/

3.2

思想：用S1设置一个标志链表，记作B，当S1，S2中的p1，p2指针前进一个节点时，B中的q指针前进两个节点，当q指针达到B的尾端时，p1p2指针正好达到S1，S2的中间，即可求得S1S2的中位数。

时间复杂度为O(n)

typedef struct Node{

int data;

struct Node \*next;

}SNODE;/\*struct Node\*/

int Getmid(SNODE \*s1,SNODE \*s2)

{

SNODE \*q,\*p1,\*p2;

q=s1->next;

p1=s1->next;

p2=s2->next;

while(q->next!=NULL)

{

p1=p1->next;

p2=p2->next;

q=q->next->next;

}/\*while\*/

return (0.5\*(p1->data+p2->data));

}/\*Getmid\*/

3.3

思想:使用辅助数组a[n]。初始设置a[n]全为0。遍历链表，设当前指向的数字为i。若a[|i|]=0,则置a[|i|]为1，指针前进一个节点；若a[|i|]=1，则删除此节点。则只需一次遍历链表即可完成操作。

时间复杂度为O(n)

typedef struct Node{

int data;

struct Node \*next;

}SNODE;/\*struct Node\*/

void Delete\_abs(SNODE \*head)

{

int a[N];

SNODE \*p,\*q;

p=head->next;

while(p->next!=NULL)

{

if(a[abs(p->data)]==0)

{

a[abs(p->data)]=1;

q=p;

p=p->next;

}/\*a[|i|]等于0，即第一次遇见|i|\*/

else

{

q->next=p->next;

free(p);

p=q->next;

}/\*a[|i|]不等于0，即前面出现过与i绝对值相同的数\*/

}

}

3.4

思想：用p遍历链表，当p指向的值等于x时，用q记录p的下一个位置，将p从原位置断开，并将p的freq与第一个指针比较，如果p的freq更小，则与下一个指针比较，直到p的freq比该指针大时，将p插入该指针前端。之后p从q的位置开始继续遍历链表。

时间复杂度为O(n^2)

typedef struct Node{

int data;

int freq;

struct Node \*next;

struct Node \*pred;

}SNODE;/\*struct Node\*/

void LocateNode(SNODE \*head,int x)

{

SNODE \*p,\*q,\*m;

p=head->next;

while(p->next!=NULL)

{

if(p->data==x)

{

p->freq+=1;

q=p->next;

p->pred->next=p->next;

p->next->pred=p->pred;

for(m=head->next;p->freq<=m->freq;m=m->next);

p->next=m;

p->pred=m->pred;

m->pred->next=p;

m->pred=p;

p=q;

}

else

{

p=p->next;

}

}

}

3.5

递归：

int MAX(int a[],int m,int n)

{

static int i=1;

if(i==n||a[i]>=MAX(a,i+1,n))

{

return a[i];

}

else

{

return MAX(a,i+1,n);

}

}

时间复杂度O(n)

空间复杂度O(n)

非递归

int MAX(int a[],int n)

{

int max;

for(i=1;i<=n;i++)

{

if(a[i]>=a[i+1])

{

max=a[i];

}

else

{

max=a[i+1];

}

}

return max;

}

时间复杂度O(n)

空间复杂度O(1)