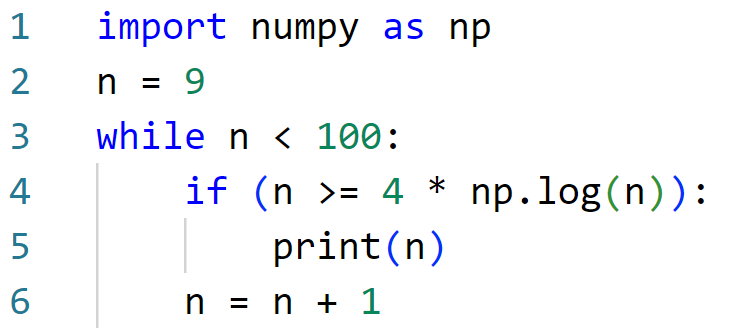
**Luyện tập thêm 1: về độ phức tạp của thuật toán**

**Đỗ Mạnh Hùng – 20002053 – K65A5**

**Phần 1: Reinforcement**

**R – 4.2:** The number of operations executed by algorithms A and B is 8nlogn and , respectively. Determine such that A is better than B for .

* Bằng cách sử dụng lập trình:



* Nhận thấy rằng biểu thức luôn đúng khi

**R – 4.3:** The number of operations executed by algorithms A and B is and , respectively. Determine such that A is better than B for .

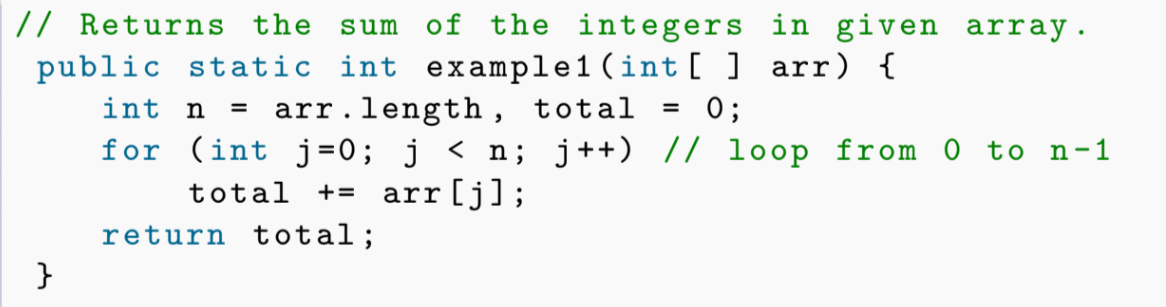
**R – 4.6:**

* Áp dụng công thức tổng các số hạng của cấp số cộng với số số hạng là n, công sai d = 2, :

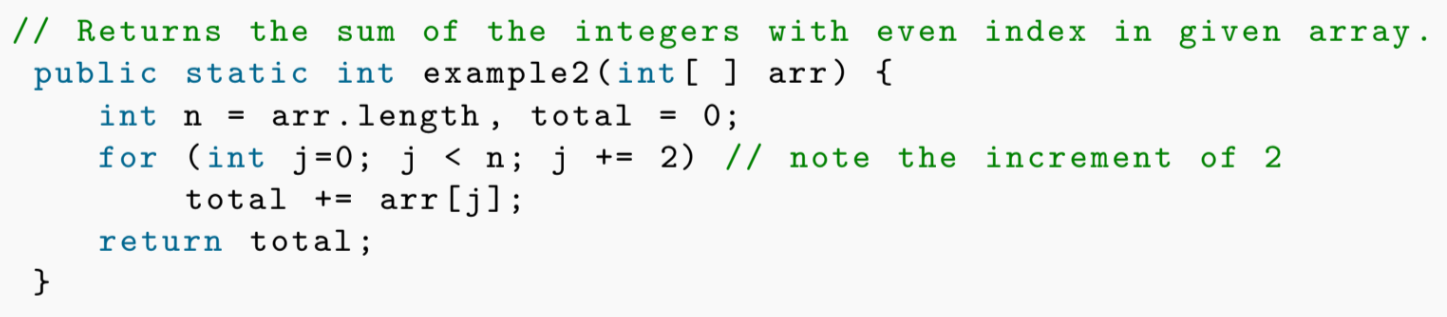
**R – 4.8:**

-

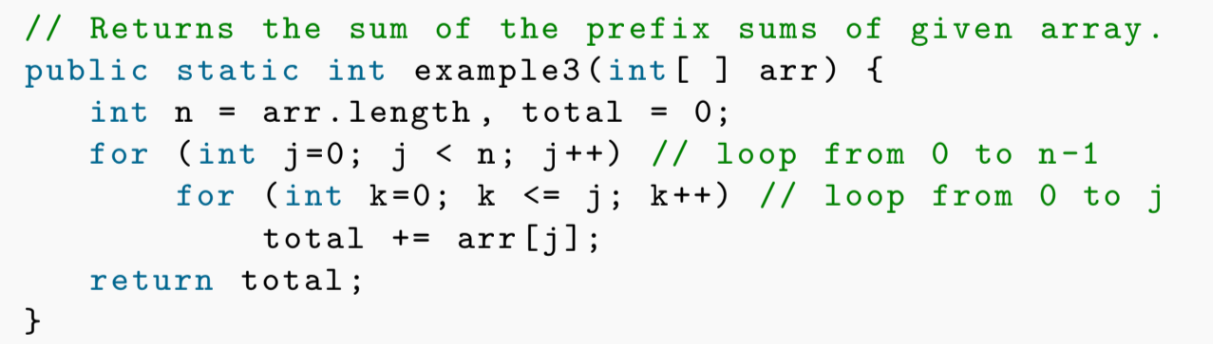
**R – 4.9:** Give a big-Oh characterization, in terms of n, of the running time of the example1 method shown in Code Fragment 4.12.



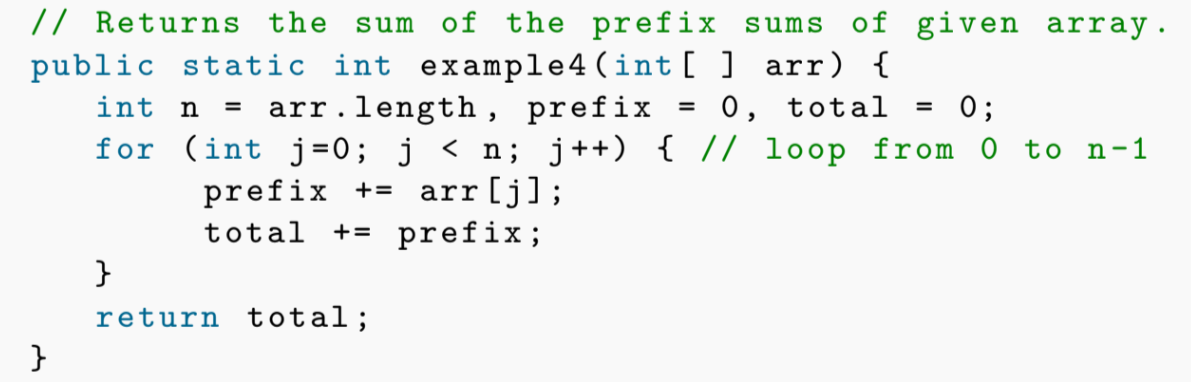
**R – 4.10:** Give a big-Oh characterization, in terms of n, of the running time of the example2 method shown in Code Fragment 4.12.



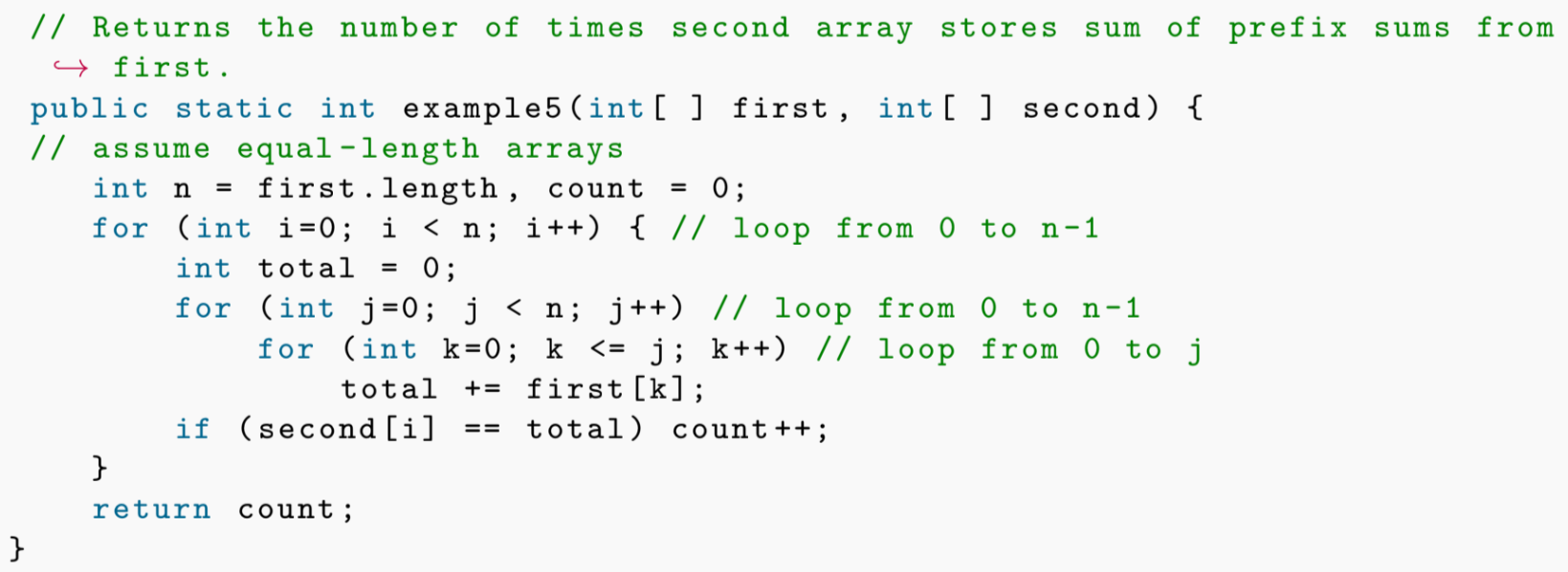
**R – 4.11:** Give a big-Oh characterization, in terms of n, of the running time of the example3 method shown in Code Fragment 4.12.



**R – 4.12:** Give a big-Oh characterization, in terms of n, of the running time of the example4 method shown in Code Fragment 4.12.



**R – 4.13:** Give a big-Oh characterization, in terms of n, of the running time of the example5 method shown in Code Fragment 4.12.



**Phần 2: Creativity**

**C – 4.2:** Describe an efficient algorithm for finding the ten largest elements in an array of size n. What is the running time of your algorithm?

* Tạo một mảng ten\_larges\_element chứa 10 phần tử của mảng có kích thước n ban đầu.
* Bắt đầu từ vị trí thứ 10 trong mảng ban đầu, lặp từng phần tử và so sánh với phần tử nhỏ nhất trong mảng ten\_larges\_element

+ Nếu phần tử trong mảng ban đầu > phần tử nhỏ nhất trong ten\_larges\_element thì thay thế phần tử nhỏ nhất trong ten\_larges\_element = phần tử trong mảng ban đầu

+ Ngược lại lấy phần tử tiếp theo mảng ban đầu so sánh tiếp.

* Lặp đến khi kết thúc ta được mảng gồm 10 phần tử lớn nhất từ mảng ban đầu
* Thời gian chạy của thuật toán này là: O(n)

**C – 4.4:**

* Áp dụng công thức tổng n số tự nhiên đầu tiên:

**C – 4.6:** Determine the total number of grains of rice requested by the inventor of chess.

* Mô tả bài toán: Trên một bàn cờ 64 ô, ta thực hiện việc đặt gạo với ô đầu tiên 1 hạt và các ô tiếp theo số lượng hạt gạo gấp đôi ô trước. Ví dụ: 1, 2, 4, 8, 16, 32, … .

Nhận thấy đây là một cấp số nhân với công bội q = 2 và số hạng đầu tiên , ta thực hiện tính tổng của một cấp số nhân:

**C – 4.11:** An array A contains n − 1 unique integers in the range [0, n − 1], that is, there is one number from this range that is not in A. Design an O(n)-time algorithm for finding that number. You are only allowed to use O(1) additional space besides the array A itself.

* Khởi tạo mảng A có kích thước n - 1.
* Sử dụng phép xor và ứng dụng tính chất giao hoán của nó:  
  + Vòng lặp đầu: Xor từng phần tử trong mảng A

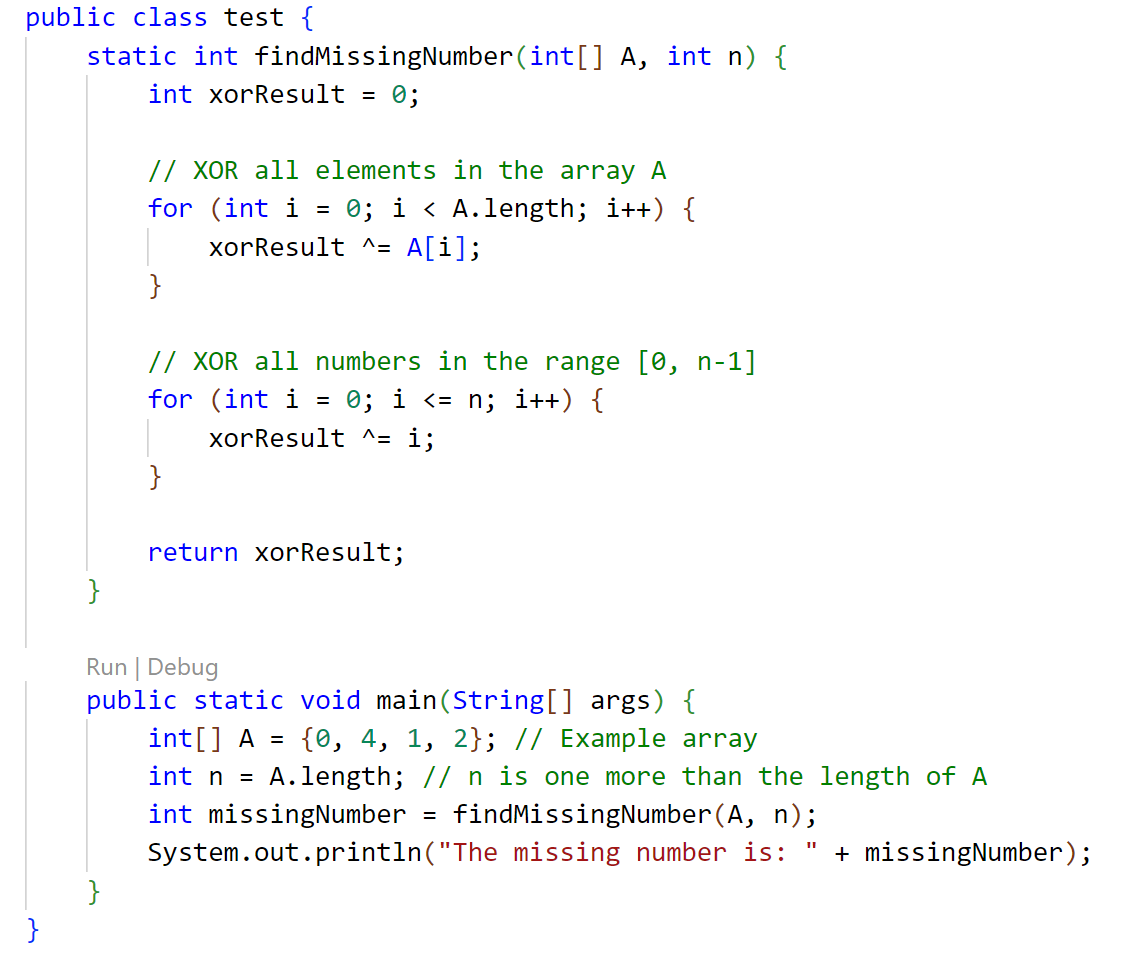
+ Vòng lặp tiếp theo: Xor từng số trong phạm vi [0, n - 1] với kết quả xor phía trước. Các giá trị trùng nhau sẽ triệt tiêu lẫn nhau chỉ còn giá trị không xuất hiện cần tìm.

* Ví dụ minh họa: ta có mảng A{1, 4, 2, 0} (chứa 4 giá trị không trùng nhau trong phạm vi [0, 4])

+ Xor từng phần tử trong mảng A: 1 xor 4 xor 2 xor 0 = xorResult.

+ Xor từng số trong phạm vi [0, 4] với xorResult:   
xorResulr xor 0 xor 1 xor 2 xor 3 xor 4 = 1 xor 4 xor 2 xor 0 xor 0 xor 1 xor 2 xor 3 xor 4 = 0 xor 0 xor 1 xor 1 xor 2 xor 2 xor 4 xor 4 xor 3 = 3.

* Code thực hiện:



**C – 4.22:** An array A contains n integers taken from the interval [0, 4n], with repetitions allowed. Describe an efficient algorithm for determining an integer value k that occurs the most often in A. What is the running time of your algorithm?

* Tạo mảng count\_accurs có kích thước 4n + 1 có các giá trị ban đầu là 0 nhằm lưu số lần xuất hiện của số nào đó trong mảng A (chỉ số vị trí của mảng coutn\_accurs là các số có khả năng xuất hiện trong A)
* Duyệt qua các giá trị trong mảng A và đếm các giá trị trương ứng. Mỗi khi đếm một giá trị trong mảng A ta sẽ lưu vào mảng count\_accurs với vị trí là giá trị mảng A và giá trị là số lần xuất hiện giá trị đó trong mảng A.
* Sau khi đếm lưu vào vị trí tương ứng trong count\_accurs.
* Sau khi duyệt hết các 4n + 1 vị trí ta được mảng count\_accurs với các giá trị là số lần xuất hiện tương ứng với các chỉ số. lấy ra giá trị lớn nhất và chỉ số mảng tại giá trị lớn nhất đó là số k cần tìm.
* Thời gian thực hiện:   
  + Khi duyệt n giá trị trong mảng A ta có O(n)

+ Khi duyệt 4n + 1 giá trị trong mảng count\_accurs ta có O(4n + 1)

* Code thực hiện:

