ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2020

Homework 2

Due date: 23:59, September 21, 2020

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL Name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of above may result in zero score.

1: (4*1') Multiple Choices

Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 0.5 point if you select a non-empty subset of the correct answers.
Note that you should write you answers of section 1 in the table below.

Question 1	Question 2	Question 3	Question 4
A/AB	D	A/AB	A

Question 1. *Using linked list to implement a stack, where are the pushes and pops performed?*

(A) Push in front of the first element, pop the first element

(B) Push after the last element, pop the last element

(C) Push after the last element, pop the first element

(D) Push in front of the first element, pop the last element

(E) Push after the first element, pop the first element

Question 2. *Suppose we use a circular array with index range from 0 to N - 1 to implement a queue, and currently Front is pointing to index m. We will know that this queue is full if* ***Back*** *is point to index =* _____. *(Options below are* ***Integers Modulo N****:* $\mathbb{Z}_N = \{0, 1, 2, ..., N-1\}$*)*

(A) 0

(B) m

(C) $N-1$

(D) $m-1$

Question 3. *Stack cannot be used for*_____

(A) IO buffers

(B) resource allocation and scheduling

(C) compilers/word processors

(D) handling function calls

Question 4. *What function does the following code achieve?*

```
void Q4(Queue &Q)
{
  Stack S;
  int d;
  S.InitStack();
  while(!Q.IsEmpty())
  {
```

```
      d = Q.DeQueue();
      S.Push(d);
    }
    while(!S.IsEmpty())
    {
      d = S.Pop();
      Q.EnQueue(d);
    }
  }
```

(A) Use a stack to reverse a queue.

(B) Use a queue to reverse a stack.

(C) Use a stack to implement a queue.

(D) Use a queue to implement a stack.

2: (3'+8') Stack and Queue

Question 5. ***(1.5'+1.5')***

Consider the following sequence of stack operations:

push(d), push(h), pop(), push(f), push(s), pop(), pop(), push(m).

(a) We assume the stack is initially empty, please write down the sequence of the popped values, and the final stack (Please label in your answer the top and the bottom of the stack) .

(b) Now we work on a queue instead, the queue goes through the same operations as listed above. What will be the sequence of the popped values, and the final queue? (Please label in your answer the front and the back of the queue)

(a) popped value: h, s, f
final stack(left is bottom): d, m

(b) popped value: d, h, f
final stack(left is front): s, m

Question 6. ***(8')Postfix expression***

Reverse Polish notation (RPN) is a mathematical notation in which operators follow their operands. Using a stack, we can evaluate postfix notation equations easily.

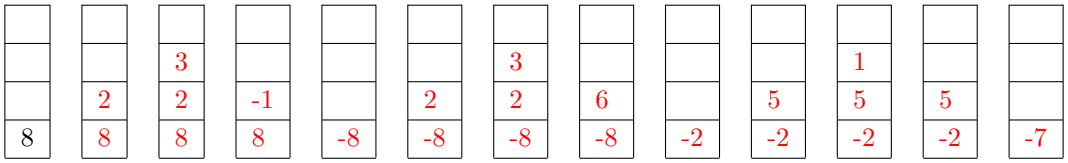
(a) ***Calculating(1'+2')***

A post-fix expression (Reverse-Polish Notation) with single digit operands is shown below:

8 2 3 − / 2 3 ∗ + 5 1 ∗ −

*Its in-fix expression is: 8/(2-3)+2*3-5*1*

The changing of the stack to calculate the final result is:



(b) ***Conversion(2*0.5'+2*1')***

Now, try to convert the following in-fix expression into post-fix expression: (You don't need to calculate them)

Note that ^ represents the exponentiation operator.

1) $1+2+3$

1 2 + 3 +

2) $1+5*9$

1 5 9 ∗ +

3) $1+3*5+(2*4+6)*8$

1 3 5 ∗ + 2 4 ∗ 6 + 8 ∗ +

4) $5+8*9\hat{}2/(5+3)+1$

5 8 9 2 ^ ∗ 5 3 + / + 1 +

(c) ***Validity(4*0.5')***

Please judge whether the following post-fix expression is legal, if legal, please write "T", otherwise please write "F".

(a) 1 2 + − 3 5 + *F*

(b) 4 5 6 / ∗ 1 / *T*

(c) 1 + 2 − 3 + 4 *F*

(d) 7 8 9 1 + − ∗ *T*

3: (5'+5') Big-O Notation

Question 7. ***(5') Asymptotic Analysis***

For each pair of functions $f(n)$ and $g(n)$, give your answer whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ or $f(n) = \Theta(g(n))$. (Try to give your answer in the most precise form.) For example, for $f(n) = n^2$ and $g(n) = n^2 + 2n + 1$, write $f(n) = \Theta(g(n))$. ***Briefly*** *justify your answers.*

1. $f(n) = n^{100}$ *and* $g(n) = 1.01^n$
   $f(n) = \mathcal{O}(g(n))$
   Note: Exponential dominates polynomial.

2. $f(n) = n^{1.01}$ *and* $g(n) = n \log n$
   $f(n) = \Omega(g(n))$
   Note: Polynomial dominates logarithm.

3. $f(n) = 3n + \log n$ *and* $g(n) = 2n + \log^2 n$
   $f(n) = \Theta(g(n))$
   Note: Polynomial dominates logarithm.

4. $f(n) = n!$ *and* $g(n) = n^n$
   $f(n) = \mathcal{O}(g(n))$
   Note: $n! = \mathcal{O}(n^n)$, *but* $log(n!) = \Theta(log(n^n))$, *You can refer to [Stirling formula] for details.*

5. $f(n) = \log^2(n)$ *and* $g(n) = \log \log n$
   $f(n) = \Omega(g(n))$
   Note: $\log^2 n = \Omega(\log n), \log n = \Omega(\log \log n)$.

Question 8. ***(5') Complexity Analysis***

Consider the factorial computing problem: $N! = 1 \times 2 \times \cdots \times N$.

1. *(2') To store the number of* N*, we need at least* $\log N$ *bits of space. Find an* $f(N)$ *such that* $N!$ *is* $\Theta(f(N))$ *bits long. Simplify your answer and justify your answer.*

   Solution:
   Method 1:
   Using Stirling's formula, you may get $N \log N$ *bits.*

   Method 2:
   When we multiply an m *bit number by an* n *bit number, we get an* $(m+n)$ *bit number. When computing factorials, we multiply* N *numbers that are at most* $\log N$ *bits long, so the final number has at most* $N \log N$ *bits. But if you consider the numbers from* $\frac{N}{2}$ *to* N*, we multiply at least* $\frac{N}{2}$ *numbers that are at least* $\log N - 1$ *bits long, so the resulting number has at least* $\frac{N(\log N - 1)}{2}$ *bits.*

2. *(3') Consider this vanilla factorial computing algorithm.*

   procedure *FACTORIAL(N)*
   $f = 1$
   for $i = 2$ *to* N ***do***
   $f = f \cdot i$
   end for
   return f
   end procedure

   Assume the runtime of multiplying an m-bit number and an n-bit number is $\Theta(mn)$*. What's the runtime of this factorial computing algorithm in Big-O notation? Justify your answers.*

   Solution: *Running time :* $(N^2 \log^2 N)$.
   We have N *iterations, each one multiplying an* $N \cdot \log N$*-bit number (at most) by an* $\log N$*-bit number. Using the naive multiplication algorithm, each multiplication takes time* $\mathcal{O}(N \cdot log^2 N)$*. Hence, the running time is* $\mathcal{O}(N^2 \log^2 N)$.

   A lower bound of $\Omega(N^2 \cdot \log^2 N)$ *follows because the product of the first* $N/2$ *numbers will be at least* $\Omega(N \log N)$ *bits long by the previous part, and the last* $N/2$ *multiplications take* $\Omega(N \log^2 N)$ *time each.*