Algorithm and DS - test #1

Всего 24/30

front-end_37-38m

Электронная почта *

v.zhevaga@gmail.com

Баллов: 24 из 30.

✓ An algorithm is... 2 из 2

- ☐ what a computer does
- ☐ a word from Wikipedia.
- ☑ a finite set of well-defined rules. ✓
- ☐ the rules at the airport.

✗ Check algorithm properties... 0 из 2

- ☑ should return a value. ✗
- ☑ should terminate after a finite time. ✓
- ☑ makes sure every step should do some work. ✓
- ☐ should save data.
- ☐ makes sure every step in the algorithm must have a method in the code.

Правильный ответ

- ☑ should terminate after a finite time.
- ☑ makes sure every step should do some work.

✗ Asymptotic analysis is... 0/2

- an analysis of the time it will take to process a very large dataset.
- [✓ an analysis of processing time, regardless of the data set.]
- an analysis of a large data set, regardless of processing time.
- an analysis of a large data set, with an algorithm set processing time.

Правильный ответ

- [✓ an analysis of the time it will take to process a very large dataset.]

✓ What does 'n' O(n) mean? 2/2

- 'n' is the data that the algorithm received.
- [✓ that the algorithm will require at most 'n' steps.]
- that it is a slow algorithm.
- 'n' is something important, but I forgot.

✓ The asymptotic running time of an algorithm is expressed by... 2/2

- small "o" notation.
- [✓ big "O" notation.]
- "Speed" notation.
- big "P" notation.

✓ Analyzing algorithms we count... 2/2

- [ ] time complexity and break time.
- [x] time complexity and space complexity.
- [ ] space complexity and time recursion
- [ ] time complexity and memory space

✓ Which algorithms do have O(n log n) average time complexity? 2/2

- [x] Merge sort
- [ ] Bubble sort
- [ ] Linear search
- [x] Quick sort
- [ ] Binary search

✓ What is recursion? 2/2

- [ ] Spit roasting meat on an open fire.
- [ ] A method in java that prints something.
- [x] The process where a function defines itself or its type.
- [ ] A function that calculates how many calories I have eaten.
- [ ] When a function overflows the Stack.

✓ A typical 'divide and conquer' algorithm solves a problem through the following steps:

- [ ] start algorithm
- [x] divide
- [ ] return
- [ ] calculate
- [x] conquer
- [x] combine
- [ ] sort

✓ Amortized analysis are used for... 2/2

- [x] algorithms where some operations are very slow, but others are faster.
- [ ] algorithms where some operations are very fast, but others are faster.
- [ ] algorithms where we cannot apply Asymptotic Analysis.
- [ ] algorithms that have a lot of incoming and outgoing data.

✓ What are the main types of amortized analysis? 2/2

- [ ] managerial analysis
- [x] aggregate analysis
- [ ] logical method
- [x] accounting method
- [x] potential method

✓ What is Dynamic programming? 2/2

- ☐ Creating a dynamic array.
- [✓ Simple recursion optimization.]
- ☐ High speed programming.
- ☐ Programming one recursion that defines another recursion.

✓ Stack is... 2/2

- ☐ FIFO
- [✓ LIFO]
- ☐ FULL
- ☐ FILO
- ☐ FIFA

✓ Queue is... 2/2

- [✓ FIFO]
- ☐ LIFO
- ☐ FULL
- ☐ FILO
- ☐ FIFA

✗ Select all that classify data structures. 0 / 2

- [✓] Array
- [ ] Map
- [✓] LinkedList
- [✓] Stack
- [✓] Queue
- [✓] Tree
- [✓] Graph

Правильный ответ

- [✓] Array
- [✓] Map
- [✓] LinkedList
- [✓] Stack
- [✓] Queue
- [✓] Tree
- [✓] Graph

Given an array 'arr[]' of positive integers, flip each group of subarrays to size 'K.'

Example 1:
K = 3
arr[] = {1,2,3,4,5}
Output: 3 2 1 5 4
Explanation: The first group consists of elements
1, 2, 3. The second group consists of 4,5.

Example 2:
K = 3
arr[] = {5,6,8,9}
Output: 8 6 5 9

Your task:
To write a reverse (arr, k) function that takes 'arr[]' and 'K' as input and modifies the array into place.

```
function reverseGroup(arr, k) {
    if (k <= arr.length) {
        if (k == 0) {
            return arr;
        }
        if (k == arr.length) {
            return arr.reverse();
        }
        return [
            ...arr.slice(0, k).reverse(),
            ...arr.slice(k, arr.length).reverse(),
        ];
    }
}

const arr = [1, 2, 3, 4, 5];
const arr2 = [5, 6, 8, 9];
console.log(reverseGroup(arr, 3)); // => [ 3, 2, 1, 5, 4 ]
console.log(reverseGroup(arr2, 3)); // => [ 8, 6, 5, 9 ]
```

Divide and Conquer

Баллов: 0 из 0.

Ниже представлены задачи трёх уровней, обычные, с одной и двумя звёздочками. Можно выбрать Нужно решить одну любую задачу, уровень сложности выбираете самостоятельно

// Find the smallest positive element, which given sorted array doesn't contain. All elements of an array are sorted

// Example: [1, 2, 6, 31]
// Result: 3
//
// Example: [2, 3, 4, 6, 9, 11, 15]
// Result: 1

//Expected time complexity O(log(n))

signature example java
public static int smallestMissing(int[] arr) {
}

```
// Решение O(n)
function smallestMissing(arr) {
    let count = 0;
    while (arr[count] == count + 1) {
        count++;
    }
    return ++count;
}

const arr_1 = [1, 2, 6, 31];
const arr_2 = [2, 3, 4, 6, 9, 11, 15];
console.log(smallestMissing(arr_1)); // => 3
console.log(smallestMissing(arr_2)); // => 1
```

// Find in a sorted array the closest element to the given number from below and above, -1 otherwise
// Example: arr = [0, 1, 2, 6, 31], n = 5
// Result: below = 2, above = 6
//
// Example: arr = [7, 10, 15, 21, 29], n = 31
// Result: below = 29, above = -1

// Example: arr = [7, 10, 15, 21, 29], n = 5
// Result: below = -1, above = 7

//Expected time complexity O(log(n))

signature example java
public static int[] findFloor(int[] arr) {
}

```
// Задачка со звёздочкой *
// Implement merge sort algorithm for a singly linked list

Example: given Node(5) -> Node(3) -> Node(6) -> Node(2)

return Node(2) -> Node(3) -> Node(5) -> Node(6)



java example of Node

class Node {
private int data;
private Node next;

Node(int data, Node next) {
this.data = data;
this.next = next;
}

public int getData() {
return data;
}

public void setData(int data) {
this.data = data;
}

public Node getNext() {
return next;
}

public void setNext(Node next) {
this.next = next;
}
}

public static Node mergeSort(Node head) {

}
```

```
// Задачка со звёздочкой *
// Find `k` closest elements to a given value in a sorted array
// Example: arr = [0, 5, 8, 10, 12, 16, 17, 22], k = 3, n = 11
// Result: 8, 10, 12
//
// Example: arr = [8, 9, 11, 15, 19,22, 25, 26, 27], k = 4, n = 22
// Result: 19, 22, 25, 26

public static int[] findKClosest(int[] arr, int k, int n) {
}
```

*// Задачка со двумя звёздочками ***
// Sort a doubly-linked list using quick sort
Example:
given Node(5) <-> Node(3) <-> Node(6) <-> Node(2)

return Node(2) <-> Node(3) <-> Node(5) <-> Node(6)

```
class Node {
int data;
Node next;
Node prev;

public Node(int data, Node next, Node prev) {
this.data = data;
this.next = next;
this.prev = prev;
}

public int getData() {
return data;
}

public void setData(int data) {
this.data = data;
}

public Node getNext() {
return next;
}

public void setNext(Node next) {
this.next = next;
}

public Node getPrev() {
return prev;
}

public void setPrev(Node prev) {
this.prev = prev;
}
}
```

```
public static Node quickSort(Node head) {

}
```

*// Задачка со двумя звёздочками ***
// You are given an array that consists of positive and negative integers. Find the sum of maximum subarray using divide and conquer
Subarray - any consequent array within array
arr = [1, 2, 3] has following subarrays:
[]
[1]
[2]
[3]
[1,2]
[2,3]
[1,2,3]

// Example:
arr = [0, -5, -3, 10, 9, -11, 17, -22]
// Result: 25 (= 10 + 9 - 11 + 17)
//
// Example: arr = [8, -9, 11, -15, 9, -5, 6, -1, 3, 4]
// Result: 16 (= 9 - 5 + 6 - 1 + 3 + 4)

```
public static int findTheMax(int[] arr) {;
}
```

Recursion and dynamic programming

Баллов: 0 из 0.

Ниже представлены задачи трёх уровней, обычные, с одной и
 двумя звёздочками. Можно выбрать Нужно решить одну любую задачу,
уровень сложности выбираете самостоятельно

// Given a number representing a distance.
The task is to count total number of possible ways to cover the distance with 1, 2 and 3 steps.
// Example: n = 3
// Result: 4
// Notes:
// 1 + 1 + 1
// 1 + 2
// 2 + 1
// 3
//
// Example: n = 4
// Result: 7
// Notes:
// 1 + 1 + 1 + 1
// 1 + 2 + 1
// 2 + 1 + 1
// 1 + 1 + 2
// 2 + 2
// 3 + 1
// 1 + 3

```
public static int coverDistance(int n) {

}
```

// Given an integer array representing coins
// You can consider each coin can be obtained infinite number of times
//
// You have to find the optimal way to make sum by using different combinations of coins.

// Example: sum = 4, coins[] = {1,2,3},
// Optimal solutions: {2, 2} or {1, 3}

```
public static int[] findCoins(int[] arr, int sum) {

}
```

*// Задачка со звёздочкой **
// Given an integer array representing coins
// You can consider each coin can be obtained infinite number of times
//
// You have to find the all ways to make sum by using different combinations of coins.

// Example: sum = 4, coins[] = {1,2,3},
// Result: {1, 1, 1, 1} or {1, 1, 2} or {2, 2} or {1, 3}.

```
public static int[] findCoins(int[] arr, int sum) {

}
```

*// Задачка со двумя звёздочками ***
// Given an integer array representing coins
// You can consider each coin can be obtained only one time
// You are given k the number of coins that should be returne
// You have to find the all ways to make sum by using different combinations of coins.

// Example: sum = 4, coins[] = {1,1,1,2,3}, k = 3
// Result: {1, 1, 2}

```
public static int[] findCoins(int[] arr, int sum, int k) {

}
```

Data structures

Баллов: 0 из 0.

```
// Validate brackets sequence given as string

// Example (())
// Result: true

// Example (()()
// Result: false

// Example )()(
// Result: false

public static boolean validate(String sequence) {

}
```

```
// Задачка со звёздочкой *
// Validate arithmetic expression with numbers and + - * /

// Example 4+5-6*6
// Result: true

// Example 4+-5-6*6
// Result: false

// Example -4/6//6+1-2
// Result: false

public static boolean validate(String sequence) {

}
```

// Задачка со звёздочкой *
You are given a singly linked list where each node can contain a child list (which is also a singly linked list). Your task is to transform that structure to the flat singly linked list so each node will have no child lists

Example:
Node(5) -> Node(3) -> Node(6) -> Node(2)
child of Node(5) is Node(1) -> Node(7)
Node(3) has no children
child of Node(6) is Node(9) -> Node(11)
child of Node(2) is Node(8) -> Node(0)

result
Node(5) -> Node(1) -> Node(7) -> Node(3) -> Node(6) -> Node(9) -> Node(11) -> Node(2) -> Node(8) -> Node(0)

Example:
Node(5) -> Node(3)
child of Node(5) is Node(1) -> Node(7)
child of Node(1) is Node(9) -> Node(11)
child of(7) is Node -> 8
(3) has no childrenNode

result
5) -> -> -> Node(1) -> Node(9) -> Node((11)Node7)Node(8) -> Node(3)Node(

```
class Node {
int data;
Node next;
Node child;

public Node(int data, Node next, Node child) {
this.data = data;
this.next = next;
this.child = child;
}

public int getData() {
return data;
}

public void setData(int data) {
this.data = data;
}
public Node getNext() {
return next;
}

public void setNext(Node next) {
this.next = next;
}

public Node getChild() {
return child;
}

public void setChild(Node child) {
this.child = child;
}
}ansfor
```

○ Option 1

```
// Задачка с двумя звёздочками **
// Validate arithmetic expression with numbers and + - * / and brackets

// Example 4+5-6*6
// Result: true

// Example (4+)5-6*6
// Result: false

// Example (-4/6/(6(2)
// Result: false

public static boolean validate(String sequence) {

}
```

*// Задачка с двумя звёздочками ***
Реализайте очередь на основе структуры данных Stack.

```
class Queue {


    // Добавляем элемент в очередь
    public void enqueue(int data)


    }


    // Удалить элемент из queue
    public T dequeue()
    {


    }
}
```