A function that takes exactly  n+ n \* log(n2) + 2  steps

  O(n2 \* log(n))

  O(n)

O(n log(n))

O(log(n2))

 None of the above

Cosider the following lines of codes and compute the overall complexity -

for (int i = 1; i \* i < n; i += 2)

{  // Do Something

}

  for (int j = 1 ; j < n ; j \*= 2)

{  // Do Something

}

O(sqrt(n))

O(n2)

O( log(n))

O(sqrt(n)\*log(n))

None of the above

 Consider the following line of codes find the overall complexity -

struct node\* current1 = lst1->firstLink->next;

struct node\* current2 = lst2->firstLink->next;

int found =0;

while (current1 != lst1->lastlink)

{

while (current2 != lst2->lastlink)

 {  
  
if ( current1 ->value == current2 ->value)

 found ++;

else

current2 = current2->next;

}

current2 = lst2->firstLink->next;

current1= current1->next;

}

O(logn)

O(n+1)

O(n)

O(n3)

None of the above

 Consider an empty but initialized stack using a dynamic array implementation with initial capacity 4. Suppose writing a new element to the array costs 1 unit, and copying a single element during reallocation also costs 1 unit. How many total units will it cost to call 10 consecutive push operations on this stack?

20

10

25

22

None of the above

 Find the term on the right that BEST matches the description on the left. You may use a term more than once)

**Correct!**

**A. The order that elements are placed into that ADT is completely unimportant.**



**You Answered**

**B. This ADT would be ideal to implement a collection of files being held until they can be printed on a printer**

**Correct Answer**

F. queue

**You Answered**

**C. This ADT would be ideal for implementing a finite length undo in an application like “Photoshop”**

**Correct Answer**

C. deque

**You Answered**

**D. ADT that simplifies the allocation of memory for program variables**

**Correct Answer**

H. stack

**You Answered**

**E. This ADT provides the ability for a user to loop over elements and maintains encapsulation for the developer**

**Correct Answer**

I. Iterator

**You Answered**

**F. ADT that has a property that elements can only be added or removed from the end points**



**Correct Answer**

C. deque

**Correct!**

**G. Last in First Out**



**You Answered**

**H. ADT that includes the Add, Contains, and Remove behaviors**

**Correct Answer**

A. bag

**You Answered**

**I. ADT used to design searching a maze using breadth-first search**

**Correct Answer**

F. queue

**You Answered**

**J. Resizing is never required if we use the data structure**

**Correct Answer**

E. linked list

**Question 6**

**5.4 / 8 pts**

[**Skip to question text.**](https://oregonstate.instructure.com/courses/1524715/quizzes/2306413?headless=1#question_46157770_question_text)

**Please read the following question, and then provide the answer to this question as a file attachment.**

Write a function \_dynArrayAddAt(…) that adds an element at a particular index in the dynamic array.  It **does not overwrite** a value, rather adds a new value. Don’t use any other helper function.

void dynArrayAddAt (struct DynArr \* da, int index, TYPE val) { }

 Please draw the dynamic array stack structure (\*\*you must mention the size and capacity) after the following commands 1-10 are executed.  You do not have to show all steps, however, it is recommended for partial credit. You can assume that a resize doubles the capacity.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

  0      1     2      3

1. struct dynArr \*stack  = createDynArray(4);

2. pushDynArray (&stack, 1);

3. popDynArray (&stack);

4. pushDynArray (&stack, 4);

5. pushDynArray (&stack, 5);

6. pushDynArray (&stack, 3);

7. popDynArray (&stack);

8.  popDynArray (&stack);

9.  pushDynArray (&stack, 3);

10. pushDynArray (&stack, 11);

Your Answer:

Question 7:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

  0      1     2      3

1. struct dynArr \*stack  = createDynArray(4);
2. pushDynArray (&stack, 1);
3. popDynArray (&stack);
4. pushDynArray (&stack, 4);
5. pushDynArray (&stack, 5);
6. pushDynArray (&stack, 3);
7. popDynArray (&stack);
8. popDynArray (&stack);
9. pushDynArray (&stack, 3);
10. pushDynArray (&stack, 11);

Push = add, pop = remove

1. Create array with 4 blocks.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Cap = 4, cnt = 0, beg = 0

1. Add 1 to block 0

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  |  |

Cap = 4, cnt = 1, beg = 0

1. Remove 1 from block 0

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Cap = 4, cnt = 0, beg = 0

1. Add 4 to block 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 |  |  |  |  |

Cap = 4, cnt = 1, beg = 0

1. Add 5 to block 1

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 |  |  |

Cap = 4, cnt = 2, beg = 0

1. Add 3 to block 2

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 | 3 |  |

Cap = 4, cnt = 3, beg = 0

1. Remove 3 from block 2

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 |  |  |

Cat = 4, cnt = 2, beg = 0

1. Remove 5 from block 1

|  |  |  |  |
| --- | --- | --- | --- |
| 4 |  |  |  |

Cap = 4, cnt = 1, beg = 0

1. Add 3 to block 1

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 |  |  |

Cap = 4, cnt = 2, beg = 0

1. Add 11 to block 2

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 11 |  |

Cap = 4, cnt = 3, beg = 0

Write down the worst-case computational complexity for adding to the back of a queue using a linkedlist with a first-link pointer and a last-link pointer, assuming the first link is the front.

O(n)

O(1+)

O(1)

O(logn)

None of the above

Write down the worst-case computational complexity for removing from the back of a deque using a linked list with only next pointers, assuming the next pointers point towards the back.

O(n)

O(n2)

 O(1)

O(logn)

None of the above

 What is the big-O execution time of a function that requires to iterate through the indexes on a linked list ?

  O(n3)

  O(n2)

  O(sqrt(n))

O(logn)

None of the above

 Fill in the following Chart with the average big-O execution times for the simple un-ordered dynamic array bag , the linked list bag and the ordered dynamic array bag .

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dynamic Array Bag | Linked list Bag | Ordered Array Bag |
| Add |  |  |  |
| Contains |  |  |  |
| Remove |  |  |  |

Your Answer:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dynamic Array Bag | Linked List Bag | Ordered Array Bag |
| Add | 1 | 1 | 1 |
| Contains | N | N | N |
| Remove | 1 | N | N |