**Bài 1:**

O(2logn)

O(2n)

O(3n+100logn)

O(4n)

O(4nlogn + 2n)

O(n2 + 10n)

O(nlogn)

O(n3)

**Bài 2:**

C1 :

def calculate\_2n\_1(n): {

if n == 0:

return 1;

else:

return 2 \* calculate\_2n\_1(n - 1);

} Độ phức tạp O(1)

C2:

def calculate\_2n\_2(n): {

result = 1

while n > 0: {

result \*= 2;

n --;

}

return result;

} Độ phức tạp O(1)

**Bài 3:**

class Queue {

public:

Queue(int size) {

array = new int[size];

front = 0;

rear = 0;

}

~Queue() {

delete[] array;

}

bool is\_empty() {

return front == rear;

}

bool is\_full() {

return rear == size;

}

void enqueue(int element) {

if (is\_full()) {

return;

}

array[rear] = element;

rear++;

}

int dequeue() {

if (is\_empty()) {

return INT\_MIN;

}

int element = array[front];

front++;

return element;

}

int front() {

if (is\_empty()) {

return INT\_MIN;

}

return array[front];

}

private:

int\* array;

int front;

int rear;

};

Độ phức tạp O(1)

**Bài 4:**

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

this->next = nullptr;

}

};

class Queue {

public:

Queue() {

head = nullptr;

tail = nullptr;

}

~Queue() {

while (head != nullptr) {

Node\* next = head->next;

delete head;

head = next;

}

}

bool is\_empty() {

return head == nullptr;

}

void enqueue(int element) {

Node\* new\_node = new Node(element);

if (head == nullptr) {

head = new\_node;

tail = new\_node;

} else {

tail->next = new\_node;

tail = new\_node;

}

}

int dequeue() {

if (head == nullptr) {

return INT\_MIN;

} else {

int element = head->data;

Node\* next = head->next;

delete head;

head = next;

if (head == nullptr) {

tail = nullptr;

}

return element;

}

}

int front() {

if (head == nullptr) {

return INT\_MIN;

} else {

return head->data;

}

}

private:

Node\* head;

Node\* tail;

};  
độ phức tạp O(1)

**Bài 5:**

class Stack {

public:

Stack(int size) {

array = new int[size];

top = -1;

}

~Stack() {

delete[] array;

}

bool is\_empty() {

return top == -1;

}

bool is\_full() {

return top == array.size() - 1;

}

void push(int element) {

if (is\_full()) {

// Stack is full.

return;

}

top++;

array[top] = element;

}

int pop() {

if (is\_empty()) {

// Stack is empty.

return INT\_MIN;

}

int element = array[top];

top--;

return element;

}

int top() {

if (is\_empty()) {

return INT\_MIN;

}

return array[top];

}

private:

int\* array;

int top;

};

Độ phức tạp O(1)

**Bài 6:**

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

this->next = nullptr;

}

};

class Stack {

public:

Stack() {

head = nullptr;

}

~Stack() {

while (head != nullptr) {

Node\* next = head->next;

delete head;

head = next;

}

}

bool is\_empty() {

return head == nullptr;

}

void push(int element) {

Node\* new\_node = new Node(element);

new\_node->next = head;

head = new\_node;

}

int pop() {

if (is\_empty()) {

return INT\_MIN;

}

int element = head->data;

Node\* next = head->next;

delete head;

head = next;

return element;

}

int top() {

if (is\_empty()) {

return INT\_MIN;

}

return head->data;

}

private:

Node\* head;

};

Độ phức tạp O(1)