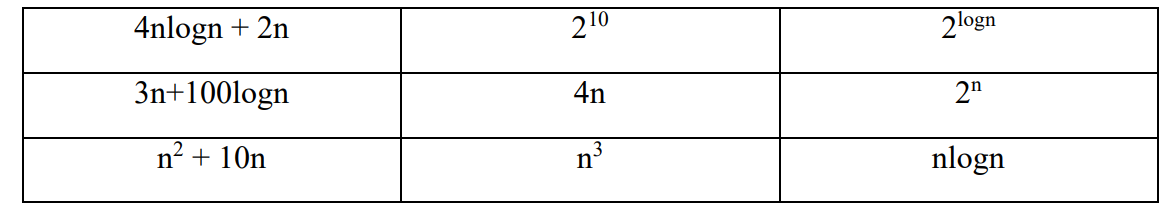
HOMEWORK LECTURE

Complexity analyses

1. Sort the following functions in the ascending order of Big O notation:



SOLUTION:

|  |  |  |
| --- | --- | --- |
| O(NLOGN) | O(1) | O(2LOGN) |
| O(N) | O(N) | O(2N) |
| O(N2) | O(N3) | O(NLOGN) |

SORTED SEQUENCE: O(1) –> O(2LOGN) -> O(N) -> O(NLOGN) -> O(N2) - > O(N3) -> O(2N).

1. Given an integer number n, your task is to write two different algorithms in pseudo-codes to calculate 2n , and evaluate the complexity of the algorithms.

SOLUTION :

+Algorithm1 :

Initialize ans = 1;

FOR(i,1,N) ans = 2 \* ans;

complexity of the algorithms : O(N)

+Algorithm2 :

Đệ quy: 2n = 2n/2 \* 2n/2 (nếu n chẵn)

2n = 2n/2 \* 2n/2 \* 2 (nếu n lẻ)

complexity of the algorithms : O(logN)

1. Your task is to write operations of queue data structure in pseudo-codes using an array, then evaluate the complexities of the operations.

SOLUTION:

QUEUE {

Queue(int size) {

capacity = size;

arr = new int[size];

front = rear = -1;

}

//KIỂM TRA QUEUE RỖNG

bool isEmpty() {

Return front == -1;

} -> O(1)

//KIỂM TRA QUEUE ĐẦY

bool isFull() {

Return (rear + 1) % capacity == front;

}

->O(1)

//THÊM MỘT PHẦN TỬ VÀO QUEUE

Void enqueue(int value) {

If(isFull()) return;

If(isEmpty) {

Front = rear = 0;

} else {

Rear = (rear + 1) % capacity;

}

Arr[rear] = value;

Size++;

}

->O(1)

// LOẠI BỎ PHẦN TỬ ĐẦU TIÊN KHỎI QUEUE

int dequeue() {

If(isEmpty) return -1;

If(front = rear) front = rear = -1;

Else front = (front + 1) % capacity;

}

->O(1)

}

1. Your task is to write operations of queue data structure in pseudo-codes using a linked list, then evaluate the complexities of the operations.

SOLUTION:

ENQUEUE(value)

newNode.value = value;

if(front == null) front = newNode;

rear = front;

else

rear.next = newNode;

rear = newNode;

->O(1)

DEQUEUE()

TMP = FRONT;

TMP = TMP.NEXT;

FRONT = TMP;

->O(1)

1. Your task is to write operations of stack data structure in pseudo-codes using an array, then evaluate the complexities of the operations.

SOLUTION:

Void PUSH(value)

Arr[ n ] = value;

->O(1);

Void POP()

Arr[n-1] = arr[n] ->O(1)

1. Your task is to write operations of stack data structure in pseudo-codes using a linked list, then evaluate the complexities of the operations.

Void PUSH(value)

newNode.value = value;

newNode.next = head;

head = newNode

->O(1)

Void POP()

Head = head - > next;

->O(1)