# Solved problems for time complexity of loops

Last updated 9/7/2021

General comments/hints

If a variable is decreasing (e.g. i=N; i>=1; i--) you can write the summation in increasing order () without doing a change of variable.

lg = log2

A) 6.

// int mistery(int len, int v); has Θ(len2)

for(i=N; i>=0; i--){

int res = mistery(N, i); // 🡺 Θ(N2)

for(k=N; k>=1; k=k-1)

printf("B");

}

for-k: TC1iter( **k** ) = Θ( 1 ) dependent / independent

Change of var: no. consecutive

/ repetitions : Θ(N) repetitions => Θ(N) \* Θ(1)

Closed form: Θ(N) \* Θ(1) Θ( **N** )

for-i: TC1iter( **i** ) = Θ(N2) + Θ( N ) = Θ(N2+N) = Θ(N2) dependent / independent

mistery for-k

Change of var: No. consecutive

/ repetitions: Θ(N) repetitions => Θ(N) \* Θ(N2)

Closed form: Θ(N) \* Θ(N2) Θ( **N3** )

A) 7.

// int mistery(int N, int v); has Θ(N2)

for(i=N; i>=0; i--){

int res = mistery(i, i); // 🡺 Θ(i2)

for(k=N; k>=1; k=k-1)

printf("B");

}

for-k: TC1iter( **k** ) = Θ( 1 ) dependent / independent

Change of var: no. consecutive

/ repetitions : N

Closed form: N\* Θ(1) Θ( **N** )

for-i: TC1iter( **i** ) = Θ(i2) + Θ( N ) = Θ(i2+N) **dependent** / independent

mistery for-k If am not sure how i2 and N will contribute, I’ll keep both. It is the safest choice.

Change of var: No. consecutive

**/** repetitions:

Closed form: Θ( )

A) 8

for (i = 101; i<=(100+N); i++)

for (k=1; k<=i; k = k+1)

printf("B ");

for-k: TC1iter( **k** ) = Θ( 1 ) independent

Change of var: \_\_no\_\_ **/ repetitions** \_\_i\_ Closed form: i\* Θ(1) Θ( **i** )

for-i: TC1iter( **i** ) = Θ( i ) dependent

Change of var: \_\_NO\_\_\_\_

/ repetitions

Closed form: Θ( )

A) 9

for (i=1; i<=N; i=i\*2)

for (k=1; k<=2\*i; k=k+1)

for (t=0; t<7; t=t+1)

printf("A");

for-t: TC1iter( **t** ) = Θ( 1 ) independent

Change of var: No

/ repetitions: 7 repetitions => 7 \* Θ(1)

Closed form: 7\* Θ(1) Θ( **1** ) (7 is a constant. It must be dropped. Keep 1.)

for-k: TC1iter( **k** ) = Θ( 1 ) independent

Change of var: no

/ repetitions: about 2i repetitions => 2i \* Θ(1)

Closed form: 2i\* Θ( 1 ) Θ( **i** )

for-i: TC1iter( **i** ) = Θ( 2i ) dependent

Change of var: yes: i: 1,2,4,8,16,…, i=2x, …, 2p = ilast = N => p = lgN

/ repetitions: Note: do not leave it in p. Write p as an expression of N

Closed form: Θ( N )

A) 10

for(t=1; t<=N; t=t\*3)

for(i=10; i<=N; i=i+2)

for(k=N; k>=1; k=k-5)

printf("G");

for-k: TC1iter( **k** ) = Θ( 1 ) independent

Change of var: yes: k: N, N-5, N-10, N-15,… , N-5x, …, (N-5p)=klast=1 => p = (N-1)/5

/ **repetitions:** (N-1)/5 => (N-1)/5 \* Θ(1)

Closed form: (N-1)/5 \* Θ(1) Θ( **N** )

for-i: TC1iter( **i** ) = Θ( N ) independent

Change of var: yes: i = 10,12,14,16,…, 10+2x, …, 10+2p=ilast=N => p = (N-10)/2

/ repetitions: (N-10)/2 => (N-10)/2 \* Θ( N )

Closed form: (N-10)/2 \* Θ( N ) = N(N-10)/2 = Θ( N2 ) Θ( N2 )

for-t: TC1iter( **t** ) = Θ( N2 ) independent

Change of var: yes: t: 1,3,9,27,…, 3x, …3p=tlast=N => p = log3N

/ repetitions: log3N => log3N\* Θ(N2)

Closed form: log3N\* Θ(N2) Θ( N2 log3N )

A) 11

for(i=1; i<=N; i=i+1)

for(k=1; k<=i; k=k+k)

for(t=1; t<=S; t++)

printf("G");

for-t: TC1iter(**t**) = \_\_3 instr = **Θ (1)** ~~dependent~~ / independent

Change of var: No (t is consecutive)

∑ / repetitions : S => S\* Θ (1)

Closed form: S\* Θ (1) Θ(**S**)

for-k: TC1iter( **k** ) = 1 (k<=i)+1(k=k+k)+1(t=1)+ Θ(**S**)= **Θ(S)** ~~dependen~~t / independent

Change of var: Yes. k:1, 2, 4, 8, 16, …, 2x, …. 2p; k = 2x, 2p= klast = i => p = log2i

∑ / repetitions: p = log2i => log2i \* Θ(S)

Closed form: log2i \* Θ(S) Θ(**S log2i**)

for-i: TC1iter( **i** ) = 1+1+1+ Θ(**S log2i**)= Θ(**S log2i**) dependent / ~~independent~~

Change of var: No. (i takes consecutive values)

∑ / ~~repetitions~~: ∑Ni=1 Θ(**S log2i**) = ∑Ni=1 **S log2i = S\*** ∑Ni=1 (**log2i**)

~~Closed form: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Θ(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)~~

We will see later how to solve this summation.

A) 12

for (i=1; i<=N; i=i+1) {

for (k=1; k<=i; k++) {

printf("B");

for (t=1; t<=k; t++)

printf("C");

} // end for-k

…

} // end for-i

for-t: TC1iter( **t** ) = Θ( 1 ) independent

Change of var: no.

/ repetitions: k => k\* Θ(1)

Closed form: k\* Θ(1) Θ( **k** )

for-k: TC1iter( **k** ) = Θ( k ) dependent

Change of var: No

/ repetitions:

Closed form: Θ( )

for-i: TC1iter( **i** ) = Θ( ) dependent / independent

Change of var: no

/ repetitions:

Closed form: Θ( )

A) 13

// Assume void do\_smth(int N);has Θ(N)

for (i=1; i<=N; i=i+1) {

for (k=1; k<=i; k++) {

printf("B");

for (t=1; t<=k; t++)

do\_smth(k);

} // end for-k

…

} // end for-i

for-t: TC1iter( **t** ) = Θ( k ) independent

Change of var: no

/ repetitions: k => k\* Θ(k)

Closed form: k\* Θ(k) = Θ(k2) Θ( **k2** )

for-k: TC1iter( **k** ) = Θ( k2 ) dependent

Change of var: no

/ repetitions:

Closed form: Θ( )

for-i: TC1iter( **i** ) = Θ( i3 ) dependent / independent

Change of var: No

/ repetitions: (ok to leave it like this and not give closed form or Theta when approximation by integrals is not known).

Solving with approximation by integrals:

Closed form: Θ( )

A) 14

// example from Dr. Weems

for (i=1; i<=N-1; i++) {

…

for (k=i+1; k<=N; k++) {

temp=ab[ell][k];

ab[ell][k]=ab[i][k];

ab[i][k]=temp;

for (t=i+1; t<=N; t++)

ab[t][k] -= ab[t][i]\*ab[i][k];

}// end for-k

…

} // end for-i

for-t: TC1iter( **t** ) = Θ( 1 ) independent

Change of var: no.

/ repetitions: N-i => (N-i)\* Θ(1)

Closed form: (N-i)\* Θ(1) Θ( **N-i** )

for-k: TC1iter( **k** ) = Θ( N-i ) independent

Change of var: no

/ repetitions: N-i => (N-i)\* Θ(N-i)

Closed form: (N-i)\* Θ( N-i ) = (N-i)2 Θ( (N-i)2 )

for-i: TC1iter( **i** ) = Θ( (N-i)2 ) dependent

Change of var: no.

/ repetitions:

Closed form: Θ( N3 )

A) 16

for (i=1; i<=N; **i=i\*2**) {

for (k=0; k<=i; k=k+3) {

printf("B");

for (t=1; t<=k; t++)

printf("C");

} end for-k

…

} // end for-i

for-t: TC1iter( **t** ) = Θ( 1 ) / independent

Change of var: No

/ repetitions: k => k\* Θ(1)

Closed form: k\* Θ(1) Θ( **k** )

for-k: TC1iter( **k** ) = Θ( k ) dependent

Change of var: yes: k = 0, 3,6, 9, 12, … k=3x,….3p=klast=I => p = i/3

/ repetitions:

Closed form: Θ( i2 )

for-i: TC1iter( **i** ) = Θ( i2 ) dependent

Change of var: yes: i: 1, 2, 4, 8, 16, …, 2x, …2p=ilast=N => p = lgN

/ repetitions:

Closed form: Θ( )

**II. A.**  Give a piece of code with nested loops that has time complexity NlgN.

int j, k;

for(j = 1; j <= N; j=j+1) {

for(k = 1; k <= N; k = k\*2){

printf("A");

}}