Nhóm 3:

Mai Nhất Tùng - 21520523

Hà Trọng Tài - 21520436

Nguyễn Ngọc Nhung – 21521248

BÀI LÀM

10. The ***range*** of a finite nonempty set of n real numbers S is defined as the difference between the largest and smallest elements of S. For each representation of S given below, describe in English an algorithm to compute the range. Indicate the time efficiency classes of these algorithms using the most appropriate notation (O, Θ, or Ω).

a. An unsorted array

b. A sorted array

c. A sorted singly linked list

d. A binary search tree

a) An unsorted array (Mảng chưa sắp xếp)

- Dùng một vòng for các phần tử trong mảng để tìm ra phần tử nhỏ nhất và lấn nhất của mảng đó. Lấy số lớn nhất trừ số nhỏ nhất sẽ ra được phạm vi

Mã giả:

```
for i ← 0 to n-1 do
    if giá trị lớn nhất < giá trị mảng thứ i
        cập nhật giá trị lớn nhất
    if giá trị nhỏ nhất > giá trị mảng thứ i
        cập nhật giá trị nhỏ nhất
Phạm vi ← (giá trị lớn nhất - giá trị nhỏ nhất)
```

- Độ phức tạp:
  - O(n)
  - Ω(n)
  - Θ(n)

b) A sorted array(Mảng đã sắp xếp)

- Lấy trị tuyệt đối giá trị đầu trừ giá trị cuối trong mảng sẽ có được phạm vi của mảng

Mã giả

```
Phạm vi ← |giá trị đầu tiên của mảng - giá trị cuối cùng của mảng|
```

- Độ phức tạp:
  - O(1)
  - Ω(1)
  - Θ(1)

c) A sorted singly linked list(Danh sách liên kết đơn đã sắp xếp)

- Ta phải thực hiện đi qua từng node để đi đến node cuối và lấy giá trị node đầu trừ node cuối

Mã giả:

```
p ← List.head
q ← null

// lấy giá trị cuối
While p do
        q = p
        p = p->next

Phạm vi <- |q->value – List.head->value|
```

- Độ phức tạp:
  - O(n)
  - Ω(n)
  - Θ(n)

d) A binary search tree (Cây tìm kiếm nhị phân)

- Do cây nhị phân tìm kiếm có cấu trúc rất đặc biệt nên ta có cây con bên trái luôn lưu giá trị nhỏ hơn và cây con bên trái luôn lưu giá trị lớn hơn.
- Duyệt cây con bên trái ngoài cùng để tìm ra giá trị nhỏ nhất.
- Duyệt cây con bên phải ngoài cùng để tìm ra giá trị lớn nhất.

Mã giả:

```
// tìm giá trị nhỏ nhất
while node hiện tại có node con bên trái khác NULL then
```

liên kết đến node con bên trái
giá trị nhỏ nhất ← giá trị của node hiện tại
// tìm giá trị lớn nhất
while node hiện tại có node con bên phải khác NULL then
liên kết đến node con bên phải
giá trị lớn nhất ← giá trị của node hiện tại
Phạm vi ← (giá trị lớn nhất - giá trị nhỏ nhất)

- Độ phức tạp:
  - O(n) (Trong trường hợp cây mất cân bằng)
  - Ω(logn)
  - Θ(logn)

11. *Lighter or heavier?* You have $n > 2$ identical-looking coins and a two-pan balance scale with no weights. One of the coins is a fake, but you do not know whether it is lighter or heavier than the genuine coins, which all weigh the same. Design a $\Theta(1)$ algorithm to determine whether the fake coin is lighter or heavier than the others.

- Các bước thực hiện
  - Bước 1: Chia n đồng xu thành 4 nhóm: (nhóm 1: n/3 đồng xu, nhóm 2: n/3 đồng xu, nhóm 3: n/3 đồng xu, nhóm 4: các đồng xu còn lại)
  - Bước 2: Cân nhóm 1 với nhóm 2
  - Bước 3: Cân nhóm 1 với nhóm 3
- Kết quả thu được:
- Cân bằng cả 2 lần: các đồng xu trong 3 nhóm 1, 2, 3 đều là thật, đồng xu giả nằm trong nhóm 4. Trong những đồng xu thật, chọn ra số đồng xu bằng số đồng xu của nhóm 4, đem cân với nhóm 4 để so sánh.
- Có ít nhất 1 lần cân bị lệch: trong 3 nhóm sẽ có 1 nhóm có trọng lượng khác 2 nhóm còn lại, nhóm đó chứa đồng xu giả
- Cụ thể:
  - Lần 1 cân cân bằng, lần 2 cân lệch: đồng xu giả nằm trong nhóm 3
  - Lần 1 cân lệch, lần 2 cân bằng: đồng xu giả nằm trong nhóm 2
  - Lần 1 cân lệch, lần 2 cân lệch: đồng xu giả nằm trong nhóm 1)

➔ **Kết luận**: Nếu nhóm có đồng xu giả nặng hơn thì kết luận đồng xu giả nặng hơn; ngược lại kết luận đồng xu giả nhẹ hơn

ALGORITHM $GE(A[0..n-1, 0..n])$

```
//Input: An n × (n + 1) matrix A[0..n − 1, 0..n] of real numbers
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        for k ← i to n do
            A[j, k] ← A[j, k] − A[i, k] ∗ A[j, i] / A[i, i]
```

a. Find the time efficiency class of this algorithm.

b. What glaring inefficiency does this pseudocode contain and how can it be eliminated to speed the algorithm up?

a) Độ phức tạp của từng dòng for là:
   - For biến i: O(n)
   - For biến j: O(n)
   - For biến k: O(n)

➔ Độ phức tạp của thuật toán trên: $O(n^3)$

b) Ta có khi k = i

⇔ A[j,i] = A[j,i] – A[i, i] * A[j,i] / A[i, i]

= A[j,i] – A[j,i] = 0

➔ A[j,k] = A[j, k]

- Vậy bài toán ở đây chỉ là chúng ta gán các phần tử A[j,i] = 0

Mã giả

```
for i ← 0 to n – 2 do
    for j ← i + 1 to n – 1do
        A[j, i] ← 0
```

⇨ Độ phức tạp giảm xuống còn $O(n^2)$