## Họ và tên : Lương Thiện Chí

## MSSV: 1610304

**Bài tập : phân tích độ phức tạp trong trường hợp trung bình với tác vụ được chọn là so sánh của giải thuật lặp cho bài toán kiểm tra mảng có các phần tử duy nhất không.**

- Thông số để thiết lập hàm: n
- Trường hợp phân tích: trung bình
- Tác vụ để phân tích: so sánh
- Tính toán độ phức tạp: Kiểm tra từng phần tử trong dãy xem liệu có xảy ra trùng lặp hay không.

+) Tại vị trí đầu tiên trong dãy (i1), so sánh với n-1 phần tử còn lại

Số lần so sánh để tìm i1 tại vị trứ thứ 2 là 1

Số lần so sánh để tìm i1 tại vị trứ thứ 3 là 2

….

Số lần so sánh để tìm i1 tại vị trứ thứ n là n-1

+) Tại vị trí thứ 2 trong dãy (i2), so sánh với n-2 phần tử còn lại

Số lần so sánh để tìm i2 tại vị trứ thứ 3 là n-1 + 1 = n

Số lần so sánh để tìm i2 tại vị trứ thứ 4 là n-1 + 2 = n + 1

….

Số lần so sánh để tìm i2 tại vị trứ thứ n là n-1 + n-2 = 2n-3

….

+) Tại vị trí thứ n-1 trong dãy (i(n-1)), so sánh với 1 phần tử còn lại

Số lần so sánh để tìm i(n-1) tại vị trứ thứ n là: n-1+ n-2 +… + 1 = n(n-1)/2

Xác suất để xảy ra bất kì 2 phần tử trùng nhau trong mảng là : p = 2/[n(n-1)]

Vậy độ phức tạp giải thuật cho trường hợp trung bình là :

$C_{avg}(n)$

= 2/[n(n-1)] * [1+2+3+…n(n-1)/2]

= 2/[n(n-1)] * [n(n-1)/2 * (n(n-1)/2 + 1) / 2]

= [n(n-1)/2+1] / 2

$= (n^2 – n + 2)/ 4$

**Bonus: định nghĩa giải thuật đệ quy tương đương với giải thuật lặp của bài toán**

Ngôn ngữ C++

```
bool Unique(int a[], int n){
        if (n<=1) return true;
        else{
                for(j=0; j < n-1;j++){
                        if(a[n-1] == a[n]) return false;
                }
                return Unique(a, n-1);
        }
}
```

**Bài tập được mở rộng: với phần cải thiện giải thuật để độ phức tạp ít hơn n*(n-1)/2 và phần phân tích độ phức tạp của giải thuật mới. Các giải thuật phải được hiện thực với ngôn ngữ lập trình cụ thể.**

Dựa vào giải thuật quick sort:

```
void swap(int &a, int &b)
{
   int t = a;
   a = b;
   b = t;
}


int partition (int arr[], int low, int high)
{
   int pivot = arr[high];    // pivot
   int left = low;
   int right = high - 1;
   while(true){
      while(left <= right && arr[left] < pivot) left++;
      while(right >= left && arr[right] > pivot) right--;
```

```
        if (left >= right) break;
        swap(arr[left], arr[right]);
        left++;
        right--;
    }
    swap(arr[left], arr[high]);
    return left;
}

/* Hàm thực hiện giải thuật quick sort */
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        /* pi là chỉ số nơi phần tử này đã đứng đúng vị trí
         và là phần tử chia mảng làm 2 mảng con trái & phải */
        int pi = partition(arr, low, high);

        // Gọi đệ quy sắp xếp 2 mảng con trái và phải
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
bool unique(int arr[], int n){

            quickSort(arr, 0, n-1);

            for (int i=0;i<n-1; i++){

                        if (arr[i] == arr[i+1]){

                                    return false;

                        }

            return true;

            }

}
```

Phân thích độ phức tạp của giải thuật:

- Thông số để thiết lập hàm : n
- Trường hợp phân tích: xấu nhất
- Tác vụ để phân tích: so sánh
- Độ phức tạp giải thuật quick sort là O(nlog(n))

⇨ Độ phức tạp của giải thuật Unique là: O(n.log(n)) + n