COMP 352: Data Structure and Algorithms

Name: Wang Haochen

Assignment 1

1.(a)

sum ← 0 1

fori ← 1 to n do n

for j ← 1 to n do n

k ← 1

while k <*n* do log(n)

k ← k \* C

sum ← sum + 1

so Big-(O) is O(n^2log(n));

(b)

sum ← 0 1

fori ← 1 to n do n

for j ← 1 to n do n

k ← 1

while k <*n* do n/c

k ← k + C

sum ← sum + 1

so Big-(O) is O(n^3);

2.(a)

Algorithm MyAlgorithm (A)

Input: Array A storing *n* ≥ 1 integers.

Output: Possibly modified Array A

forI ← 0 to A.length - 1 do n

forj ← A.lengthdowntoi + 1 do n+(n-1)+(n-2)+……..+2+1=n(n+1)/2

ifA[i] > A[j - 1]

s ← A[j - 1]

A[j - 1] ← A[i]

A[i] ← s

returnA 1

f(n)= n+4\*n(n+1)/2+1=2n^2+3n+1<=n^2\*c, for every n>=n0;

2n^2<=2n^2 for n>=0;

3n<=3n^2 for n>=0;

1<=n^2 for n>=1;

2n^2+3n+1<=6\*n^2,for every n>=1, C=6, n0=1; g(n) = n^2, so Big(O) is O(n^2);

2n^2+3n+1>=2n^2>=n^2 for n>=0

2n^2+3n+1>=n^2, for every n>=0, C=1, no=0; So Big-Omega(Ω(n)) is Ω(n^2)

(b) IN THE NEXT PAGE

(c).MtAlgorithm simply sorts a given array. It start from the end of the array goes to the beginning, then come back to the beginning.

(b).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Description | i | j | A[i] | A[j-1] | s | A[ ] |
| 1st for loop | 0 | 0 |  |  |  | (48512) |
| 2nd for loop | 0 | 5 | 4 | 2 |  | (48512) |
| If condition is true | 0 | 5 | 2 | 4 | 2 | (28514) |
| 2nd for loop | 0 | 4 | 2 | 1 |  | (28514) |
| If condition is true | 0 | 4 | 1 | 2 | 1 | (18524) |
| 2nd for loop | 0 | 3 | 1 | 5 |  | (18524) |
| If condition is false | 0 | 3 |  |  |  | (18524) |
| 2nd for loop | 0 | 2 | 1 | 8 |  | (18524) |
| If condition is false | 0 | 2 |  |  |  | (18524) |
| 1st for loop | 1 |  |  |  |  | (18524) |
| 2nd for loop | 1 | 5 | 8 | 4 |  | (18524) |
| If condition is true | 1 | 5 | 4 | 8 | 4 | (14528) |
| 2nd for loop | 1 | 4 | 4 | 2 |  | (14528) |
| If condition is true | 1 | 4 | 2 | 4 | 1 | (12548) |
| 2nd for loop | 1 | 3 | 2 | 5 |  | (12548) |
| If condition is false | 1 | 3 |  |  |  | (12548) |
| 1st for loop | 2 |  |  |  |  | (12548) |
| 2nd for loop | 2 | 5 | 5 | 8 |  | (12548) |
| If condition is false | 2 | 5 |  |  |  | (12548) |
| 2nd for loop | 2 | 4 | 5 | 4 |  | (12548) |
| If condition is true | 2 | 4 | 4 | 5 | 4 | (12458) |
| 1st for loop | 3 |  |  |  |  | (12458) |
| 2nd for loop | 3 | 5 | 5 | 8 |  | (12458) |
| If condition is false | 3 | 5 |  |  |  |  |
| Return A |  |  |  |  |  | (12458) |

So the result is A=(12458).

3.(a)nlogn is O(n^5)

nlogn<=Cn^5for n>=n0;

nlogn<=n^5, for every n>=1, C=1 no=1;

nlogn is O(n^5);

(b).30n^3+60000n^4+4000n^2-10 is Θ(n^4)

30n^3+60000n^4+4000n^2-10<=(30+6000+4000-10)n^4, for every n>=0;

30n^3+60000n^4+4000n^2-10<=10020n^4, for every n>=0, C=10020, n0=0;

O(n^4);

30n^3+60000n^4+4000n^2-10>=6000n^4, for every n>=1, C=1, n0=1;

Ω(n^4);

30n^3+60000n^4+4000n^2-10 is Θ(n^4);

(c).10000000n^2+0.0000001n^4 is O(n^2)

10000000n^2+0.0000001n^4<=cn^4 for every n>=n0;

10000000n^2+0.0000001n^4<=(10000000.0000001)n^4, for every n>=1, C=10000000.0000001, n0=1;

10000000n^2+0.0000001n^4 is not O(n^2);

10000000n^2+0.0000001n^4 is O(n^4);

(D).1000000n^2+0.0000001n^3 is Ω(n^3)

1000000n^2+0.0000001n^3>=0.0000001n^3, for every n>=1, C=0.0000001, n0=1;

1000000n^2+0.0000001n^3 is Ω(n^3);

(e).n! isΩ(2^n)

n!=n\*(n-1)\*(n-2)\*……\*2\*1>=c2^n=c(2\*2\*2\*……2\*2), for every n>=n0;

n!>=2^n, for every n>=1, C=1, n0=1;

n! isΩ(2^n);

(f).n^n is O(n!)

n^n=n\*n\*n\*n……\*n\*n<= cn!=c(n\*(n-1)\*(n-2)\*……\*2\*1), for every n>=n0;

this is false, because “n\*n\*n\*n……\*n\*n” is bigger than “n\*(n-1)\*(n-2)\*……\*2\*1”,(e.g. when n>=2, n^n=2^2=4, but n!=2\*1=2, and 4>2), so this is wrong.