Homework 2

Bài 1:

1. Mã giả:

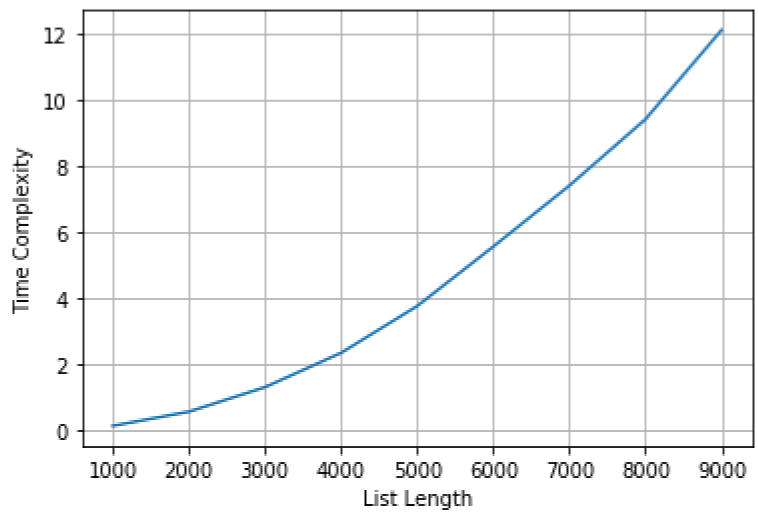
```
program selectionSort

input: Mảng A có n phần tử sắp xếp ngẫu nhiên
output: Mảng A sau khi sắp xếp từ bé đến lớn

start:
minIdex = i
for i := 0 to length(A) - 1 do
        for j := i + 1 to length(A) do
                if A[minIdex] > A[j]
                minIdex = j
        temp = A[i]
        A[i] = A[minIndex]
        A[minIdex] = A[i]
End;
```

2. Ta chỉ cần thực hiện với n - 1 phần tử đầu tiên vì:

   Sau n - 1 lần thực hiện sắp xếp thì các phần tử của mảng từ vị trí đầu tiên tới vị trí thứ n -1 của mảng là n - 1 phần tử nhỏ nhất của mảng, chúng được sắp xêp theo đúng thứ tự từ nhỏ đến lớn. Do đó phần tử a[n] >= a[n - 1] cho nên mảng đã được sắp xêp theo đung thứ tự sau n - 1 lần lặp.

3. Đánh giá thời gian
   - Trường hợp tốt nhất: Với mỗi lần lặp thi phần tử nhỏ nhất cần tìm chinh là phần tử đầu tiên của mảng con và ta không cần đổi chõ phần tử nhỏ nhất với phần tử đầu tiên của mảng con. Hay nói cách khác mảng cần sắp xếp đã được sắp xếp rồi. Nhưng để tim phần tử nhỏ nhất của mảng con ta vẫn phải dùng lặp con. Vậy số lần thực hiện vong lặp con là n*(n - 1)/2. Thuật toán có độ phức tạp $O(n^2)$.
   - Trường hợp xấu nhất: Với mỗi lần lặp thì phần tử nhỏ nhất không phải là phần tử đầu tiên của mảng, nên ta cần đổi chõ 2 phần tử đó. Trường hợp này ta vòng lặp con vẫn chạy hết để tìm phần tử nhỏ nhất. Vì vậy thuật toán có độ phức tạp $O(n^2)$.
4. Biểu đồ.

```
Sắp xếp 1000  phần tử trong  0.15390270000000328
Sắp xếp 2000  phần tử trong  0.5813892000000465
Sắp xếp 3000  phần tử trong  1.3222351000000003
Sắp xếp 4000  phần tử trong  2.35827119999999
Sắp xếp 5000  phần tử trong  3.7712386000000038
Sắp xếp 6000  phần tử trong  5.565982200000008
Sắp xếp 7000  phần tử trong  7.405143099999975
Sắp xếp 8000  phần tử trong  9.405168400000036
Sắp xếp 9000  phần tử trong  12.091447700000003
```



Bai 2:

1. Ý tưởng thuật toán:
    - Tạo biến x := 0;
    - Vòng lặp duyệt qua mọi giá trị của mảng và so sánh từng giá trị a[i] của mảng với v. Nếu a[i] == v thì gán x = i và hàm sẽ trả về giá trị x ngay trong vòng lặp. Nếu a[i] != v thì tiếp tục vòng lặp.
    - Kết thúc vong lặp trả về giá trị x.
2. Mã giả:

```
Program findIndex
Input: mảng A có n phần tử và giá trị x
Output: chỉ số của giá trị x trong mảng hoặc 0 nếu x hông xuât hiên trong mảng

Start:
For i := 0 to length(A) do
        If A[i] == x return i
return 0;
End;
```

3. Chứng minh tính đúng

   Bất biến vòng lặp: findIndex(x) = i nếu A[i] == x
   Khởi tạo: i = 0: findIndex = 0 nếu A[0] == 0
   Duy trì: i = j , findIndex = j nếu A[j] == x
   Kết thúc: i = n sau n lần lặp, findIndex = n nếu A[n] == x, nếu A[n] != x findIndex = 0

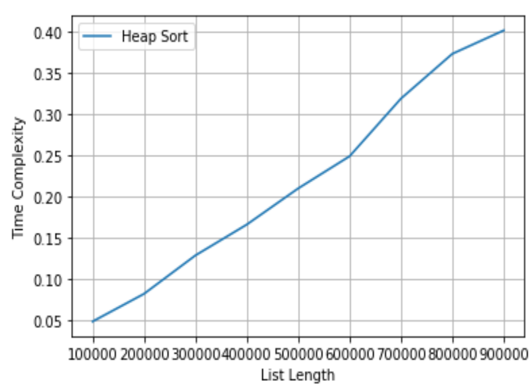
4. Đánh giá thời gian chạy

   Trường hợp tốt nhất: Giá trị cần tìm là giá trị đầu tiên của mảng => độ phức tạp O(1)

   Trường hợp xấu nhất: Giá trị cần tìm ở vị trí cuối cùng trong mảng hoặc không xuất hiện trong mảng => Độ phức tạp O(n).

5. Biểu đồ trong trường hợp không có giá trị của x trong mảng

```
Tìm giá trị 100001 trong mảng  100000 phần tử từ 0 đến 100000 tại vị trí 0 trong thời gian  0.048436100000003535
Tìm giá trị 200001 trong mảng  200000 phần tử từ 0 đến 200000 tại vị trí 0 trong thời gian  0.08191859999999451
Tìm giá trị 300001 trong mảng  300000 phần tử từ 0 đến 300000 tại vị trí 0 trong thời gian  0.12848680000001877
Tìm giá trị 400001 trong mảng  400000 phần tử từ 0 đến 400000 tại vị trí 0 trong thời gian  0.16589510000000018
Tìm giá trị 500001 trong mảng  500000 phần tử từ 0 đến 500000 tại vị trí 0 trong thời gian  0.20970170000001076
Tìm giá trị 600001 trong mảng  600000 phần tử từ 0 đến 600000 tại vị trí 0 trong thời gian  0.24866729999999393
Tìm giá trị 700001 trong mảng  700000 phần tử từ 0 đến 700000 tại vị trí 0 trong thời gian  0.3185925000000225
Tìm giá trị 800001 trong mảng  800000 phần tử từ 0 đến 800000 tại vị trí 0 trong thời gian  0.37287000000000603
Tìm giá trị 900001 trong mảng  900000 phần tử từ 0 đến 900000 tại vị trí 0 trong thời gian  0.4012230999999815
```



Bài 3:

1. 105 trang 29

   Ta có $\lim_{n\to\infty} \frac{n + log(n)}{\sqrt{n}} = +\infty$

   ⇨ $\sqrt{n} = O(n + log(n))$

2. 106 trang 29

   Ta có: $\lim_{n\to\infty} \frac{log(n) + 1}{2(log\ n)^2} = 0$

   ⇨ $\log n + 1 = O(2(\log n)^2)$

3. 107 trang 29

   Ta có: $\lim_{n\to\infty} \frac{4nlog\ n + n}{n^2 - n/_2} = 0$

   ⇨ $4n\log n + n = O((n^2 - n)/2)$

4. 157 trang 32

   Ta có : $\lim_{n\to\infty} \frac{100n + log\ n}{n + (log\ n)^2} = 100$

   Vậy chọn $c_1 = 99$, $c_2 = 101$ thì

   $C_1 (n + (\log n)^2) < 100n + \log n < c_2 (n + (\log n)^2)$

   Do đó $n + log\ n = \Theta(100n + log\ n)$

5. 158 trang 32

   Xét $\log(\log(n^2)) = \log(2\log(n))$

   ⇨ $\lim_{n\to\infty} \frac{log\ log(n^2)}{log\ n} = 0$

   ⇨ $\log\log(n^2) = O(\log n)$

6. 159 trang 32

   Xét $\lim_{n\to\infty} \frac{n\ logn^2}{n^2/_{log\ n}} = \lim_{n\to\infty} \frac{logn^3}{n} = 0$

   ⇨ $n(\log n)^2 = O\ (n^2/_{log\ n})$

Bài 4:

1. Bài 264 trang 51

   Bất biến vòng lặp: $v_i = \sum_{i=0}^{n} A[i].x^{n-i}$

   Khởi tạo: i = 0 , $v_0 = \sum_{i=n}^{n} A[n].x^{n-n}$

   Duy trì: lần lặp thứ j, v_{n-j} = A[j + 1] + A[j] * x = A[j]$.x^0 + (\sum_{i=j+1}^{n} A[i].x^{x-j-1}) * x$
   $= \sum_{i=j}^{n} A[i].x^{n-j}$

   Kết thúc: i = 0 sau n lần lặp, Horner(A, n) = $v_n = \sum_{i}^{n} A[i].x^{n-0}$

2. Bài 265 trang 51

   Bất biến vòng lặp: F(n) là số fibonacci thứ n

   Khởi tạo: F(0) = 0; F(1) = 1; F(2) = F(0) + F(1) = 2;

   Duy trì: F(k + 1) = F(k -1) + F(k)

   Kết thúc: i = n sau n – 2 lần lặp => F(n) = F(n - 1) + F(n)