**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

---------- 🙢★🙠 ----------

****

**BÀI TẬP MÔN PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN**

**HOMEWORK #01.b: KỸ THUÂT SƠ CẤP PHẦN 1**

***Giảng viên hướng dẫn:*** ThS. Huỳnh Thị Thanh Thương

***Nhóm sinh viên:***

1. Phan Thanh Hải 18520705

**TP. HỒ CHÍ MINH, 17/09/2019**

**BÀI 1**

Tính:

**a.** 1 + 3 + 5 + 7 + … + 999 **f.**

**b.** 2 + 4 + 8 + 16 + … + 1024 **g.**

**c.** **h.**

**d.** **i.**

**e.** **j.**

**Bài làm**

**a.** 1 + 3 + 5 + 7 + … + 999

= (1 + 2.0) + (1 + 2.1) + (1 + 2.2) + (1 + 2.3) + … + (1 + 2.499)

*(có 500 số hạng)*

Ta có:

**b.** 2 + 4 + 8 + 16 + … + 1024

= 2.20 + 2.21+ 2.22 + 2.23 + … + 2.29

Ta có:

**c.**

= 1 + 1 + 1 + … + 1

*(có n – 1 số hạng)*

= (*n* – 1).1 = *n* – 1.

**d.**

**e.**

**f.**

**g.**

**h.**

**i.**

**j.**

**BÀI 2**

For each of the following algorithms, indicate (i) a natural size metric for its inputs, (ii) its basic operation, and (iii) whether the basic operation count can be different for inputs of the same size:

1. computing the sum of *n* numbers
2. computing *n*!
3. finding the largest element in a list of *n* numbers

**Bài làm**

1. computing the sum of *n* numbers
2. a natural size metric for its inputs: *n*
3. its basic operation: addition of 2 numbers
4. whether the basic operation count can be different for inputs of the same size: no
5. computing the sum of *n* numbers
   1. a natural size metric for its inputs: *n*
   2. its basic operation: multiplication of 2 numbers
   3. whether the basic operation count can be different for inputs of the same size: no
6. finding the largest element in a list of n numbers
   1. a natural size metric for its inputs: *n*
   2. its basic operation: comparison of 2 numbers
   3. whether the basic operation count can be different for inputs of the same size: no

**BÀI 3**

Consider the following algorithm.

**ALGORITHM** *Mystery(n)*

//Input: A nonnegative integer *n*

*S* ← 0

**for** *i* ← 1 **to** *n* **do**

*S* ← *S* + *i* ∗ *i*

**return** *S*

1. What does this algorithm compute?
2. What is its basic operation?
3. How many times is the basic operation executed?
4. What is the efficiency class of this algorithm?
5. Suggest an improvement, or a better algorithm altogether, and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

**Bài làm**

1. This algorithm computes the sum of squares of first *n* natural numbers.
2. Its basic operations are the assignment and comparison operation.
3. The basic operation executed *n* times.
4. The efficiency class of this algorithm is *O*(*n*).
5. Instead of using loops, we can use fomular to find the sum of squares of first *n* natural numbers:

Its efficiency class is *O*(1).

**BÀI 4**

Đếm số phép gán và phép so sánh:

sum = 0;

i = 1;

while (i ≤ n) do

j = 1;

while (j ≤ n) do

sum = sum + i\*j;

j = j + 1;

end do;

i = i + 1;

end do;

**Bài làm**

Gọi *αi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

sum = 0; {1 gán}

i = 1; {1 gán}

while (i ≤ n) do {*n* + 1 so sánh}

j = 1; {*n* gán}

while (j ≤ n) do {*αi* + 1 so sánh}

sum = sum + i\*j; {*αi* gán}

j = j + 1; {*αi* gán}

end do;

i = i + 1; {*n* gán}

end do;

Số phép gán:

Số phép so sánh:

Tính ???

= số lần lặp của vòng while trong

= số *j* chạy từ 1 đến *n*, bước tăng là 1

= (*n* – 1) + 1 = *n*.

Kết luận:

Số phép gán:

Số phép so sánh:

**BÀI 5**

Đếm số phép gán và phép so sánh:

i = 1; res = 0;

while (i ≤ n) do

j = 1;

while (j ≤ i) do

res = res + i\*j;

j = j + số thứ tự của nhóm;

end do;

i = i + 1;

end do;

(Lưu ý: hiện tại cô chưa có danh sách nhóm nên "số thứ tự của nhóm" có thể chọn là 1 số bắt kỳ trừ số 1)

**Bài làm**

số thứ tự của nhóm = 2;

Gọi *αi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

i = 1; res = 0; {2 gán}

while (i ≤ n) do {*n* + 1 so sánh}

j = 1; {*n* gán}

while (j ≤ i) do {*αi* + 1 so sánh}

res = res + i\*j; {*αi* gán}

j = j + 2; {*αi* gán}

end do;

i = i + 1; {*n* gán}

end do;

Số phép gán:

Số phép so sánh:

Tính ???

= số lần lặp của vòng while trong

= số *j* chạy từ 1 đến *i*, bước tăng là 2

Kết luận:

Số phép gán:

Số phép so sánh:

**BÀI 6**

Đếm số phép gán và phép so sánh:

float Alpha(float x, long n)

{ long i = 1; float z = 0;

while (i ≤ n)

{ long j = 1; float t = 1;

while (j ≤ i) do

{ t = t\*x;

j = 2\*j;

}

z = z + i\*t;

i = i + 1;

}

return z;

}

**Bài làm**

Gọi *αi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

float Alpha(float x, long n)

{ long i = 1; float z = 0; {2 gán}

while (i ≤ n) {*n* + 1 so sánh}

{ long j = 1; float t = 1; {2*n* gán}

while (j ≤ i) {α*i* + 1 so sánh}

{ t = t\*x; {*αi* gán}

j = 2\*j;; {*αi* gán}

}

z = z + i\*t; {*n* gán}

i = i + 1; {*n* gán}

}

return z;

}

Số phép gán:

Số phép so sánh:

Tính ???

= số lần lặp của vòng while trong

= số *j* chạy từ 1 đến *i*, bước tăng là 2*j*

Ta có:

Do đó

Kết luận:

Số phép gán:

Số phép so sánh:

**BÀI 7**

Đếm số phép gán và phép so sánh:

i = 1;

res = 0;

while i ≤ n do

j = 1;

k = 1;

while j ≤ i do

res = res + i\*j;

k = k + 2;

j = j + k;

endw;

i = i + 1;

endw;

**Bài làm**

Gọi *αi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

i = 1; {1 gán}

res = 0; {1 gán}

while i ≤ n do {*n* + 1 so sánh}

j = 1; {*n* gán}

k = 1; {*n* gán}

while j ≤ i do {*αi* + 1 so sánh}

res = res + i\*j; {*αi* gán}

k = k + 2; {*αi* gán}

j = j + k; {*αi* gán}

endw;

i = i + 1; {*n* gán}

endw;

Số phép gán:

Số phép so sánh:

Tính ???

= số lần lặp của vòng while trong

= số *j* chạy từ 1 đến *i*, bước tăng là *k* (mỗi lần lặp tiếp theo thì *k* tăng lên 2 đơn vị)

(*j* có dạng là một số chính phương)

Ta có:

Do đó

Kết luận:

Số phép gán:

Số phép so sánh:

**BÀI 8**

Đếm số phép gán và phép so sánh:

i = 1; count = 0;

while (i ≤ 3\*n)

{

x = i – 2\*n;

y = n – i;

j = 1;

while (j ≤ x)

{

count = count – 1;

j = j + 2;

}

if (y > 0)

if (x > 0)

count = count + 1;

i = i + 1;

}

**Bài làm**

Gọi *αi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

Gọi *βi*, χ*i* lần lượt là số lần thực hiện câu lệnh so sánh và câu lệnh gán trong đoạn if lồng nhau (tính độc lập với vòng while ngoài).

i = 1; count = 0; {2 gán}

while (i ≤ 3\*n) {3*n* + 1 so sánh}

{

x = i – 2\*n; {3*n* gán}

y = n – i; {3*n* gán}

j = 1; {3*n* gán}

while (j ≤ x) {*αi* + 1 so sánh}

{

count = count – 1; {*αi* gán}

j = j + 2; {*αi* gán}

}

if (y > 0)

{*βi* so sánh}

if (x > 0)

count = count + 1; {χ*i* gán}

i = i + 1; {3*n* gán}

}

Số phép gán:

Số phép so sánh:

Tính ???

Vòng lặp chỉ được thực hiện khi *i –* 2*n* ≥ 1 ⇔ *i* ≥ 2*n* + 1

= số lần lặp của vòng while ngoài

= số *j* chạy từ 1 đến *i – 2n*, bước tăng là 2

Do đó:

Tính ???

Ta có:

|  |  |
| --- | --- |
| *i* | 1 *n 2n* |
| *x = i –* 2\**n* | *– –* 0 *+* |
| *y = n – i* | *+* 0 *– –* |

Xét các trường hợp sau:

**a.** Với *i* < *n* thì ta có:

*βi* = 2

χ*i* = 0 (do câu lệnh if thứ 2 điều kiện không thỏa mãn)

**b.** Với *i* ≥ *n* thì ta có:

*βi* = 1

χ*i* = 0

Kết luận:

Số phép gán:

Số phép so sánh:

**BÀI 9**

Đếm số phép gán và phép so sánh:

i = 1; res = 0;

while i ≤ n do

j = 1;

while (j ≤ i) do

res = res + i\*j;

j = j + 1;

end do;

i = i + số thứ tự của nhóm;

end do;

**Bài làm**

số thứ tự của nhóm = 2;

Gọi *α* là số lần lặp của vòng while ngoài và *βi* là số lần lặp của vòng while trong (tính độc lập với vòng while ngoài).

i = 1; res = 0; {2 gán}

while i ≤ n do {*α* + 1 so sánh}

j = 1; {*α* gán}

while (j ≤ i) do {*βi* + 1 so sánh}

res = res + i\*j; {*βi* gán}

j = j + 1; {*βi* gán}

end do;

i = i + 2; {*α* gán}

end do;

Số phép gán:

Số phép so sánh:

Tính ???

*α* = số lần lặp của vòng while ngoài

= số *i* chạy từ 1 đến *n*, bước tăng là 2

Tính ???

= số lần lặp của vòng while trong

= số *j* chạy từ 1 đến *i*, bước tăng là 1

= (*i* – 1) + 1 = *i*.

Kết luận:

Số phép gán:

Số phép so sánh: