(\*) Log mặc định là cơ số 2!

**1. Arranging:**

**2^10** (hằng số) **-> 2^log(n)** (=n**) -> 3n + 100logn** (logn < n => <=4n) **-> 4n -> nlogn -> 4nlogn + 2n**

**-> n^2 +** **10 -> n^3.**

**2. Pseudo-codes to calculate 2^n:**

**1)** int n, result = 1;

For (i = 0 to n-1, i++): result \*= 2;

cout << result;

* O(n)

**2)** int pow(int x, int n) {

int result = 1;

while (n>0) {

if n%2==1 then result \*= a;

a\*=a;

n/=2;

}

return result; }

* O(log(n))

**3. Operations of queue data structure using array:**

stuct Queue {

int front, rear, capacity;

int \*queue;

Queue(int c) {

front = rear = 0;

capacity = c;

queue = new int;

}

void Enqueue(int data) {

if capacity != rear then // the queue isn’t full.

{queue[rear] = data;

rear++;

}

void DeQueue(){

if front = rear then //the queue is null.

cout << “null”;

} else {

for int i = 0; i < rear – 1; i++: queue[i] = queue[i+1]; //lap n-1 lan

rear--;

}

}

* O(n)

**4**. **Operations of queue data structure using linked list:**

struct Node {

int data;

Node \*next;

Node (int data) {

this.data = data;

next = null;

}

}

Struct Queue {

Node \*front, \*rear;

Queue() {front = rear = null}

enQueue(int data) {

// create new node

Node \*temp = new Node(data);

if rear = null then front = rear = temp, return //queue is null

rear->next = temp;

rear = temp;

//add new node to the end of list

}

deQueue(){

if front = null then return //queue is null

//delete last element of the list

Node \*temp = front;

front = front->next;

if (front = null) then rear = null; // the queue has only 1 element

delete(temp);

}

}

* O(1) -> nhanh hơn bằng mảng

**5. Operations of stack data structure using array:**

struct Stack {

int top, capacity;

int \*stack;

Stack(int c) {

top = -1;

capacity = c;

stack = new int;

}

void push (int x) {

if top > = capacity -1 then print: “stack overflow”

top++;

stack[top] = x;

void pop () {

if top = -1 then print “there is nothing to pop”

top--;

}

}

* O(1)

**6.** **Operations of stack data structure using linked list:**

struct Node {

int data;

Node \*next;

}

Node \*newNode(int data) {

Node \*newNode = new Node();

newNode->data = data;

newNode->next = null;

return newNode;

}

void push(Node \*\*root, int data)

Node \*temp = newNode(data);

temp->next = \*root;

\*root = temp;

//add to head

void pop(Node \*\*root)

Node \*temp = \*root;

\*root = \*root->next;

free(temp);

// delete head

* O(1)