83

Data Structure(B 卷)

Midterm Exam. 1

2022/10/20

學號: 41047025S 姓名: 王重鈞

總分 105 分 最高以 100 分計

1. Describe the worst case running time of the following code in "big-Oh" notation in terms of the variable *n* (i.e. O(n), O(n^2), O(2^n), O(n^3), O(n^4), O(log n), O(1), O(nlogn), O($\log^2$ n), etc.) You should give the tightest bound possible. (15 分)

15

(a)
```
void f1(int n) {
for(int i=0; i < n; i++) {
    for(int j=0; j < 10; j++) {
        for(int k=0; k < n; k++) {
            for(int m=0; m < 10; m++) {
                printf("!");
    }}}}
}
```

(b)
```
int f2(int n) {
if (n < 10) {
    printf("!");
    return n+3;}
else {
        return f2(n-1) + f2(n-1);
        }
 }
```

(c)
```
int f3(int n) {
if (n < 10) {
    printf("!");
    return n+3;}
else {
        return f3(n-1) + 1;
        }
 }
```

Ans:

<a>	$O(n^2)$
<b>	$O(2^n)$
<c>	$O(n)$

2. Please complete the following Table to show the progress of converting the infix expression 1-2*((3+4)-1*2)/4 to its postfix expression using a stack. (10 分)

20

Ans:

Input Token	Content in Stack (from bottom to top)	Output
1		1
-	-	1
2	-	12
*	-X	12
(	-X(	12
(	-X((	12
3	-X((	123
+	-X((+	123
4	-X((+	1234

Input Token	Content in Stack (from bottom to top)	Output
)	-X(	1234+
)	-X(-	1234+
1	-X(-	1234+1
*	-X(-X	1234+1
2	-X(-X	1234+12
)	-X	1234+12X-
/	-/	1234+12X-X
4	-/	1234+12X-X4

1234+12X-X4/-

(b) Then describe the process of evaluating the obtained postfix expression to get the value of the expression by using a stack. (10 分)

Ans:

Input Token	Content in Stack (from bottom to top)	Output
1	1	
2	12	
3	123	
4	1234	
+	127	
1	1271	
2	12712	
X	1272	
-	125	

Input Token	Content in Stack (from bottom to top)	Output
X	1 10	
4	1 10 4	
/	1 2.5	
-	-1.5	-1.5

3. Please complete the following code for inserting to and deleting from a queue implemented by a **circular linked list**, where ***rear** points to the last node of the queue. Besides, when the queue is empty, ***rear** is NULL. (15 分)

```
void addq (queuePointer *rear, element item)
{ queuePointer temp = (queuePointer) malloc(sizeof(queuenNode));
  If (IS_FULL(temp)){
     fprintf(stderr "The memory is full\n");
     exit(1);
   }
  temp.item = item;

  { if (!(*rear)) {   /* queue is empty, rear points to node */
      *rear = ___<1>____;
      node->link = node;
    }
    else {
      /* queue is not empty, insert to the last */
      node->link = ___<2>_______;
      (*rear)->link = node;
      *rear = _____<3>_____;
  }
 }
```

```
Element delete(queuePointer *rear)
queuePointer front = (*rear)->link
if (!(*rear)){
    fprintf(stderrm "The queue is empty\n");
    exit(1);
    }
item = front->item;
if (front->link)==front
   *rear = _____<4>_____;
else
   (*rear) ->link =  _____<5>_____;
free(front);
return item;
}
```

Ans:

<1>	temp	<4>	NULL
<2>	*rear (*rear)->link	<5>	(*rear)->link->link front->link
<3>	(*rear)->link temp		

4. Suppose a polynomial is represented by an array of non-zero terms in the polynomial, where each non-zero term is composed of *coef* and *expon* fields.

```
void padd(int starta,int finisha,int startb, int finishb,
                                  int *startd,int *finishd)
{
/* add A(x) and B(x) to obtain D(x) */
   float coefficient;
   *startd = avail;
   while (starta <= finisha && startb <= finishb)
      switch(COMPARE(terms[starta].expon,
                     terms[startb].expon)) {
         case -1: /* a expon < b expon */
               attach(terms[startb].coef,terms[startb].expon)
               startb++;
               break;
         case 0: /* equal exponents */
               coefficient = terms[starta].coef +
                             terms[startb].coef;
               if (coefficient)
                  attach(coefficient,terms[starta].expon);
               starta++;
               startb++;
               break;
         case 1: /* a expon > b expon */
               attach(terms[starta].coef,terms[starta].expon)
               starta++;
   }
}
```

(a) Let $A(x)=x^{2n}+x^{2n-2}+...+x^2+x^0$ (the non-zero terms have even exponentials) and $B(x)=x^{2n+1}+x^{2n-1}+...+x^3+x$(the non-zero terms have odd exponentials). Given the **padd** function shown above, please determine the number of times the "COMPARE" function is executed. (5 分)

Ans: $2n+1-0+1-1=2n+1$

(b) What's wrong with the given **padd** function used for adding two polynomials? (5 分) How to modify the function? (5 分)

Ans: It should (a) the the term which haven't pushed into polynomial. So we can add two while behind the code.
If starta ≠ finisha, then attach a term
If startb ≠ finish b. then attach b term.

-3
-4

The smallest term will not add to new ponomial after while (starta <= finisha && startb <= finishb)
while (starta <= finisha) { attach (term[starta].coef, term[starta].expon); starta ++; }
while (startb <= finishb) { attach (term[startb].coef, term[startb].expon); startb ++; }

B4

5. Please order the following efficiencies (according to their complexities) from smallest to largest: (5 分)

a. 2^n b. n! c. n^2 d. 10^5 e. $n\log_2(n)$

Ans:

$10^5 < n\log_2(n) < n^2 < 2^n < n!$

6. If the efficiency of the function doIt() can be expressed as $O(n)=n^2$, please give the complexity of the following program segment in terms of n. (5 分)

```
i=1;
for (i<n)
  {doIt();
   i=i*2;}
```

Ans: i will x2 each time, it need $\log_2(n)$

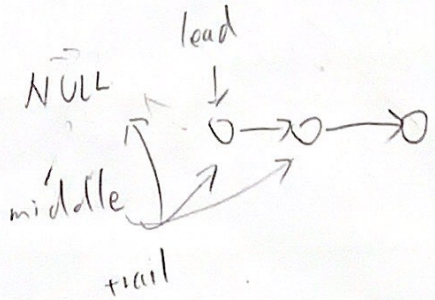
So, total time complexity is $O(n^2\log_2(n))$

7. The following C program code is used to invert the links of a singly linked list. Assume pointer *p* points to the first node. The pointer *p* has to point to the new first node after inversion. Please fill in the blanks. (The *p*, *trail*, *middle*, and *lead* are all pointer variables of node.)(15 分)

```
if (p == NULL) exit(1);
lead = p;
middle = NULL;
while (      <1>      ) {
trail = middle;
      <2>      ;
lead =      <3>      ;
middle->link =      <4>      ;
}
p =      <5>      ;
```

Ans:

<1>	lead->link ≠NULL ~~lead~~ lead !=NULL
<2>	middle = lead
<3>	lead = lead->link
<4>	trail
<5>	~~lead~~ middle



8. (a) (12 分) Declare the node in a doubly linked list to have three fields:

llink: points to the previous node

rlink: points to the next node

value: store the value in the node

Suppose we would like to insert a new node **newnode** to the next of **node**, please fill in the blanks in the following program code.

```
void dinsert (nodePointer node, nodePointer newnode)
{     /* insert newnode to the right of node */
      newnode->llink = node;
      _____<1>_____ = node->rlink;
      _____<2>_____ = newnode;
      node->rlink = newnode;
}
```

Suppose we would like to delete the node pointed by **deleted**, please fill in the blanks in the following program code.(6 分)

```
void ddelete (nodePointer Hnode, nodePointer deleted)
{  /* Hnode points to the header node of a doubly linked list
     if (Hnode == deleted)
          printf ("Deletion of head node not permitted.\n");
      else {
     deleted->llink->rlink = _____<3>_____;
     _____<4>_____ = deleted->llink;
     free (deleted);
      }
}
```

(b) What is the main advantage by using doubly linked list?(3 分)

Ans:

<1>	newnode → rlink	<3>	deleted → rlink
<2>	newnode → rlink → llink	<4>	deleted → rlink → llink
<b>	We can get previous node easily.		