**Homework 2**

**Group members: Nguyen Van Ninh, Ha Vu Long, Dam Thi Hong Nhung**

**Exercise 2.1**

**1.** Đối với mỗi thuật toán sau, chỉ ra (i) số liệu kích thước tự nhiên cho các đầu vào của nó, (ii) hoạt động cơ bản của nó và (iii) liệu số lượng hoạt động cơ bản có thể khác nhau đối với các đầu vào có cùng kích thước hay không:

a. tính toán trung bình của n số

b. tính toán n / n!

c. tìm phần tử nhỏ nhất trong danh sách n số

d. đảo ngược hiển thị danh sách n số

e. đảo ngược danh sách n số

f. thuật toán pen-and-pencil để cộng hai số nguyên thập phân có n chữ số

**Solve**

a. (i) n; (ii) phép cộng hai số; (iii) không

b. (i) độ lớn của n là số bit trong biểu diễn nhị phân của nó; (ii) phép nhân hai số nguyên; (iii) không

c. (i) n; (ii) so sánh hai số; (iii) không

d. (i) độ lớn của hai số đầu vào lớn hơn, hoặc độ lớn của số nhỏ hơn của hai số đầu vào hoặc tổng độ lớn của hai số đầu vào; (ii) phép chia modul; (iii) có

e. (i) độ lớn của n là số bit trong biểu diễn nhị phân của nó; (ii) loại bỏ một số khỏi danh sách, các số còn lại trở thành số nguyên tố; (iii) không

f. (i) n; (ii) phép nhân hai chữ số; (iii) không

**2.** a. Xem xét thuật toán dựa trên định nghĩa để tìm sự khác biệt giữa hai ma trận n × n. Hoạt động cơ bản của nó là gì? Nó được thực hiện dưới dạng một hàm của ma trận bậc n bao nhiêu lần? Là một hàm của tổng số phần tử trong ma trận đầu vào?

b. Trả lời các câu hỏi tương tự cho thuật toán dựa trên định nghĩa cho nghịch đảo của ma trận.

**Solve**

a. Phép cộng hai số. Nó được thực hiện n2 lần (một lần cho mỗi n2 phần tử trong ma trận đang được tính toán). Vì tổng số các phần tử trong hai ma trận đã cho là N = 2n2, tổng số phép cộng cũng có thể được biểu thị là n2 = N / 2.

b. Vì trên hầu hết các máy tính, phép nhân mất nhiều thời gian hơn phép cộng, nên phép nhân là lựa chọn tốt hơn để được coi là hoạt động cơ bản của thuật toán chuẩn cho phép nhân ma trận. Mỗi phần tử trong số n2 phần tử của tích của hai ma trận n x n được tính dưới dạng tích vô hướng của hai vectơ có kích thước n, yêu cầu n phép nhân. Tổng số phép nhân là n x n2 = n3 = (N /2)3/2.

**3.** Hãy xem xét một tìm kiếm tuần tự cổ điển quét danh sách để tìm kiếm các lần xuất hiện của một khóa tìm kiếm nhất định trong danh sách. Thảo luận về trường hợp xấu nhất, trường hợp trung bình và hiệu quả trường hợp tốt nhất của tìm kiếm tuần tự cổ điển.

**Solve**

Thuật toán này sẽ luôn thực hiện so sánh n khóa trên mọi đầu vào có kích thước n, trong khi con số này có thể thay đổi giữa n và 1 đối với phiên bản cổ điển của tìm kiếm tuần tự.

**4.**

***a. Chọn găng tay***: Trong ngăn kéo có 22 chiếc găng tay: 5 đôi găng tay màu đỏ, 4 đôi màu vàng và 2 đôi màu xanh lá cây. Bạn chọn găng tay trong bóng tối và chỉ có thể kiểm tra chúng sau khi đã chọn xong. Số lượng găng tay nhỏ nhất bạn cần chọn là bao nhiêu để có ít nhất một đôi phù hợp trong trường hợp tốt nhất? Trong trường hợp xấu nhất?

***b. Thiếu tất:*** Hãy tưởng tượng rằng sau khi giặt 5 đôi tất khác nhau, bạn phát hiện ra rằng thiếu hai chiếc tất. Tất nhiên, bạn muốn có số lượng cặp hoàn chỉnh lớn nhất còn lại. Do đó, bạn còn lại 4 cặp hoàn chỉnh trong trường hợp tốt nhất và 3 cặp hoàn chỉnh trong trường hợp xấu nhất. Giả sử rằng xác suất biến mất của mỗi chiếc trong 10 chiếc tất là như nhau, hãy tìm xác suất của trường hợp tốt nhất; xác suất của trường hợp xấu nhất; số lượng cặp bạn nên mong đợi trong trường hợp trung bình

**Solve**

a. Con số trường hợp tốt nhất là hai. Con số trong trường hợp xấu nhất là mười hai: nhiều hơn một lần so với số găng tay của người thuận tay.

b. Có hai kết quả có thể xảy ra ở đây: hai chiếc tất bị thiếu tạo thành một cặp (trường hợp tốt nhất) và hai chiếc bị thiếu không tạo thành một cặp (trường hợp xấu nhất). Tổng số kết quả khác nhau (các cách chọn còn thiếu tất) là 10C2 = 45. Số cách chọn đúng nhất là 5; do đó xác suất của nó là 5/45 = 1/9. Số trường hợp xấu nhất là 45 - 5 = 40; do đó xác suất của nó là 40/45 = 8/9. Trung bình, ta dự đoán có 4 x 1/9 + 3 x 8/9 = 28 cặp phù hợp.

**6.** Đề xuất cách bất kỳ thuật toán sắp xếp nào có thể được tăng cường theo cách để làm cho số lượng trường hợp tốt nhất của các so sánh chính của nó chỉ bằng n - 1 (tất nhiên, n là kích thước của danh sách). Bạn có nghĩ rằng nó sẽ là một bổ sung đáng giá cho bất kỳ thuật toán sắp xếp nào không?

**Solve**

Trước khi áp dụng thuật toán sắp xếp, hãy so sánh các phần tử liền kề của đầu vào của nó: nếu ai ≤ ai + 1 với mọi i = 0, ..., n - 2 thì dừng lại. Nói chung, nó không phải là một bổ sung đáng giá vì nó làm chậm thuật toán trên tất cả các đầu vào trừ những đầu vào rất đặc biệt. Lưu ý rằng một số thuật toán sắp xếp như là sắp xếp bong bóng và sắp xếp chèn, về bản chất kết hợp thử nghiệm này trong phần nội dung của thuật toán.

**8.** Đối với mỗi hàm sau đây, hãy cho biết giá trị của hàm sẽ thay đổi bao nhiêu nếu đối số của nó được tăng lên gấp ba lần.

a. nlog2n b. n c. n3 d. n2 e. n! f. 2n

**Solve**

a.

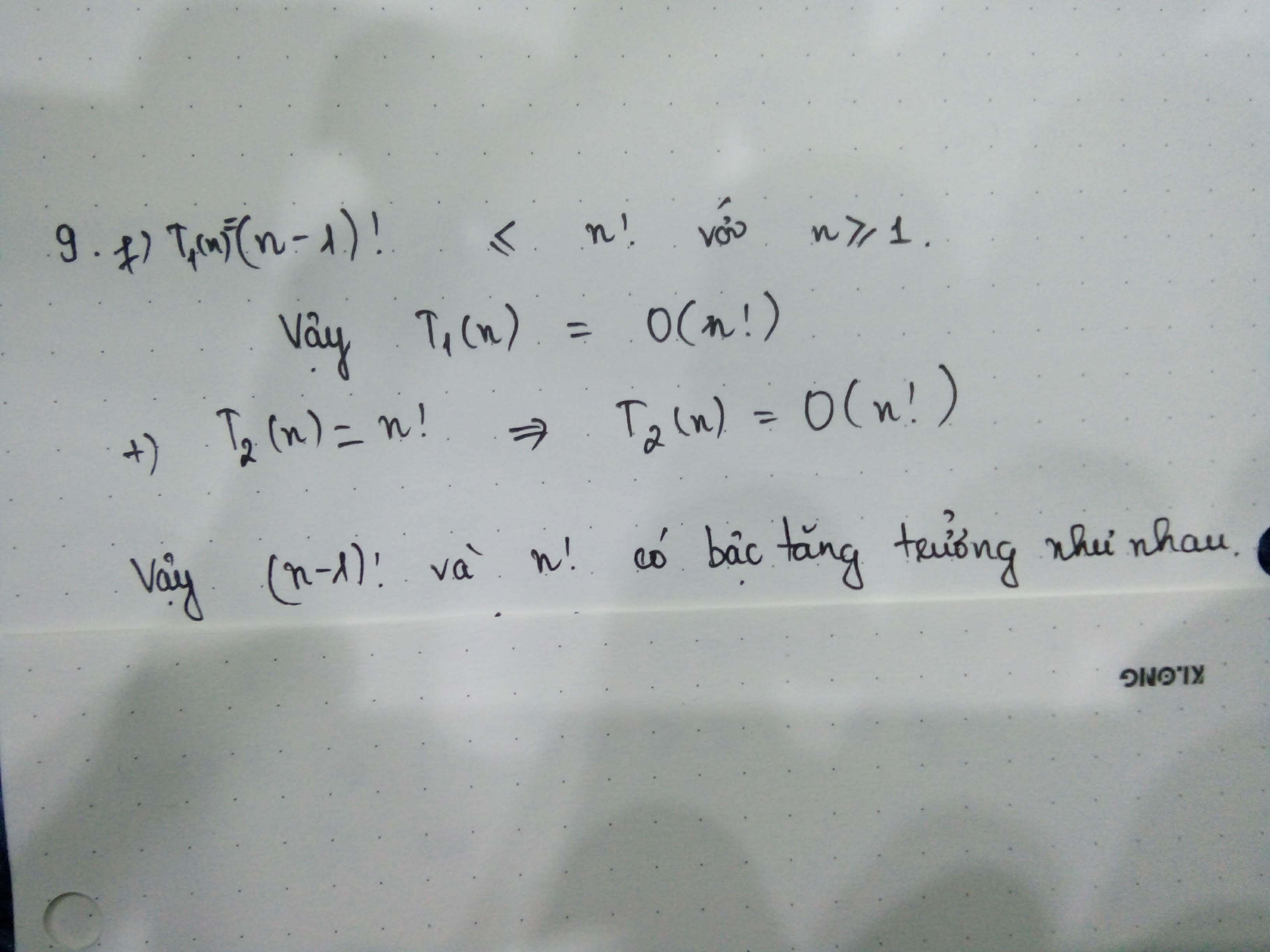
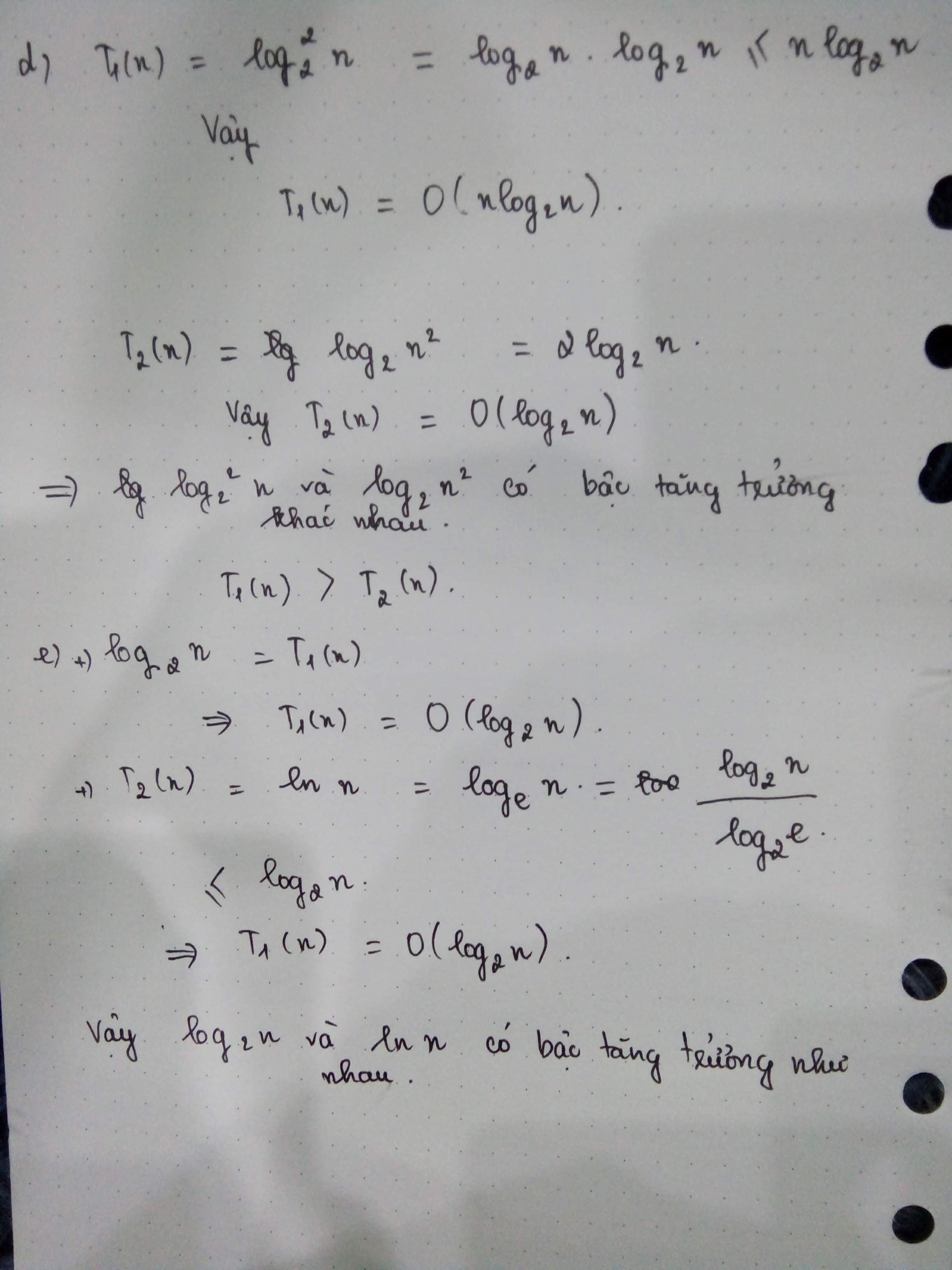
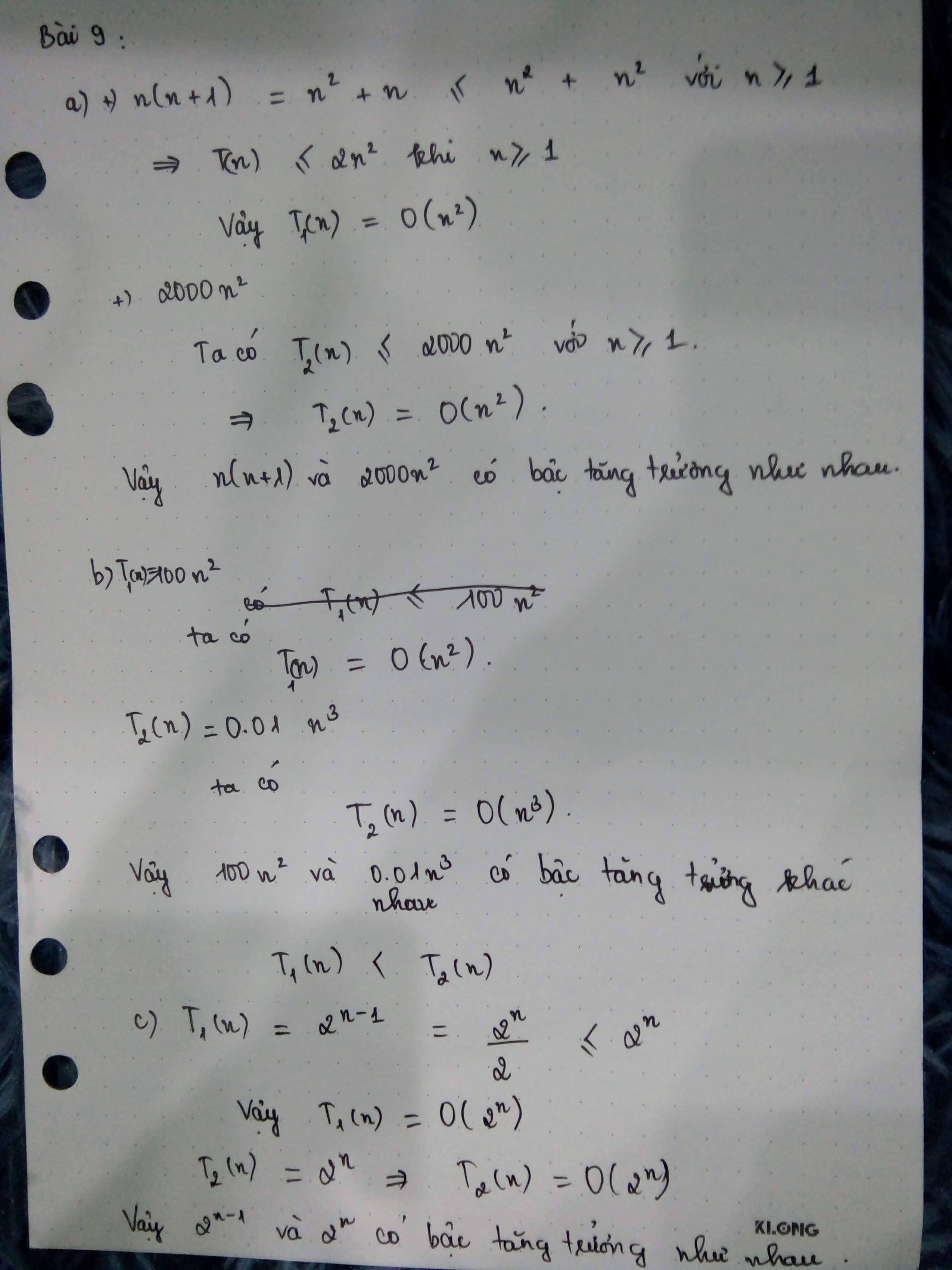
b. (4n)/n= 4

c. (4n)3/n3 = 43 = 64

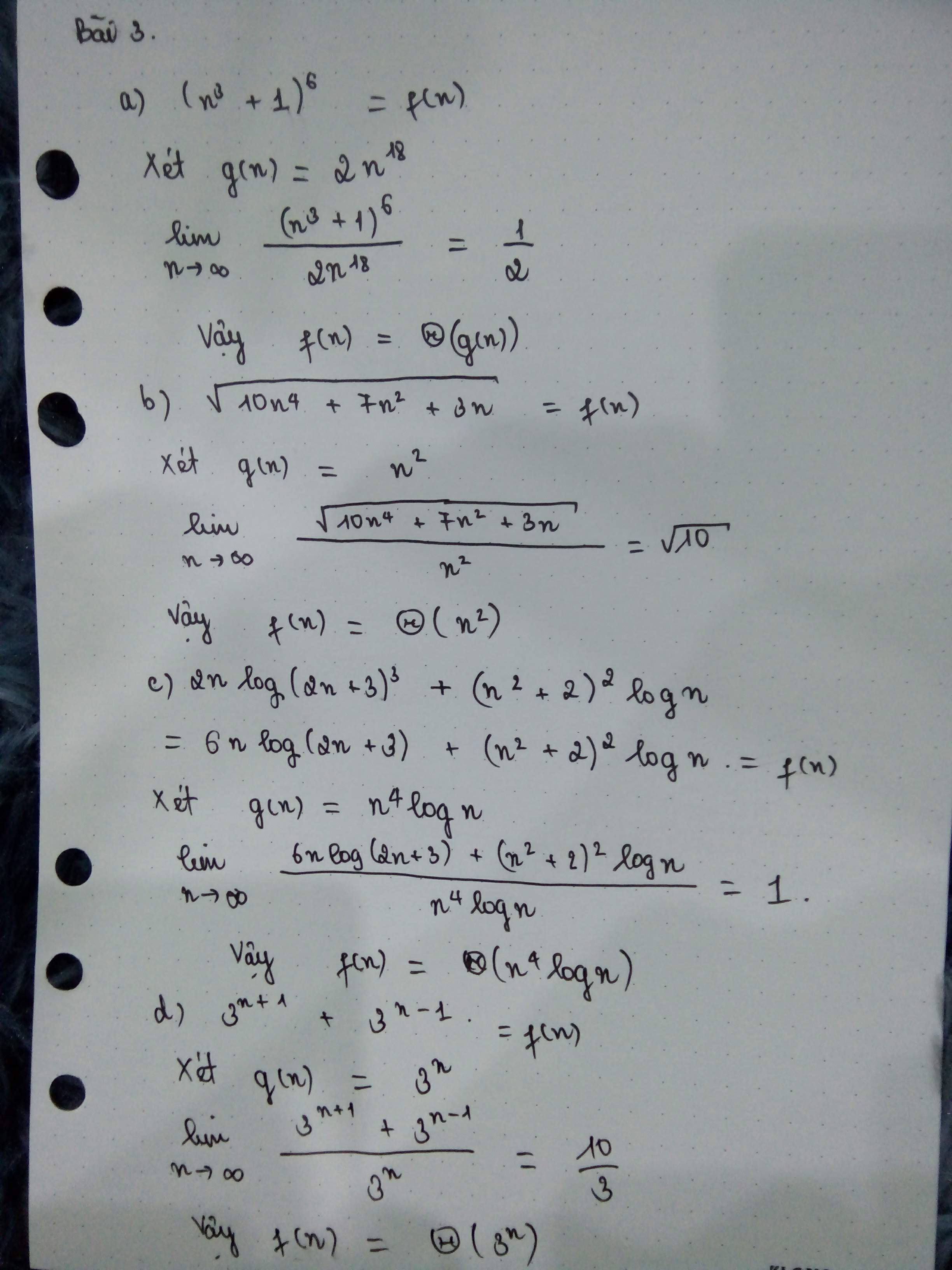
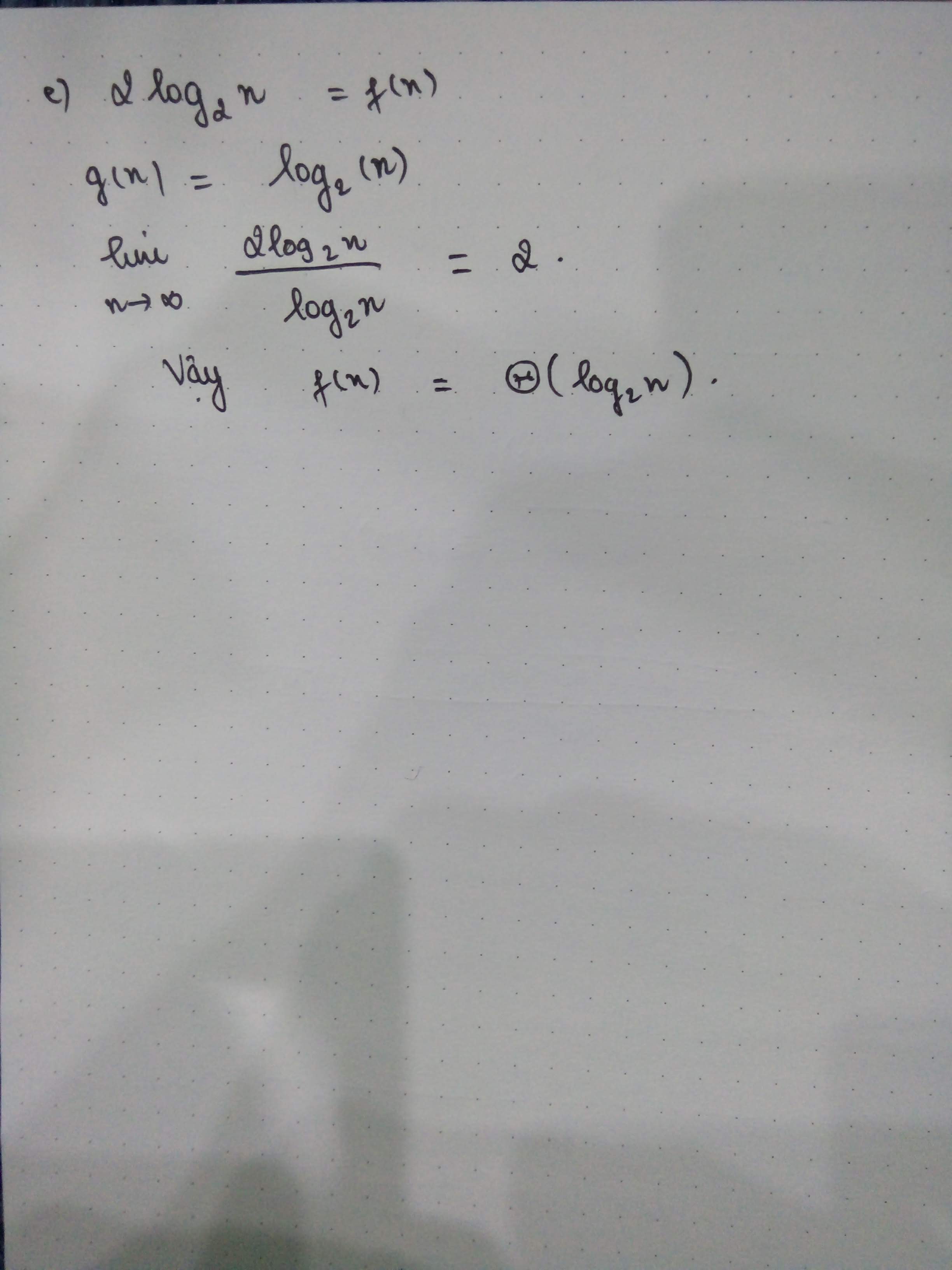
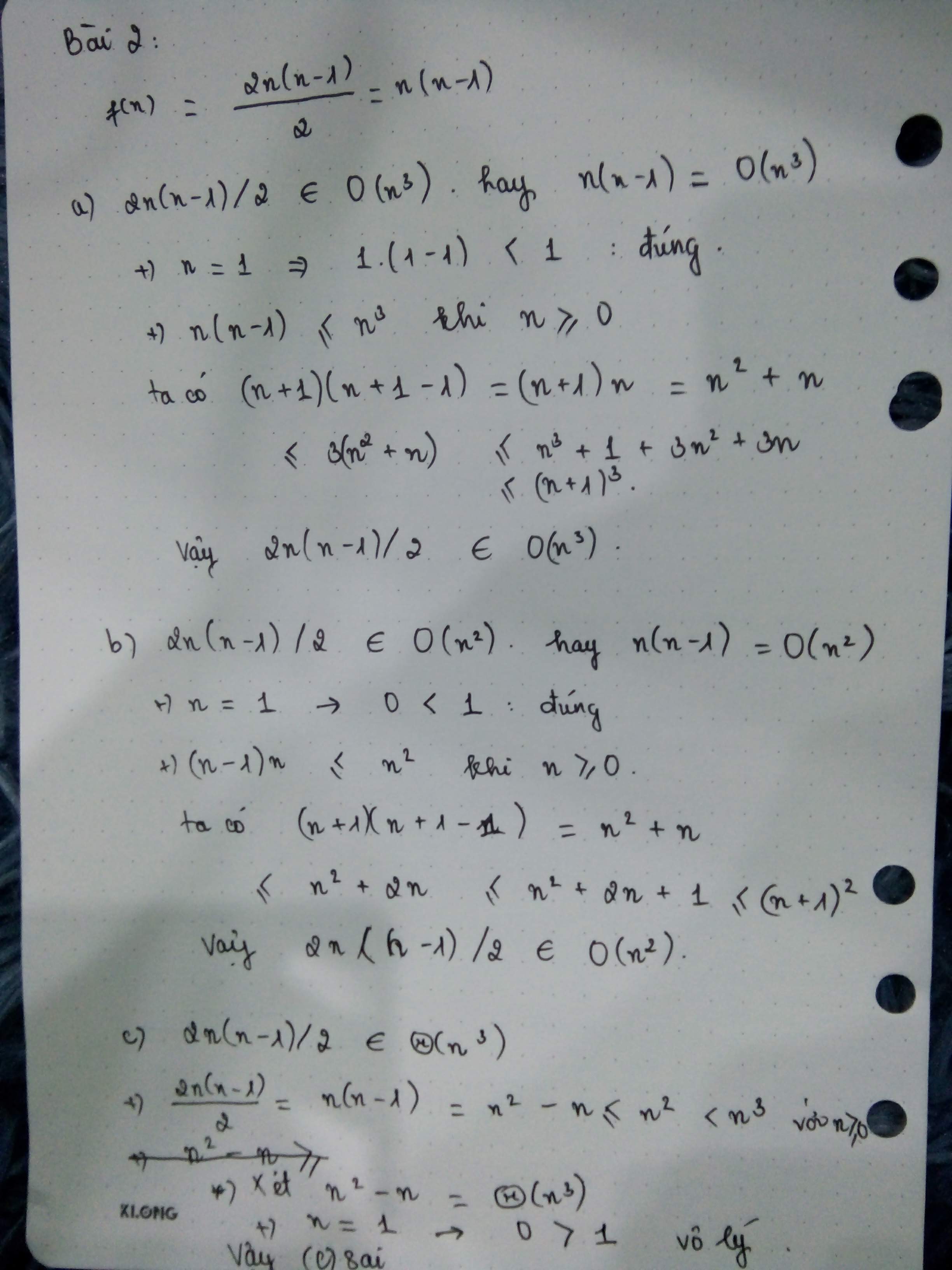
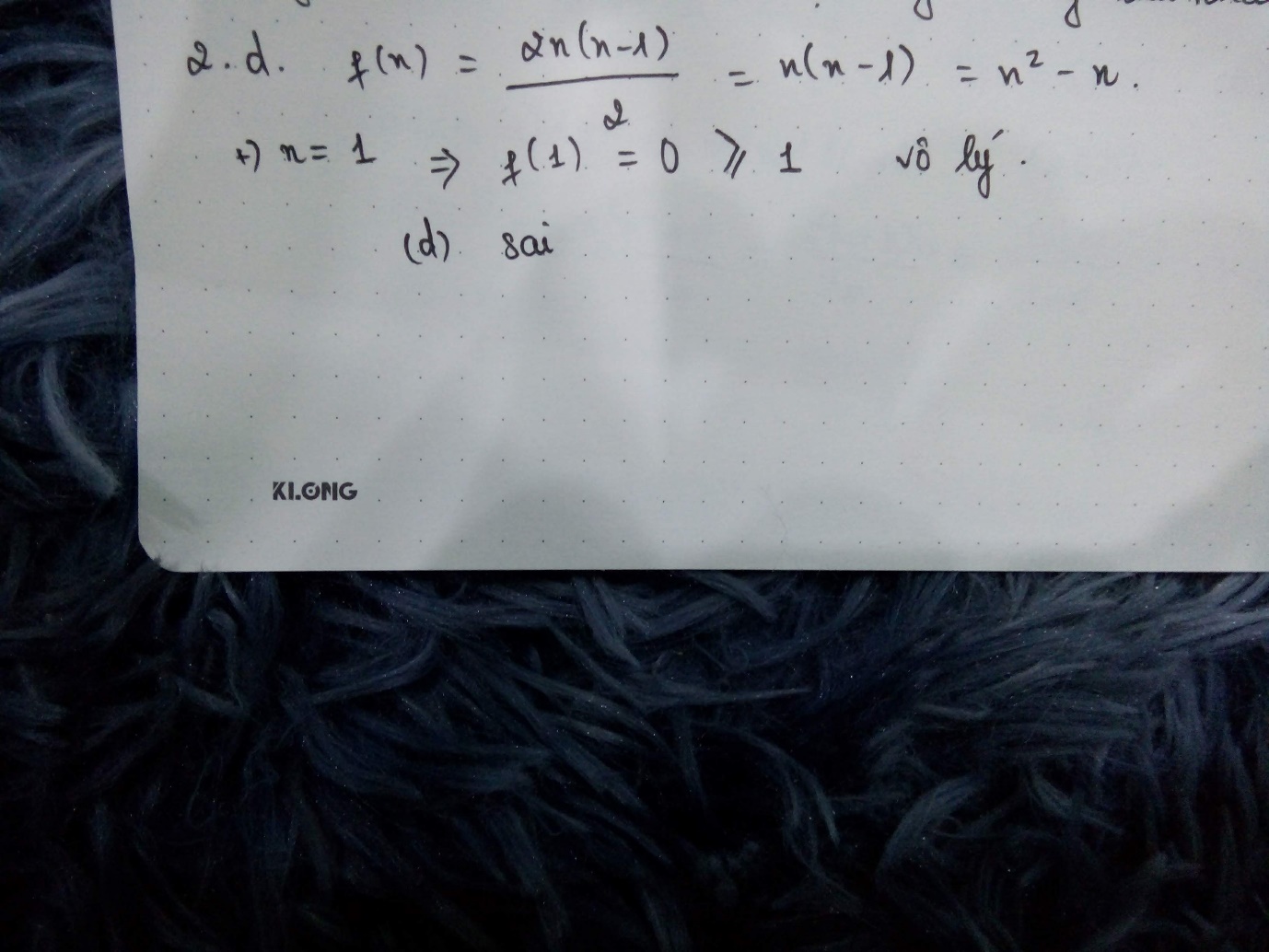
d. (4n)2/n2 = 42 = 16

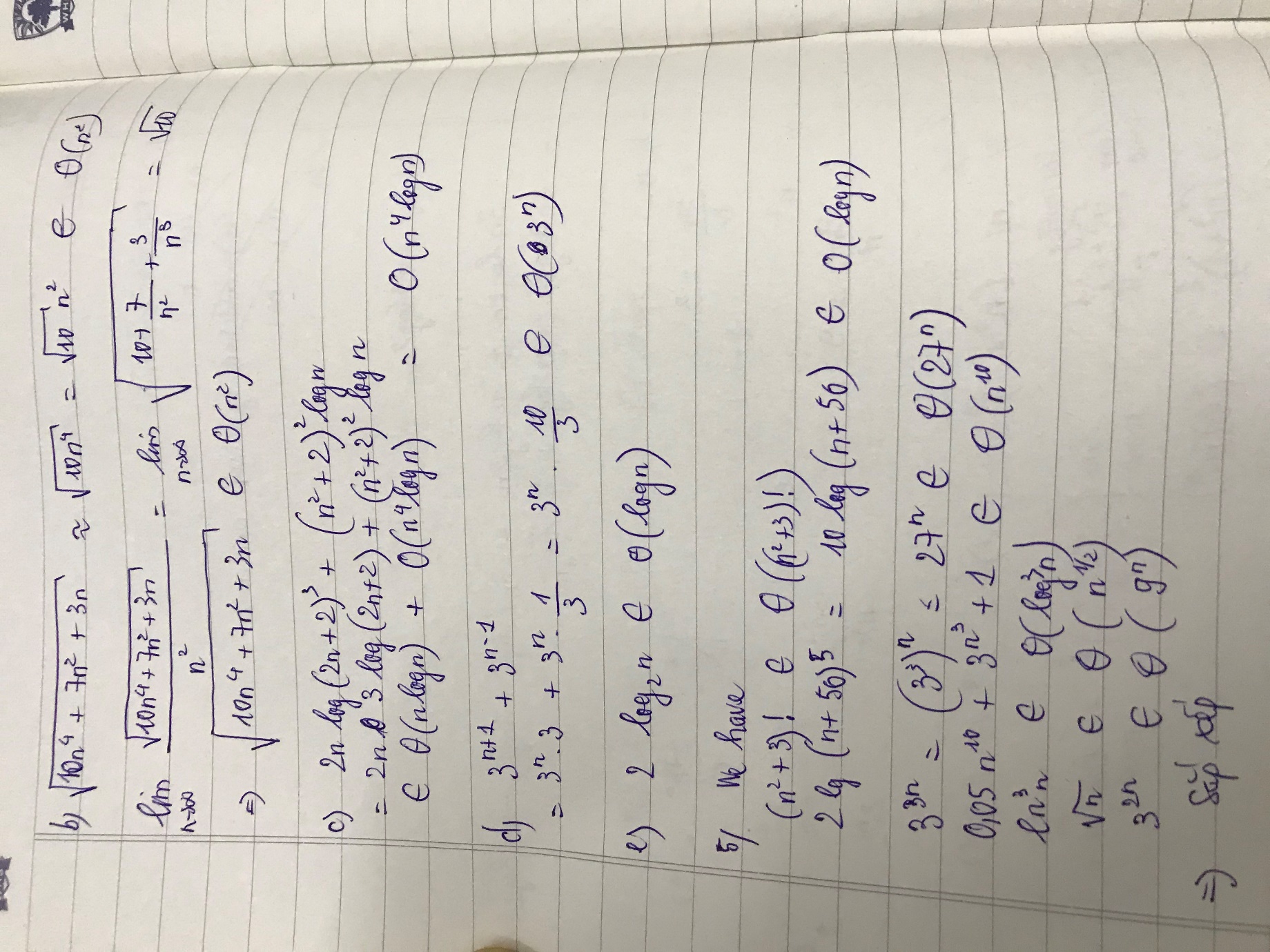
e. (4n)! / n! =

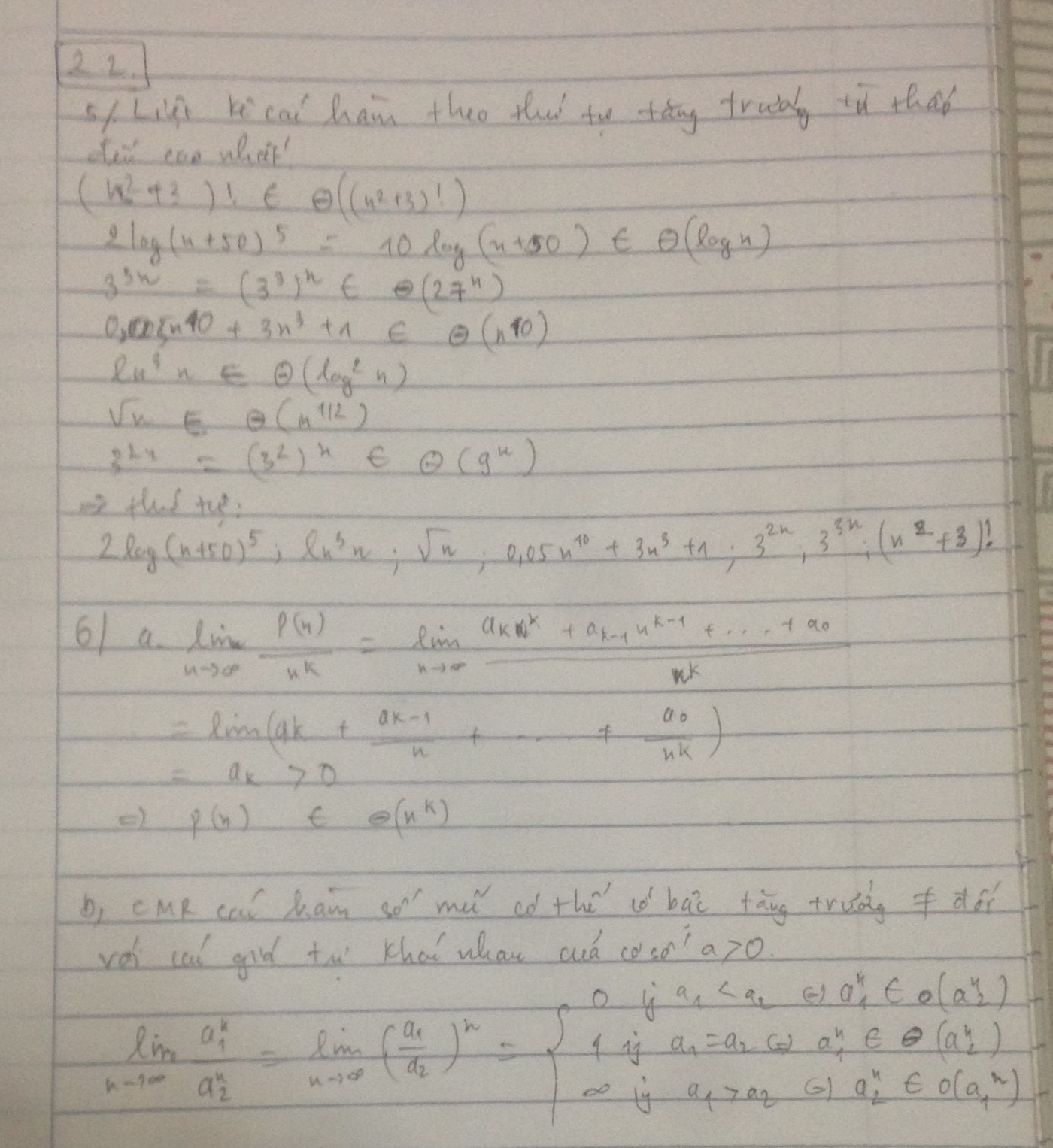
f. 24n/2n = 23n = (2n)3

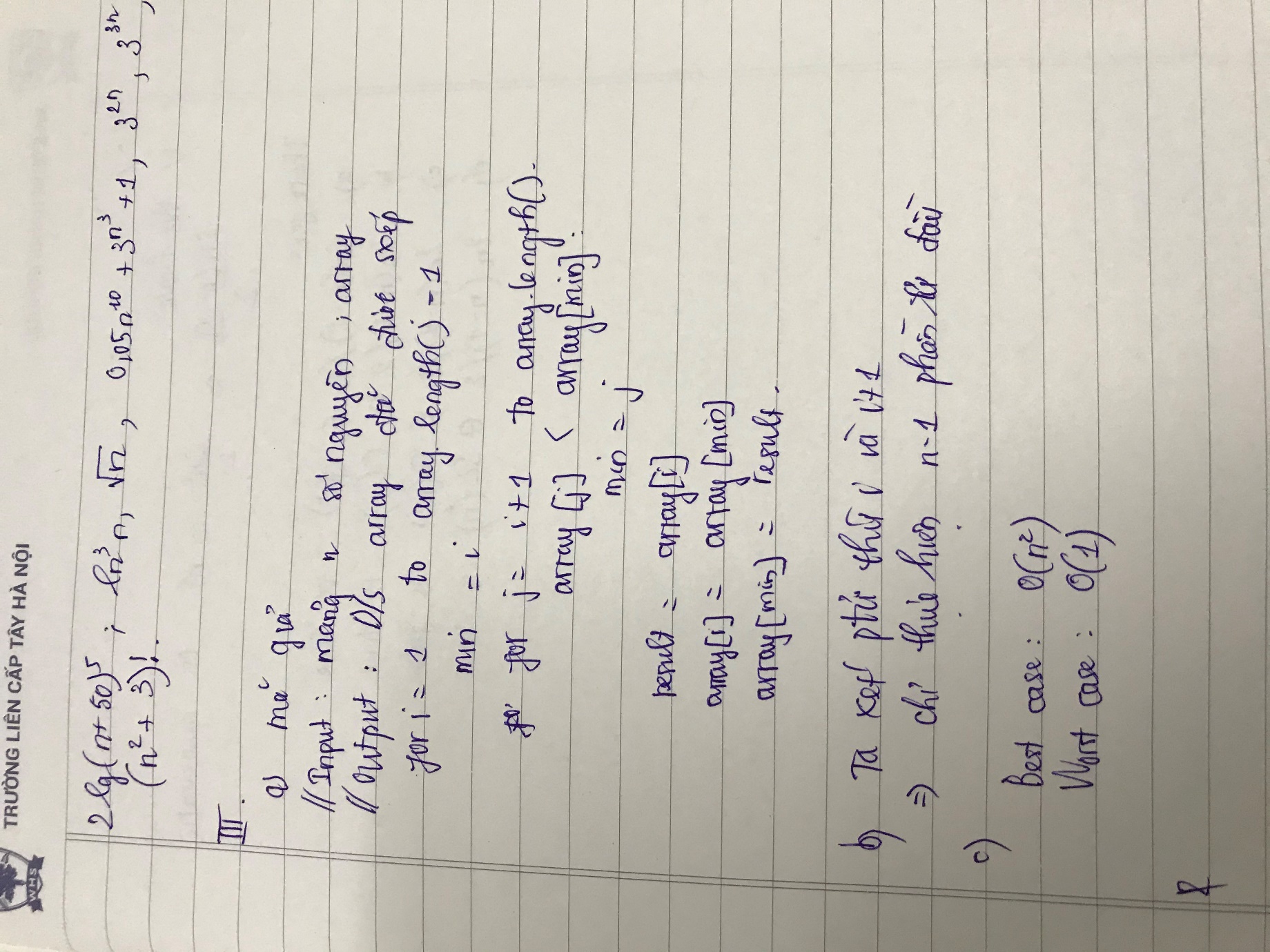


**Exercise 2.2**







**Exercise III**

*import matplotlib.pyplot as plt*

*import time*

*import numpy as np*

*def selectionSort(array):*

*for i in range(0, len(array)-1):*

*min = i*

*for j in range(i+1, len(array)):*

*if(array[j] < array[min]):*

*min = j*

*result = array[i]*

*array[i] = array[min]*

*array[min] = result*

*return array*

*def createArray(n):*

*return np.random.randint(100, size=n)*

*def runTime(n):*

*start = time.time()*

*selectionSort(createArray(n))*

*runtime = time.time() - start*

*return runtime*

*def drawthechart():*

*n\_size = [10, 10 \*\* 2, 10 \*\* 3]*

*time = [runTime(i) for i in n\_size]*

*plt.scatter(n\_size, time, s=100)*

*plt.title("Time")*

*plt.xlabel("Length")*

plt.ylabel("Time")

plt.show()

drawthechart()