Bài tập 1

Nhập môn phân tích độ phức tạp thuật toán

Nguyễn Minh Đức

1712358

Yêu cầu 1

Em chọn câu 3 câu 5

Câu 3

```
sum :=0; i:=1;
while i<=n do
     j:=1;
     while j<=n do
        if (i=j) or (i+j=n+1) then
           sum := sum+a[i,j];
        endif;
        j := j+1;
     endw;
     i := i+1;
endw;
```

Đếm số phép gán:

Ta phân đoạn thuật toán rồi đếm các phép gán trong mỗi đoạn

```
sum :=0; i:=1;                                        {2}
while i<=n do                          Q
     j:=1;                                            {n}
     while j<=n do
        if (i=j) or (i+j=n+1) then
           sum := sum+a[i,j];            S            {Si}
        endif;
        j := j+1;                      R              {n*n}
     endw;
     i := i+1;                                        {n}
endw;
```

Suy ra tổng số phép gán của thuật toán $G = n^2 + 2n + 2 + Si$

Tính Si

Vì j luôn được gán bằng 1 rồi chạy tới n rồi lại về 1 nên luôn luôn tìm được 1 giá trị sao cho i=j (và chỉ tìm được 1 giá trị duy nhất khi i không đổi), và cũng luôn tìm được 1 giá trị sao cho i+j=n+1 (cũng chỉ tìm được giá trị duy nhất khi i không đổi). Giả sử n là chẵn, khi đó n+1 là lẻ, nên 2 giá trị trên không thể có cùng j. Suy ra khi n chẵn số, thì cứ sau 1 vòng lặp Q, thì lệnh if trong S được thực hiện thành công 2 lần.

=> Si = 2*n => $G = n^2 + 2n + 2 + 2n = n^2 + 4n + 2$

Còn khi n lẻ thì chỉ tồn tại duy nhất 1 giá trị i để i=j và i+j=n+1

Nên suốt chương trình 2 điều kiện này chỉ trùng nhau 1 lần

=> $G = n^2 + 2n + 2 + 2n - 1 = n^2 + 4n + 1$

Tựu lại ta được $G = n^2 + 4n + 1 + (n + 1)mod2$

Đếm số phép so sánh:

Cũng phân đoạn thuật toán, rồi đếm số phép so sánh trong từng đoạn

S

```
sum :=0; i:=1;
while i<=n do                                 Q        {n+1}
     j:=1;
     while j<=n do                                     {n*(n+1)}
         if (i=j) or (i+j=n+1) then                    {S1+S2}
            sum := sum+a[i,j];
         endif;                                   S
         j := j+1;
     endw;                                    R
     i := i+1;
endw;
```

Suy ra tổng số phép so sánh của chương trình $S = n^2 + 2n + 1 + S1 + S2$

Tính S1+S2

Ta thấy rằng nếu điều kiện i=j thỏa thì không cần phải so sánh i+j=n+1.

Số lần i=j thỏa là n lần. Ngoài ra, nếu i=j không thỏa thì ta chắc chắn phải so sánh 2 lần.

=> $S1 + S2 = 2n^2 - n$

=> $S = n^2 + 2n + 1 + 2n^2 - n = 3n^2 + n + 1$

So với chương trình thực nghiệm

	Tính toán lý thuyết		Chương trình thực nghiệm	
n	G	S	G	S
10	142	311	142	311
21	526	1345	526	1345
100	10402	30101	10402	30101
1001	1006006	3007005	1006006	3007005

source code

```cpp
#include <iostream>

using namespace std;

int main()
{
    int g = 0;
    int s = 0;
    int n;
    cin >> n;
    int sum =(++g,0);
    int i = (++g,1);
    while (++s,i <= n)
    {
    int j = (++g,1);
    while (++s,j <= n)
    {
        if ((++s,i == j) || (++s,i + j == n + 1))
            sum  =(++g, sum + 1);

    j =(++g, j + 1);
    }
    i =(++g, i + 1);
    }
    cout << "g: " << g << endl;
    cout << "s: " << s << endl;
    system("pause");

    return 0;
}
```

Vậy tính toán lý thuyết của hoàn toàn trùng khớp với chương trình thực nghiệm.

=> Độ phức tạp thuật toán trên là $\theta(n^2)$

Câu 5

Phân đoạn thuật toán, ta được

```
sum :=0; i:=1; idx=-1;
while i<=n do
     j:=1;                                      1
     while j<=n do
        if (i=j) and (i+j=n+1) then
           idx=i;                               3
        endif;
        sum := sum+a[i,j];
        j := j+1;                               2
     endw;
     i := i+1;
endw;
if idx<>-1 then
  sum := sum - a[idx,idx];                      4
endif;
```

Số phép gán(g) và so sánh(ss):

Ngoài các vòng lặp: g = 3, ss = 0

(1): g = 2n, ss =n + 1

(2): g = $2n^2$, ss = n(n + 1) = n^2 + n

(3):

i chạy từ n về 1 và j từ 1 lên n nên luôn có 1 giá trị của j để j = i.

Tương tự ta cũng có 1 giá trị của j để i + j = n + 1.

Nếu i = j và i + j = n + 1 -> i = (n + 1) / 2 -> n lẻ. Ta có:

Nếu n lẻ:

g = 1, ss = n^2 + 1

Nếu n chẵn:

g = 0, ss = n^2

(4):

Nếu n lẻ, (4) -> idx = i -> idx <> -1:

g = 1, ss = 1

Nếu chẵn, (4) -> idx = -1:

g = 0, ss = 1

(1) + (2) + (3) + (4) + (5):

n lẻ:

$g = 3 + 2n + 2n^2 + 1 + 1 = 2n^2 + 2n + 5$

$ss = 0 + n + 1 + n^2 + n + n^2 + 1 + 1 = 2n^2 + 2n + 3$

n chẵn:

$g = 3 + 2n + 2n^2 = 2n^2 + 2n + 3$

$ss = 0 + n + 1 + n^2 + n + n^2 + 1 = 2n^2 + 2n + 2$

Vậy nếu n lẻ sẽ có $2n^2 + 2n + 5$ phép gán và $2n^2 + 2n + 3$ phép so sánh, nếu n chẵn sẽ có $2n^2 + 2n + 3$ phép gán và $2n^2 + 2n + 2$ phép so sánh.

Soure code thực nghiệm

```cpp
using namespace std;
int main()
{
    int n;
    do
    {
        cout << "Nhap n: ";
        cin >> n;
        int g = 0, ss = 0;
        int s = 0, i = 1, idx = -1; g += 3;
        while (++ss, i <= n)
        {
            int j = 1; ++g;
            while (++ss, j <= n)
            {
                if ((++ss, i == j) && (i + j == n + 1))
                {
                    ++ss;
                    idx = i; ++g;
                }
                s += 1; ++g;
                ++j; ++g;
            }
            ++i; ++g;
        }
        if (++ss, idx != -1)
        {
            s -= 1; ++g;
        }
        cout << "So phep gan: " << g << '\n'
            << "So phep so sanh: " << ss << '\n';
    } while (n);
    return 0;
```

Kết quả cho thấy tính toán lý thuyết hoàn toàn trùng khớp với thực nghiệm

Từ công thức trên suy ra độ phức tạp thuật toán là O(n^2)

Yêu cầu 2:

Em chọn câu 9 câu 10

Câu 9

```
float Delta(float x, long n) {
        long i=1; float z=0;
        while(i<=n){
             long j=1, k=1; float t=1;
             while(j<=i){
                    t = t*x;
                    k = k+2;
                    j = j+k;
             }
             z = z + i*t; i = i + 1;
        }
        return z;
}
```

Phân đoạn chương trình, ta được:

```
float Delta(float x, long n) {
        long i=1; float z=0;                 {2}
        while(i<=n){                         {n+1}
  Q          long j=1, k=1; float t=1;       {3n}
             while(j<=i){                    {Si+n}
  S                 t = t*x;                 {Si}
                    k = k+2;                 {Si}
                    j = j+k;                 {Si}
             }
             z = z + i*t; i = i + 1;         {2n}
        }
        return z;
}
```

Số phép gán trong chương trình $G = 2 + 5n + 3Si$

Số phép so sánh trong chương trình $S = 2n + 1 + Si$

Tính Si:

Ta thấy rằng k luôn là 1 dãy số lẻ liên tục 1,3,5,7,.. (do k=k+2 và k ban đầu bằng 1).

=> j luôn được tăng thêm 1 số lẻ liên tục lần trước khi lớn hơn i.

=> Vào vòng lặp cuối cùng thì $j = 1 + 3 + 5 + \cdots \leq i$

Giả sử giá trị k trong vòng lặp cuối bằng k_0 và Sj là số lần lặp khi i không đổi thì

=> $1 + 3 + 5 + .. + k_0 = \sum_{r=1}^{Sj} 2Sj - 1 = S_j{}^2$

Số lần lặp sẽ là số tự nhiên lớn nhất sao cho bình phương của nó nhỏ hơn hoặc bằng i.

$$=> Sj = \lfloor\sqrt{i}\rfloor$$

Tổng lại ta có $Si = \sum_{i=1}^{n}\lfloor\sqrt{i}\rfloor \leq n\sqrt{n} \in O(n\sqrt{n})$

=> $S = 2n + 1 + O(n\sqrt{n})$

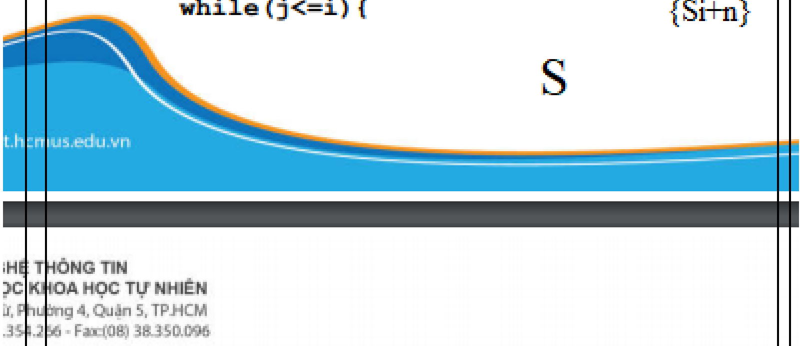
=> $G = 5n + 2 + O(n\sqrt{n})$

Vậy thuật toán có độ phức tạp O($n\sqrt{n}$)

Câu 10:

Phân đoạn chương trình, ta được:

```
float Alpha(float x, long n) {
        long i=1; float z=0;                      {2}
        while(i<=n){                  R             {n+1}
            long j=1; float t=1;                     {2Si}
            while(j<=i){                              {Si+n}
                                      S
                t = t*x;                          {Si}
                j = 2*j;                           {Si}
            }
            z = z + i*t;   i = i + 1;            {2n}
        }
        return z;
}
```



Số phép gán trong chương trình $G = 2 + 2n + 2Si$

Số phép so sánh trong chương trình $S = 2n + 1 + Si$

Tính Si:

Để ý rằng j luôn bắt đầu các lũy thừa của 2 liên tục cho tới khi lớn hơn i.

Gọi số lần lặp của S khi i chưa thay đổi là Sj => j sẽ được nhân 2 Sj lần

=> $2^{Sj} = max_j \leq i$

Từ đó suy ra số lần lặp Sj là số tự nhiên lớn nhất sao cho 2 mũ số đó nhỏ hơn hoặc bằng j và bằng $\lfloor\log_2 i\rfloor$.

Tổng lại ta có $Si = \sum_{i=1}^{n}\lfloor\log_2 i\rfloor \leq n\log_2 n \in O(n\log_2 n)$

=> $G = 2 + 2n + O(n\log_2 n) \in O(n\log_2 n)$

=> $S = 2n + 1 + O(n \log_2 n) \in O(n \log_2 n)$

Vậy thuật toán có độ phức tạp $O(n \log_2 n)$