Question 1

Consider how to implement a `Queue` as an `SLList`. When we enqueue an element, where does it go? When we dequeue an element, where does it come from?

(Here SLList is Singly linked list)

(a) We enqueue at the `head` and we dequeue at the `head`

(b) We enqueue at the `head` and we dequeue at the `tail`

(c) We enqueue at the `tail` and we dequeue at the `tail`

(d) We enqueue at the `tail` and we dequeue at the `head`

https://cglab.ca/~morin/teaching/2402/notes/sample-midterm-answers.pdf

Answer: B

Question 2

Consider a circular array-based implementation of the queue where the front and rear has been initialized to -1.

Array size is N.

Which of the following statements is true for the number of elements or size of the queue?

- S1: if front $>$ rear then size $= N-$ front $+$ rear $+1$
- S2: If front $\leq$ rear then size $=$ rear $-$ front $+1$

A. S1 is true but S2 is false
B. S1 is false but S2 is true
C. Both statements are true
D. Both statements are false

Answer: A

S2 is false because when front $=$ rear $=-1$ then number of elements would be zero but S2 says 1.

Question 3

. Consider how to implement a `Stack` as an `SLList`. When we push an element where does it go? When we pop an element where does it come from?

(a) We push at the `tail` and we pop at the `head`

(b) We push at the `tail` and we pop at the `tail`

(c) We push at the `head` and we pop at the `tail`

(d) We push at the `head` and we pop at the `head`

(e) None of the above

https://cglab.ca/~morin/teaching/2402/notes/sample-midterm-answers.pdf

Answer: C

Question 4

Identify the operations below that can be performed in $O(1)$ cost.

There is more than one correct choice; no marks are given unless you can identify all of them.

A. Push an element into a stack.
B. Insert an element into a linked list.
C. Dequeue an element from a queue.
D. Determine whether the value 10 is in a hash table.

https://www.cse.cuhk.edu.hk/~taoyf/course/2100/23-spring/quiz2-sol.pdf

Answer: A,B,C,D

Question 5

Part A

After applying the following operations to an empty stack:

push(35), push(36), push(43), push(8), pop, pop, push(51), pop

what is the content of the stack?

A. 35, 36
B. 36, 43
C. 35, 8
D. 35, 51

Part B

After applying the following operations to an empty queue:

enqueue(35), enqueue(36), enqueue(43), enqueue(8), dequeue, dequeue, enqueue(51), dequeue

what is the content of the queue?

A. 8, 51
B. 36, 43
C. 35, 8
D. 35, 51

https://www.cse.cuhk.edu.hk/~taoyf/course/2100/23-spring/quiz2-sol.pdf

Answer: A,A

Question 6

(a) We have three stacks, s1, s2 and s3. Here are their initial contents:

As you can see, initially s2 and s3 are empty. Here is a sequence of operations on the three stacks:

```
s2.push(s1.pop());
s3.push(s1.pop());
s1.pop();
s1.push(s2.pop());
s2.push(s3.pop());
s2.push(s1.pop());
```

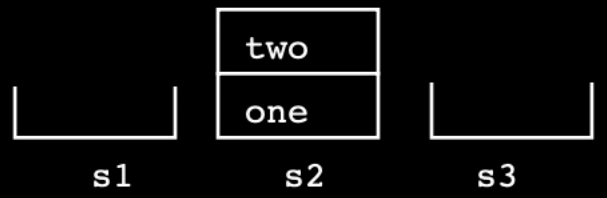
Draw the contents of the three stacks after the operations are complete.

(b) This question is about the same three stacks, `s1`, `s2` and `s3`, as in part (a). We add the rule that when you pop an element from a stack, it can only be pushed onto a higher-level stack. For example, "`s1.push(s3.pop());`" would not be allowed. Suppose we start from the same initial state as in part (a) and eventually move all the strings to `s3`. Give an example of a permutation (ordering) of the elements "`zero`", "`one`" and "`two`" that we *cannot* attain on `s3`.

http://www.cs.toronto.edu/~hojjat/148s07/tests/PastExams/20061ans.pdf

Answer

(a)



(b)

Question 7

Consider a sequence of push and pop operations used to push the integers 0 through 9 on a stack. The numbers will be pushed in order, however the pop operations can be interleaved with the push operations, and can occur any time there is at least one item on the stack. When an item is popped, it is printed to the terminal.

Which of the following could NOT be the output from such a sequence of operations?

(a) 0 1 2 3 4 5 6 7 8 9

(b) 4 3 2 1 0 5 6 7 8 9

(c) 5 6 7 8 9 0 1 2 3 4

(d) 4 3 2 1 0 9 8 7 6 5

(e) All of these output sequences are possible.

https://courses.engr.illinois.edu/cs225/sp2018/exams/practice/sp14mt2.pdf

Question 8

The array-based stack has the limitation which is we can not push more elements than capacity of the array.

Consider the following alternative :

This approach involves dynamically allocating a larger array using the malloc() function whenever the size of the stack is full. By doing so, we can push more elements than the current capacity of the array. Here's how it works:

Suppose the array is full, then to push the next element we first allocate a new array of larger size using malloc, copy all elements to the new array, and free up the old array memory.

If the stack size is initially set to 1, find the cost of performing n pushes in the following scenarios, assuming the cost of the malloc call is constant:

- Case A: When we create a larger array with a size one greater than the old array.
- Case B: When we create a larger array with a size double that of the old array.

In both cases, the objective is to determine the total cost incurred when pushing n elements onto the stack.

https://csc.csudh.edu/jhan/Fall2007/csc311/Assignments/CSC311-A2Solution.htm
https://www2.cs.uregina.ca/~mouhoubm/=postscript/=c3620/solmidt_00.pdf

Answer:

Case A: $O(n^2)$
Case B: $O(n)$

Hint/Solution for Case B -

Case 1: If we do not need to allocate new memory.

Algorithm takes $O(1)$ unit of time.

Case 2: If we need to allocate new memory.

For which values of i do we need to allocate new memory?

at the $2^1, 2^2, 2^{logn}$*th pushes, need array copies, total copies:* $2^0 + 2^1 + 2^2 + \ldots + 2^{logn} = 2^{logn+1} - 1 = 2n + 1$

Question 9

Consider an array based implementation of a stack, and suppose that it is initially empty. Upon n **push** operations the array will be resized in such a way that the running time per push is $O(1)$ per operation, on average. How many times is the array resized over the n **pushes**, using this scheme?

(a) $O(1)$
(b) $O(\log n)$
(c) $O(n)$
(d) $O(n \log n)$
(e) $O(n^2)$

This question consider the resize as doubling the size of array each time array is full

https://courses.engr.illinois.edu/cs225/sp2018/exams/practice/sp14mt2.pdf

Answer: B

Question 10

Describe the output of the following series of stack operations on a single, initially empty stack:
$push(5)$, $push(3)$, $pop()$, $push(2)$, $push(8)$, $pop()$, $pop()$, $push(9)$, $push(1)$, $pop()$, $push(7)$, $push(6)$, $pop()$, $pop()$, $push(4)$, $pop()$, $pop()$.

Answer

Solution (bottom of the stack is to the left)

5
5 3
5
5 2
5 2 8
5 2
5
5 9
5 9 1
5 9
5 9 7
5 9 7 6
5 9 7
5 9
5 9 4
5 9
5

Question 11

Suppose you have a stack in which the values 1 through 5 must be pushed on the stack in that order, but that an item on the stack can be popped at any time. Give a sequence of push and pop operations such that the values are popped in the following order:

(a) 2, 4, 5, 3, 1

(b) 1, 5, 4, 2, 3

(c) 1, 3, 5, 4, 2

It might not be possible in each case.

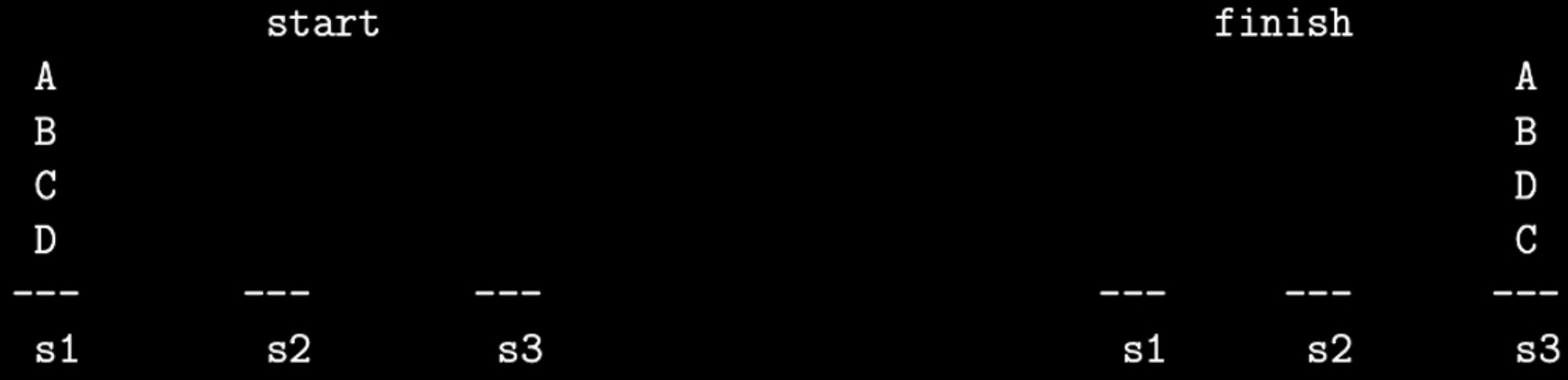
https://www.cim.mcgill.ca/~langer/250/E6-stacks-queues.pdf

Answer

24531	15423	13542
push 1	push 1	push 1
push 2	pop	pop
pop	push 2	push 2
push 3	push 3	push 3
push 4	push 4	pop
pop	push 5	push 4
push 5	pop	push 5
pop	pop	pop
pop	x	pop
pop	(not possible)	pop

Question 12

(a) Suppose you have three stacks s1, s2, s3 with starting configuration shown on the left, and finishing condition shown on the right. Give a sequence of push and pop operations that take you from start to finish. For example, to pop the top element of `s1` and push it onto `s3`, you would write `s3.push( s1.pop())`.



(b) Same question, but now suppose the finish configuration on s3 is `BDAC` (with `B` on top) ?

https://www.cim.mcgill.ca/~langer/250/E6-stacks-queues.pdf

Answer

(a)
```
s2.push( s1.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s2.pop() )
s3.push( s2.pop() )
```

(b)
```
s2.push( s1.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s1.push( s2.pop() )
s3.push( s2.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s2.pop() )
```

Question 13 (HARD !!)

Consider the scenario where we aim to design a stack with the following operations:

- push(x): Add element x to the stack.
- pop(): Remove the top element from the stack.
- getMinimum(): Retrieve the minimum value from the stack without removing it.

We have the option to utilize additional stack(s) to achieve the desired functionalities. Which of the following statements is true?

- A) A minimum of 2 extra stacks is required to implement all three functions (push, pop, and getMinimum) in O(1) time complexity.
- B) A minimum of 1 extra stack is required to implement all three functions (push, pop, and getMinimum) in O(1) time complexity.
- C) To implement all three functions (push, pop, and getMinimum) in O(1) time complexity, at least one extra stack and one extra queue are needed.
- D) Regardless of the number of extra stacks used, it is not possible to implement the getMinimum() function in O(1) time complexity.

https://cs.brown.edu/courses/cs016/static/files/assignments/optional_sol/optional_hw2_sol.pdf

Answer: B

Maintain a separate stack that has minimums in it.
Lets call this extra stack as "minimum stack"

pop(): Check if the top of the minimum stack is the same as the main stack.
If it is, pop both stacks.

push(x):

- Push to the main stack
- Check if the element you're adding is equal to or less than the top of the minimum stack. If it is, push it onto the minimum stack too.

getMinimum(): return top() of minimum stack.

See next page for example

Suppose the element to be pushed are –

5, <many input numbers greater than 5>, 3, <many input numbers greater than 3>,1, <many input numbers greater than 1>

(For example, think about:- 5, 6,10,7,9,3,8,11,6,4,1,6,9,13)

Here in the minimum stack we are storing only 5, 3 and 1. And in main stack we are storing everything.

Now let's start poping elements one by one from main stack.

- First pop is 13, once u pop it,. minimum won't change. It is still 1.
- Next pop is 9, min is still 1.
- Basically we can pop everything to right of 1, minimum will be 1 only.
- Now once you pop 1, then pop from both stack.
- Our next minimum is 3 only. It will remain to 3 till we pop 3.
- We can continue same idea. It shows that we only need to store 5,3 and 1 in helper stack.

(It is very rare to think this algorithm by yourself)

Question 14

Consider the following sequence of stack commands:

`push(a), push(b), push(c), pop(), push(d), push(e), pop(), pop(), pop(), pop().`

(a) What is the order in which the elements are popped ? (Give a list and indicate which was popped first.)

(b) Change the position of the `pop()` commands in the above sequence so that the items are popped in the following order: `b,d,c,a,e`.
You are *not* allowed to change the ordering of the `push` commands.

https://www.cim.mcgill.ca/~langer/250/E6-stacks-queues.pdf

Answer

(a) c (popped first), e, d, b, a

(b) push(a), push(b), pop(), push(c), push(d), pop(), pop(), pop() push(e), pop()

Question 15

We discussed a few different ways of initializing the `front`/`back` index variables in an array-based queues.

Which of the following choices show valid initial values for those indexes, assuming the rest of the functions are updated (if necessary) to make everything consistent.

This question may have more than one correct answer—choose all answers you believe are correct.

i. `front = 0, back = 0`

ii. `front = -1, back = -1`

iii. `front = -1, back = cap - 1`

iv. `front = cap - 1, back = cap - 1`

v. The array-based queue lectures were so confusing that I have no idea how to answer this question.

http://mjgeiger.github.io/eece3220/prev/sp19/exams/eece.3220sp19_exam3_soln.pdf

Answer: A,B,C,D

Frankly, as long as you're consistent across all of your Queue functions, any pair of initial values for array indexes will work.

Question 16

Suppose you have a stack S containing n elements.
Describe (in English) how you can scan S to see if it contains a certain element x.

You are allowed to use one additional stack OR queue.

Please note that you must preserve the original order of elements in the stack, and you are only allowed to perform push or pop operations on the stack.

https://www.eecs.yorku.ca/course_archive/2017-18/W/2011Z/problem%20sets/problem%20set%202%20solutions.pdf

Answer

Answer: We use the queue Q to process the elements in two phases. In the first phase, we iteratively pop each element from S and enqueue it in Q, and then we iteratively dequeue each element from Q and push it into S. This reverses the elements in S. Then we repeat this same process, but this time we also look for the element x. By passing the elements through Q and back to S a second time, we reverse the reversal, thereby putting the elements back into S in their original order.

Question 17

In this problem we consider two stacks A and B manipulated using the following operations (n denotes the size of A and m the size of B):

- Push $A(x)$: Push element x on stack A.
- Push $B(x)$: Push element x on stack B.
- Transfer(k): Repeatedly pop an element from A and push it on B, until either k elements have been moved or A is empty.

Assume that A and B are implemented using doubly-linked lists such that PushA and PushB, as well as a single pop from A or B, can be performed in $O(1)$ time worst-case.

What will be the TIGHEST bound on worst case running time for operation Transfer() ?

A. $O(m+n)$
B. $O(n)$
C. $O(m)$
D. $O(mn)$

https://courses.cs.duke.edu/cps130/summer02/Homeworks/Solutions/H17-solution.pdf

Answer: B

Question 18

Given a stack with 15 elements and a queue with 30 elements, performing the following set of operations 60 times:

- enqueue(pop()) /*remove the top element of the stack and insert it into the queue
- push(dequeue()) /*remove the last element of the queue and insert it into the stack

How many of the 45 total elements are involved in this process?

(Note: This is NOT asking how many times elements move, but how many elements move.)

http://seu.wangmengsd.com/ds/hw2.pdf

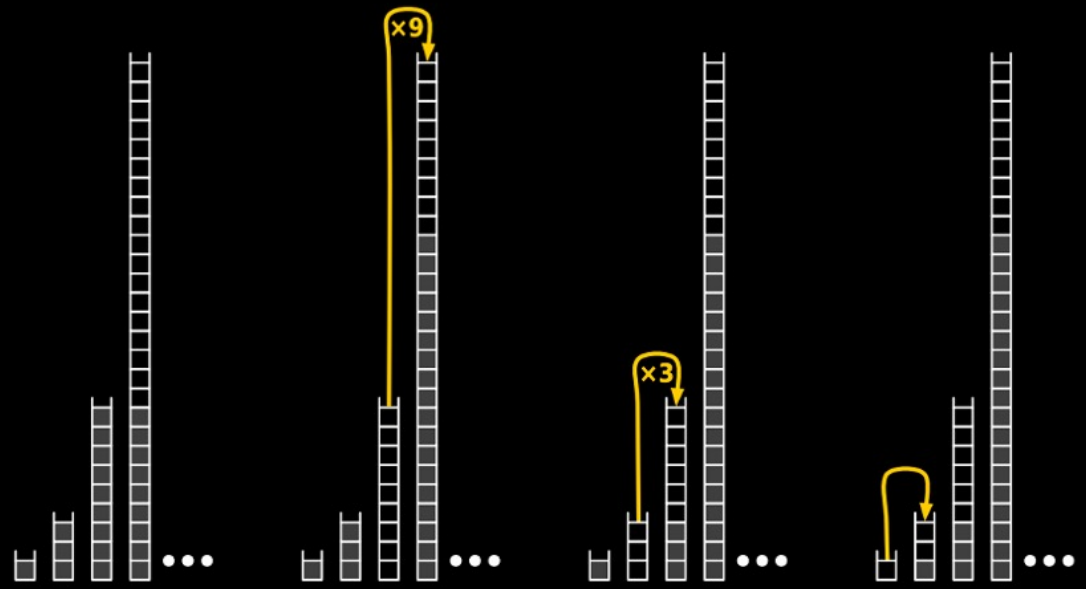
Question 19

A *multistack* consists of an infinite series of stacks $S_0, S_1, S_2, \ldots$, where the ith stack S_i can hold up to 3^i elements. The user always pushes and pops elements from the smallest stack S_0. However, before any element can be pushed onto any full stack S_i, we first pop all the elements off S_i and push them onto stack S_{i+1} to make room. (Thus, if S_{i+1} is already full, we first recursively move all its members to S_{i+2}.)

Moving a single element from one stack to another takes $O(1)$ time.

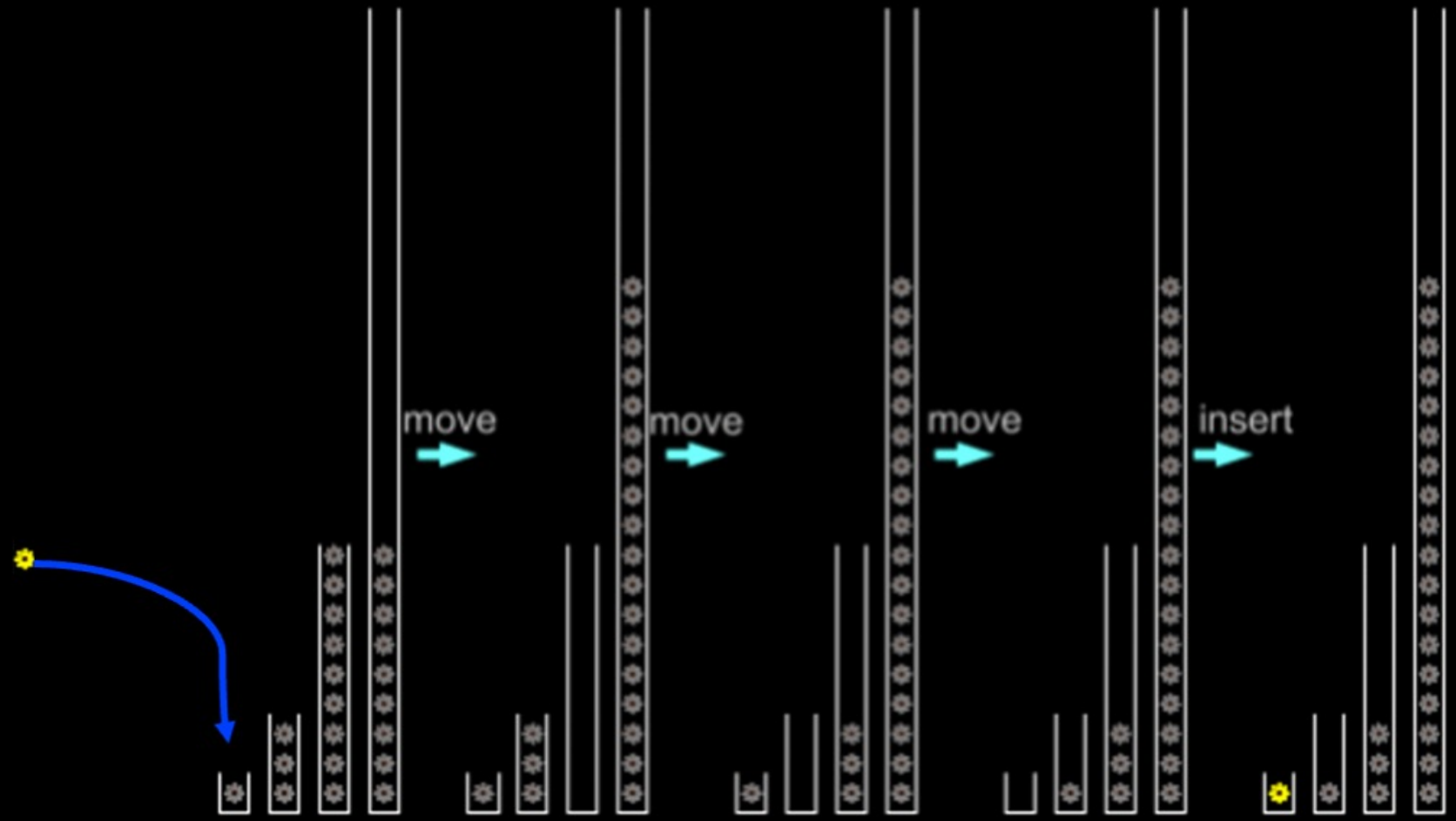
Here is pseudocode for the multistack operations MSPUSH.

```
MPUSH(x):
    i ← 0
    while S_i is full
        i ← i + 1

    while i > 0
        i ← i − 1
        for j ← 1 to 3^i
            PUSH(S_{i+1}, POP(S_i))

    PUSH(S_0, x)
```



Making room in a multistack, just before pushing on a new element.

https://courses.engr.illinois.edu/cs473/sp2010/homework/hw5.pdf



Question on next page -

a) State the worst case running time to push one more element onto a multistack containing n elements. Prove your answer by means of a worst case example.

b) What will be the total time to push n elements into multistack ?

https://ac.informatik.uni-freiburg.de/teaching/ws19_20/algo1920/algo-exams/ExamSpring18.pdf

Question 20

Suppose you implement a **queue** using a singly linked list with head and tail pointers so that the front of the **queue** is at the tail of the list, and the rear of the **queue** is at the head of the list. What is the best possible worst-case running time for **enqueue** and **dequeue** in this situation? (As a reminder, **enqueue** occurs at the rear of the queue.)

(a) O(1) for both functions.

(b) O(1) for enqueue and O(n) for dequeue.

(c) O(n) for enqueue and O(1) for dequeue.

(d) O(n) for both functions.

(e) None of these is the correct response.

https://courses.engr.illinois.edu/cs225/sp2018/exams/practice/sp14mt2.pdf

Question 21

Think of an algorithm that uses a `Stack` to efficiently check for unbalanced brackets. What is the maximum number of characters that will appear on the stack at any time when the algorithm analyzes the string `([]()[()])`?

(a) 3
(b) 4
(c) 5
(d) 6
(e) None of these is correct.

https://courses.engr.illinois.edu/cs225/sp2018/exams/practice/sp14mt2.pdf

Question 22

Alice has two queues, Q and R, which can store integers, Bob gives Alice 50 odd integers and 50 even integers and insists that she store all 100 integers in Q and R.
They then paly a game where Bob picks Q or R at random and then applies the round-robin scheduler(described below) to the chosen queue a random number of times. If the last number to be processed at the end of this game was odd, Bob wins. Otherwise Alice wins.

How can Alice allocate integers to queues to optimize her change of winnning? What is her chance of winning?

Round-robin scheduler dequeue a element from front of the queue and enqueue to the back of queue.

https://csc.csudh.edu/jhan/Fall2007/csc311/Assignments/CSC311-A2Solution.htm

Answer:

Case 1: Distribute equally

Here in this case Q will contain 25 odd integers and 25 even integers, same for $\mathbf{R}$.

The chance that Bob picks Q or R is 0.5 .

$P(\text{ Alice win }) = P(Bob \text{ picks } Q \cap \text{ Alice win }) + P(Bob \text{ picks } R \cap \text{ Alice win })$

$P(Bob \text{ picks } Q \cap \text{ Alice win }) = P(Bob \text{ picks } Q) \times P(\text{ Alice win } \mid \text{ Bob picks } Q) = 0.5 \times \frac{25}{50}$

Similarly, $P(Bob \text{ picks } R \cap \text{ Alice win }) = 0.5 \times \frac{25}{50}$

Hence $P(\text{ Alice win }) = \left(0.5 \times \frac{25}{50} + 0.5 \times \frac{25}{50}\right) = \frac{25}{50} = \frac{1}{2}$

Case 2: Allocate an even number to a queue and others at the other queue. This will lead the probablility that Alice wins to:

$$0.5 \times \left(1 + \frac{49}{99}\right) \approx 74.75(\%)$$

Looks good. Then How about allocating more than one even integers to a queue? Let's test with 25 even integers.

$$0.5 \times \left(1 + \frac{25}{75}\right) \approx 66.67(\%)$$

You might notice that allocating more than one even integer just reduce the Alice's winning probability of the case that Bob chooses the queue which have bot even and odd integers. Therefore, Alice should allocate an even integer to one queue.

Question 23

Consider the following function.

```
q = Queue();

void f () {
    stack = Stack();
    while (!q.isEmpty())
        stack.push(q.dequeue());
    while (!stack.isEmpty())
        q.enqueue(stack.pop());
    }
```

How does the execution of f affect q?

- ◯ q is unchanged after f executes
- ◯ q is empty after f executes
- ◯ q is reversed after f executes

Question 24

A stack of int is implemented using an array as the following data type:

```
#define SIZE 20
typedef struct {
    int data[SIZE];
    int top;
} Stack;
```

Fill up the missing codes in the PUSH, POP, and TOP operations of the Stack.

Ignore underflow or overflow of stack in case of pop and push respectively. You can assume that underflow or overflow are handled before calling pop or push.

```
void Push(Stack *s, int d) {
    // line X
}
void Pop(Stack *s) {
    // line Y
}
int Top(Stack *s) {
    // line Z
}
```

A.
```
line X should be s->data[++s->top] = d;
```

B.
```
line Y should be --s->top;
```

C.
```
line Z should be return s->data[s->top]
```

D. None of these

Answer: A,B,C

Question 25

Given the following sequence of letters and asterisks: EAS*Y*QUE***ST***IO*N***

Consider the stack data structure, supporting two operations *push* and *pop*, as discussed in class. Suppose that for the above sequence, each letter (such as E) corresponds to a *push* of that letter onto the stack and each asterisk (*) corresponds a *pop* operation on the stack. Show the sequence of values returned by the *pop* operations.

https://www.cse.iitb.ac.in/~sri/cs213/Midsem-questions