**安徽大学2021-2022学年第2学期**

**《数据结构》期中考试试卷答案**

一、算法分析题（共30分）

1. （1）O(n) （2）O(sqrt(n)) （3）O(1)

2. （1）第一层FOR循环判断n+1次，往下执行n次，第二层FOR执行次数为(n+(n-1)+(n-2)+…+1)，第三层循环体受第一层循环和第二层循环的控制，其执行次数如下表：

          i=     1    2    3    …   n

         j=n     n    n    n    …   n

         j=n-1    n-1   n-1   n-1   …

          …     …   …   …

j=3     3    3

j=2     2    2

j=1     1

执行次数为(1+2+…+n)+(2+3+…+n)+…+n=n\*n(n+1)/2-n(n2-1)/6。

该算法的时间复杂度为O(n3)

（2）在n=5时，f(5)=55，执行过程中，输出结果为：

sum=15

sum=29

sum=41

sum=50

sum=55

3. (1) r->prior=q->prior;∥将q结点摘下，以便插入到适当位置。

(2)p->next->prior=q;∥（2）（3）将q结点插入

(3)p->next=q;

(4)r=r->next;或r=q->next;∥后移指针，再将新结点插入到适当位置。

二、简答题（每题10分，共40分）

4. （1）表的长度为5，深度为4

（2）head（tail（head（head（head（tail（tail（tail（tail（A）））））））））

5.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| J | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 字符串 | a | b | a | c | a | b | a | a | a | D |
| next | 0 | 1 | 1 | 2 | 1 | 2 | 3 | 4 | 2 | 2 |
| nextval | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 4 | 2 | 2 |

6. 设操作数栈是opnd，操作符栈是optr，对算术表达式#3-5\*2^3#求值，过程如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **步骤** | **opnd栈** | **optr栈** | **输入字符** | **主要操作** |
| 1 |  | # | 3-5\*2^3# | push(opnd, 3) |
| 2 | 3 | # | -5\*2^3# | push(optr, -) |
| 3 | 3 | # - | 5\*2^3# | push(opnd, 5) |
| 4 | 3 5 | # - | \*2^3# | push(optr, \*) |
| 5 | 3 5 | # - \* | 2^3# | push(opnd, 2) |
| 6 | 3 5 2 | # - \* | ^3# | push(optr, ^) |
| 7 | 3 5 2 | # - \* ^ | 3# | push(opnd, 3) |
| 8 | 3 5 2 3 | # - \* ^ | # | y=pop(opnd), x=pop(opnd), θ=pop(optr), push(opnd, xθy) |
| 9 | 3 40 | # - | # | y=pop(opnd), x=pop(opnd), θ=pop(optr), push(opnd, xθy) |
| 10 | -37 | # | # | y=pop(opnd), x=pop(opnd), θ=pop(optr), push(opnd, xθy) |
| 11 |  |  |  |  |
| 12 |  |  |  |  |

7. 每个元素32个二进制位，主存字长16位，故每个元素占2个字长，行下标可平移至1到11。

（1）242 （2）22 （3）s+182 （4）s+142

三、算法设计题（每题10分，共30分）

8. void Invert(queue Q)

//Q是一个非空队列，本算法利用空栈S和已给的几个栈和队列的ADT函数，将队列Q中的元素逆置。

{makempty(S); //置空栈

 while (!isEmpty(Q))  // 队列Q中元素出队

 {value=deQueue(Q);  push(S,value);  }// 将出队元素压入栈中

while(!isEmpty(S)) //栈中元素退栈

  {value=pop(S);   enQueue(Q,value); }//将出栈元素入队列 Q

 }//算法invert 结束

9. void DisCreat1(LinkedList A)

∥A是带头结点的单链表，链表中结点的数据类型为整型。本算法将A分解成两个单链表B和C，B中结点的数据小于零，C中结点的数据大于零。

{B=A;

C=(LinkedList )malloc(sizeof(LNode));∥为C申请结点空间。

C->next=null ∥C初始化为空表。

p=A->next; ∥p为工作指针。

B->next=null; ∥B表初始化。

while(p!=null)

{r=p->next; ∥暂存p的后继。

if (p->data<0)∥小于0的放入B表。

{p->next=B->next; B->next=p; }∥将小于0的结点链入B表。

else {p->next=C->next; C->next=p; }

p=r;∥p指向新的待处理结点。

}

}∥算法结束。

10. LinkedList  DelInsert（LinkedList heada，headb，int i，j，len）

∥heada和headb均是带头结点的单链表。本算法删除heada链表中自第i个元素起的共len个元素，然后将单链表heada插入到headb的第j个元素之前。

{

if（i<1 || len<1 || j<1）{printf（“参数错误\n”）；exit（0）；}∥参数错，退出算法。

p=heada；∥p为链表A的工作指针，初始化为A的头指针，查到第i个元素时，p指向第i-1个元素

  k=0；∥计数

  while（p!=null && k<i-1）∥查找第i个结点。

{

k++；p=p->next；

}

  if（p==null）

{

printf（“给的%d太大\n”,i）；exit（0）；

} ∥i太大，退出算法

 q=p->next；∥q为工作指针，初始指向A链表第一个被删结点。

 k=0；

while（q!=null && k<len）

{

k++；u=q；q=q->next；free（u）；

} ∥删除结点，后移指针。

if（k<len）

{

printf（“给的%d太大\n”,len）；exit（0）；

}

p->next=q；∥A链表删除了len个元素。

if (heada->next!=null)  ∥heada->next=null说明链表中结点均已删除，无需往B表插入

{

while（p->next!=null）

p= p->next；∥找A的尾结点。

q=headb；∥q为链表B的工作指针。

k=0；∥计数

while（q!=null && k<j-1） ∥查找第j个结点。

      {

k++；q= q->next；

}      ∥查找成功时，q指向第j-1个结点

if（q==null）

{

printf（“给的%d太大\n”,j）；exit（0）；

}

p->next=q->next；     ∥将A链表链入

q->next=heada->next； ∥A的第一元素结点链在B的第j-1个结点之后

}//if

free（heada）；∥释放A表头结点。

}∥算法结束。