**Bài Tập Tuần 2**

***Bài 1*: *Sắp xếp chọn*.**

**Mã giải cho bài toán** (lập trình C/C++)

In: dãy các phần tử a1, a2,…., an.

Out: dãy các phần tử đã được sắp xếp theo thứ thự tăng dần.

1. for i=1 to n-1 {
2. key = i;
3. for j = i+1 to n{
4. if (a[j] > x){
5. k = j;
6. x = a[j];
7. }
8. x = a[i];
9. a[k] = a[i];
10. a[i] = x;
11. }
12. }

**Chương trình:**

# include <stdio.h>

# include <conio.h>

# include <stdlib.h>

// nhap cac phan tu cua mang

void nhap\_day(int \*a,int n)

{

int i;

for (i=0;i<n;i++)

{

printf ("a[%d] = ",i);

scanf ("%d",(a+i));

}

}

// in cac phan tu cua mang

void in\_day(int \*a, int n)

{

int i;

for (i=0;i<n;i++)

{

printf (" %d",a[i]);

}

}

// sắp xếp chọn

void select\_sort (int \*a,int n)

{

int x,k,i,j;

for (i=0;i<n-1;i++)

{

k=i;

for (j=i+1;j<n;j++)

if (a[j]<x)

{

x=a[j];

k=j;

}

x=a[i];

a[k]=a[i];

a[i]=x;

}

}

int main()

{

int n,\*a;

printf ("nhap n = ");

scanf ("%d",&n);

a=(int\*)calloc(n,sizeof(int));

printf ("\n nhap cac phan tu cua mang\n");

nhap\_day(a,n);

printf ("\nsap xep theo select sert sort \n");

select\_sort(a,n);

in\_day(a,n);

getch();

return 0;

}

**Bất biến của vòng lặp**: Bất biến của vòng lặp là một phần của dữ liệu đầu vào đã được xử lý khi vòng lặp đang được thực thi và khi kết thúc vòng lặp thì bất biến của vòng lặp cho ta tính chất để chứng minh sự đúng đắn của thuật toán. Ba điều về bất biến của vòng lặp.

1. Khởi tạo: Bất biến của vòng lặp phải đúng trước vòng lặp đầu tiên.
2. Duy trì: Nếu nó đã đúng trước một vòng lặp thì nó vẫn đúng trước vòng lặp tiếp theo.
3. Kết thúc: Khi vòng lặp kết thúc bất biến của vòng lặp cho chúng ta tính chất chứng minh sự đúng đắn của thuật toán.

Cụ thể trong bài ta có: Ngay trước bước lặp j thì a[1], a[2],….,a[j-1] là dãy được sắp xếp theo thứ tự khong giảm và a[1], a[2],….,a[j-1] là j-1 số nhỏ nhất trong các dãy a[1], a[2],….,a[n].

**Thời gian thực hiện thuật toán**

Các bước 1, 2, 8, 9, 10 luôn chạy n-1 lần

Bước 3 và 4 sẽ chạy: (n-1) + (n-2) + …. + 1

Lệnh 5: tùy từng trường hợp mà nó chạy hoặc không.

* Trường hợp tốt nhất dãy ban đầu đã được sắp xếp theo thứ tự tăng dần: lệnh 5 không thực hiện lần nào.
* Trường hợp xấu nhất, dãy ban đầu sắp theo thứ tự giảm dần lệnh 5 được thwucj hiện:

(n-1) + (n-3) + (n-5) +….. =

* Kết quả là trong mọi trường hợp, dù tốt nhất hay xấu nhất, thời gian thwucj hiện thuật toán luôn là: O ()

Thuật toán chỉ thực hiện với n -1 phần tử bởi vì phần tử cuối cùng chỉ còn một vị trí trong dãy nên tự động xếp vào vị trí cuối cùng đó.

***Bài 2*: *Bài toán tìm kiếm*.**

**Mã giải bài toán:**

1. k = 0;
2. i = 1;
3. while (k = 0) and (i<= n){
4. if (v = a[i]) k = i;
5. i = i+1;
6. }

**Thời gian thực hiện thuật toán:**

* Trường hợp tốt nhất: a[1] = v, thuật toán thực hiện

2 + 2 + 1 + 1 = 6 = O(1) bước

* Trường hợp xấu nhất: v không thuộc {a[1], a[2], …. , a[n]}, thuật toán thực hiện

2 + 2(n+1) + n = 4 +3n = O (n) bước

Dòng lệnh 3 gồm 2 lệnh so sánh đơn

**Chứng minh bằng Mã giải:**

* Khởi tạo: Trước khi vào vòng lặp, coi như i = 0, do {a[1], … a[i]} =
* Duy trì:
  + Sau bước lặp thứ I, nếu còn thực hiện tiếp bước lặp thứ i+1, chứng tỏ là: sau bước lặp i

Ta xét 2 trường hợp xảy ra sau bước lặp thứ i+1:

+ Nếu a[i+1] v như vậy k = 0 và rõ rang v không thuộc {a[1], a[2], …. , a[i+1]}

+ Nếu a[i+1]= v thì k = i+1

* Vậy bất biến của vòng lặp có tính duy trì
* Kết thúc:
  + Sau bước lặp thứ I, nếu vòng lặp dừng lại, t có 2 trường hợp:
    - Nếu k = 0 thì v không thuộc {a[1], a[2], …. , a[i]} do vòng lặp dừng lại nên i = n, chứng tỏ v không thuộc {a[1], a[2], …. , a[n]}, như vậy là tỏng dãy n số < a1, a2, … , an> không có số nào bằng v
    - Nếu k = I thì v = a[i], vòng lặp dừng lại và ta đưa ra được chỉ số I để ai  = v.
* Bất biến của vòng lặp thỏa mãn đủ 3 tính chất cần thiết, chứng tỏ thuật toán của ta là đúng đắn.

Như vậy ta đã chứng minh tính đúng đắn của bất biến vòng lặp

**100.**

**; g(n) = 6n**

= = limn→∞ n = ∞.

* g(n) = *O(f(n))*

**101.**

***f*(n) = n + log n ; g(n) = n**

limn→∞  = = limn→∞ + limn→∞ = limn→∞ + limn→∞ = 0

* f(n) = O(g(n))

**103.**

**;**

= =  = 0

* f(n) = O(g(n))

**104.**

**f(n) = n log n;**

limn→∞  = limn→∞ = limn→∞ = limn→∞

* f(n) = O(g(n))

**105.**

**f(n) = n + log n; g(n) =**

limn→∞ = limn→∞ = ∞

* g(n) = O(f(n))

**106.**

**f(n) = 2; g(n) = log n + 1**

limn→∞  = limn→∞ = limn→∞ 4log n = ∞

* g(n) = O (f(n))

**107.**

**f(n) = 4nlog n + n; g(n) =**

limn→∞ = lim→∞  = limn→∞ = limn→∞ = 0

* f(n) = O(g(n))

**127.**

**f(n) = ; g(n) = 6n +7**

limn→∞ = limn→∞ = limn→∞ n = ∞

* g(n) = O(f(n))

**128.**

**f(n) = ; g(n) = log (n+3)**

limn→∞ = limn→∞ = limn→∞ = ∞

* g(n) = O(f(n))

**129.**

**f(n) = ; g(n) =**

limn→∞ = limn→∞ = limn→∞ = 0

* f(n) = O(g(n))

**130.**

**f(n) = ; g(n) = 4nlog**

limn→∞ = limn→∞ = limn→∞ = ∞

* g(n) = O(f(n))

**131.**

**f(n) = ; g(n) = n+3**

limn→∞ = limn→∞ = ∞

* g(n) = O(f(n))

**132.**

**f(n) = ; g(n) =**

limn→∞ = limn→∞ = lim­n→∞ + limn→∞ = limn→∞ + 0

= limn→∞  = limn→∞ = ∞

* g(n) = O(f(n))

**Chứng minh các biểu thức sau đúng:**

**145.**

**c + d = với**

limn→∞ = limn→∞ = limn→∞ = 0

**146.**

**c với**

limn→∞ = limn→∞ = limn→∞ = ∞

Áp dụng định lý dưới đây cho bài 147, 148:

Nếu: g(n) < f(n) < h(n) và limn→∞ g(n) = limn→∞ h(n) = M thì limn→∞ f(n) = M

**147.**

Xét: 0 < < 5

Chứng minh:

+ với n = 2 ta có: = 1 < 5 điều này đúng.

+ giả sử đúng với n = k:

+ ta chứng minh đúng với n = k + 1: đây là biểu thức đúng do k>2

limn→∞5 => limn→∞5

Vậy ta có:

**148.**

**n! = Ω(n!)**

Từ bài 147 ta có điều phải chứng minh.

**149.**

Đặt: L = => ln L = log n ln n – ln n ln(log n).

=> L =

=> limn→∞ L =

Ta có:

limn→∞ (log n ln n – n ln(log n))

= limn→∞

= limn→∞

= -∞

Vì (ln(ln n) → ∞ khi n → ∞ mặt khác limn→∞

=> limn→∞  L = 0 => f(n) = O(g(n)).

**151.**

Theo bài 152 ta có điều phải chứng minh

**152.**

L =

Xét

Đặt log n = t => t→∞ khi n→∞

=> L =

=> limn→∞ limt→∞ (1)

Do:

t! > (2)

mà:

limt→∞ limt→∞ = limt→∞ = 0

=> (3)

từ

Từ (1), (2), (3) => limt→∞ L = 0 (nguyên lí kẹp) =>

**153.** **(n!)! = 0(((n-1)!)!(n-1)!n!** )

*Xét* :

*Ta có*:

Đặt : (n-1)! = a

Nhận xét:

…..

=>(n!)! không là 0(((n-1)!)!(n-1)!n!)

**154. (n!)! = O(((n – 1)!)!(n – 1)!n!)**

Theo bài 153: = ∞

=> (n!)! = O(((n – 1)!)!(n – 1)!n!

**155.**

Xét: O = O()

= = =

= = = ∞

Do vậy:

**259.**

Prove that the following algorithm that adds the values in an array A[1 … n] is correct:

function sum(A)

comment Return

1. s := 0;
2. for i:= 1 to n do
3. s := s + A[i]
4. return (s)

Giải:

Bất biến vòng lặp là: trước bước thứ i thì s = A[1] + A[2] + … + A[i-1].

Ta chứng minh bất biến vòng lặp trên thỏa mãn 3 tính chất: khởi tạo, duy trì, kết thúc.

+) Khởi tạo: trước bước thứ nhất (i = 1): s = 0 đúng.

+) Duy trì:

Giả sử trước bước thứ i, ta có: s = A[1] + A[2] + … + A[i-1].

Trước bước thứ i+1 thì: s = s + A[i] = A[1] + A[2] + … + A[i-1] + A[i].

Như vậy trước bước thứ i +1, bất biến vẫn đúng.

+) Kết thúc: i = n+1 thì: s = A[1] + A[2] + … + A[n] chính là tổng cần tìm.

Bất biến vòng lặp trên chứng tỏ thuật toán tính tổng các số trong một dãy là đúng.

dãy là đúng.