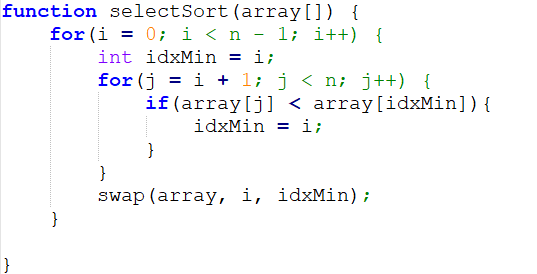
ALGORITHM DESIGN AND ANALYSIS

Assignment 2

Bài 3: Thiết kế thuật toán, chứng minh tính đúng và xác định độ phức tạp của thuật toán

1. Viết mã giả cho thuật toán sắp xếp chọn (selection sort)



1. Vì sao chỉ cần thực hiện với n – 1 phần tử đầu tiên thay vì n phần tử

Do, trong thuật toán selection sort mỗi lần thực hiên ta cần tìm ra được một phần tử min để đặt phần tử đó vào đúng vj trí.

Phần tử thứ nhất: cần được so sánh với n – 1 phần tử còn lại để tìm ra phần tử hợp lệ

Phần tử thứ hai: cần được so sánh với n – 2 phần tử còn lai để tìm ra phần tử hợp lệ

Phần tử thứ 3: cần so sánh n – 3 lần

……..

Phần tử thứ n – 1: cần được so sánh với phần tử thứ n (cuối cùng), rõ ràng trong 2 phần tử này luôn có một phần tử min, và ta sẽ xếp phần tử đó vào vị trí n - 1. Khi đó phần tử còn lại sẽ tự đứng đúng vị trí mà không cần phải so sánh với bất kỳ phần tử nào khác trong danh sách.

1. Đánh giá thời gian thực hiện thuật toán trong trường hợp xấu nhất và tốt nhất

Lần 1: ta cần n – 1 lần so sánh

Lần 2: ta cần n – 2 lần so sánh

…..

Lần n – 1: ta cần 1 lần so sánh

🡺 Số lần so sánh là: (n – 1) + (n – 2) +...+ 1 = n(n -1) / 2

f(n) <=n^2 🡪 độ phức tạp của thuật toán theo kí pháp O là: O(n^2)

\*TH tốt nhất: Xảy ra khi danh sách đã được sắp xếp sẵn, độ phức tạp O(n^2).

🡪 Độ phức tạp O(n^2), và không cần thêm bất kỳ một lần đổi chỗ nào.

\*TH xấu nhất: Xảy ra khi danh sách sắp xếp tăng dần, độ phức tạp O(n^2)

🡪 Độ phức tạp là O(n^2), và mỗi lần so sánh, ta đều cần phải thực hiện thao tác hoán đổi để tìm ra phần tử hợp lệ (min) sau mỗi lần duyệt.

1. Viết chương trình và thực hiện với số lượng lớn các phần tử được sinh ngẫu nhiên và vẽ biểu đồ phụ thuộc thời gian vào kích thước

Chương trình: <file SelectionSort.java> trong thư mục bài tập

Biểu đồ:

Đơn vị đo thời gian: ms (mili giây)

Sinh ngẫu nhiên các phần tử: random.nextInt(n)

Xác định thời gian thực hiện thuật toán: System.nanoTime().

Sau 8 lần đo với kích thước input lần lượt là 50, 100, 200, 400, 600, 800, 1000, 1500 ta được kết quả như sau:

Với n = 50, time = 0.025(ms)

Với n = 100, time = 0.16(ms)

Với n = 200, time = 0.433(ms)

Với n = 400, time = 4.937(ms)

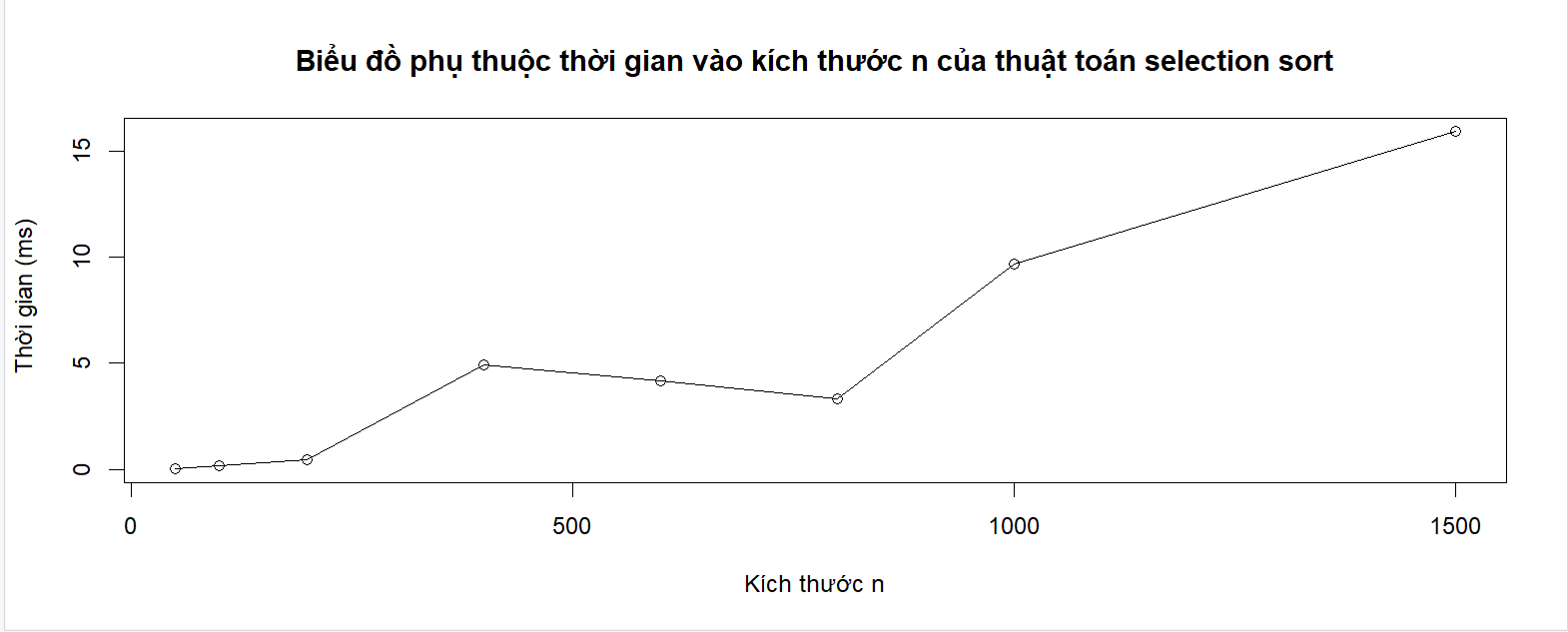
Với n = 600, time = 4.173(ms)

Với n = 800, time = 3.323(ms)

Với n = 1000, time = 9.677(ms)

Với n = 1500, time = 15.924(ms)

Sử dụng phần mềm RStudio để vẽ biểu đồ, ta có:­



1. Chứng minh tính đúng của thuật toán sử dụng bất biến vòng lặp

Claim: Với mỗi mảng số nguyên(thực) kích thước n > 0, thuật toán selection sort là đúng

Khởi tạo: Với n = 1, hiển nhiên danh sách đã được sắp xếp, thuật toán đúng

Duy trì: Cần chứng minh với n > 1, thuật toán selection sort là đúng đắn, tức phần tử a1 <=a2 <=a3…..<=an

Thật vậy:

Ở lần duyệt thứ nhất:

So sánh a1 với a2, tìm ra được min của 2 phần tử, tiếp tục so sánh min a3 ta tìm được min của 3 phần tử……cứ thế sau khi kết thúc ta tìm được min của n phần tử. Sau khi kết thúc vòng lặp thứ nhất, ta sẽ chọn ra được 1 phần tử nhỏ nhất của mảng và đặt phần tử đó ở vị trí đầu tiên của danh sách.

Ở lần duyệt thứ hai, ta tiếp tục so sánh phần tử a2 với n – 2 phần tử ở phía sau a2, nếu phần tử ai < a2, ta sẽ hoán đổi vị trí giữa a2 với a[i]. Sau khi kết thúc vòng lặp ta tìm được phần tử nhỏ nhất của lần duyệt đó, và đây cũng chính là phần tử nhỏ thứ 2 của danh sách. Như vậy sau khi kết thúc vòng lặp này, các phần tử từ vị trí đầu đến vị trí thứ 2 đã được sắp xếp.

…….

Cứ thế tiếp tục, ở lần duyệt thứ I, ta sẽ tìm được phần tử nhỏ thứ I trong mảng. Như vậy, các phần tử từ vị trí đầu tiên đến vị trị i đã được sắp xếp.

Kết thúc: Ở lần duyệt thứ n – 1 (cuối cùng), ta so sánh phần tử a(n-1) với phần tử thứ an. Rõ ràng trong 2 phần tử sẽ tồn tại 1 phần tử nhỏ hơn, đây là phần tử nhỏ nhất của lần duyệt này và cũng là phần tử nhỏ thứ n – 1 của danh sách.Khi n – 1 phần tử trong dãy được sắp xếp thì dãy đã được sắp xếp