**Bài 5**

a)Thiết kế một thuật toán chia để trị để tìm số bị thiếu.

\*Hàm kiểm tra xem mảng có đủ các số nguyên từ 0 đến n hay không

function isComplete(arr, n):

sum = 0

=>Tính tổng của tất cả các phần tử trong mảng

for num in arr:

sum = sum + num

=> Tính tổng mong đợi nếu mảng đầy đủ các số từ 0 đến n

expected\_sum = n \* (n + 1) / 2

=>So sánh tổng thực và tổng mong đợi

if sum == expected\_sum:

return true

=>Trả về true nếu mảng đủ các số từ 0 đến n

else:

return false

=> Trả về false nếu mảng không đủ các số từ 0 đến n

\*Hàm chia để trị để tìm số bị thiếu

function findMissing(arr, start, end):

=>Nếu chỉ số bắt đầu lớn hơn chỉ số kết thúc, tức là không còn phần tử nào để xem xét

if start > end:

return -1

=> Trả về -1 để chỉ ra rằng không có số bị thiếu

mid = (start + end) / 2

=> Tìm chỉ số giữa của mảng

if isComplete(arr, mid) is false:

=> Kiểm tra nếu phần bên trái không đủ số nguyên từ 0 đến mid

return findMissing(arr, start, mid - 1)

=> Tiếp tục tìm số bị thiếu trong phần bên trái của mảng

else:

=> Nếu phần bên trái đủ số nguyên từ 0 đến mid, số bị thiếu nằm trong phần bên phải

return findMissing(arr, mid + 1, end)

=> Tiếp tục tìm số bị thiếu trong phần bên phải của mảng

\*Giải thích :

Hàm isComplete được sử dụng để kiểm tra xem mảng có đủ các số nguyên từ 0 đến n hay không bằng cách tính tổng của tất cả các phần tử trong mảng và so sánh với tổng mong đợi.

Hàm findMissing sử dụng phương pháp chia để trị để tìm số bị thiếu trong mảng. Nó chia mảng thành các phần bên trái và bên phải và kiểm tra xem phần bên trái có đủ các số từ 0 đến mid không. Nếu không, nó đệ quy tìm kiếm trong phần bên trái; ngược lại, nó tìm kiếm trong phần bên phải.

b) Để viết phương trình truy hồi cho độ phức tạp của thuật toán, chúng ta sẽ xem xét cách thuật toán hoạt động. Thuật toán chia mảng thành hai phần bằng cách sử dụng một điểm chia (giữa), sau đó tiếp tục tìm kiếm trong một phần của mảng. Mỗi lần chia mảng, kích thước của mảng giảm xuống một nửa.

Gọi T(n) là độ phức tạp của thuật toán với đầu vào có kích thước là n.

Trong mỗi lần chia, thuật toán tốn O(1) để tính chỉ số giữa (mid) và gọi hàm kiểm tra isComplete.

Hàm isComplete sẽ tốn O(n) để tính tổng của các phần tử trong mảng.

Nếu mảng được chia đều mỗi lần, kích thước của mảng giảm xuống một nửa sau mỗi bước.

Do đó, phương trình truy hồi cho độ phức tạp của thuật toán có thể được viết như sau:

T(n) = T(n/2) + O(n)

Phần tử O(n) xuất phát từ hàm isComplete, trong đó mỗi phần tử của mảng cần được duyệt qua để tính tổng.

Độ phức tạp của nó là O(nlogn) do mỗi lần chia đều giảm kích thước của mảng xuống một nửa.