Time complexity of nested loops

- Time complexity of loops should be calculated using: ***sum over the values* of the loop variable of the *time complexity of the body* of the loop**. This formulation is general and covers both the dependent (see $\sum_i \left(\frac{i}{3}\right) lgM$) and independent cases (see $\sum_t 1$ and $\sum_k (\frac{i}{3})$).

Dominant term with constant: $(N^2 lgM)/6$

```
for(i=1; i<=N; i++)
    for(k=1; k<=M; k=k*2)
        for(t=0; t<=i; t=t+3)
            printf("C");
```

→ $\sum_i [\left(\frac{i}{3}\right) lgM] = \sum_{i=1}^{N} [\left(\frac{i}{3}\right) lgM] = \frac{lgM}{3} \sum_{i=1}^{N} i = \frac{lgM}{3} \frac{N(N+1)}{2} = \Theta(N^2 lgM)$

→ $\sum_k (i/3) = \sum_{e=0}^{lgM} (i/3) = \Theta(\left(\frac{i}{3}\right) lgM)$ (Use only the dominant term(s))

→ $\sum_t 1 = \sum_{e=0}^{i/3} 1 = \left(\frac{i}{3} + 1\right) = \Theta(\frac{i}{3})$ (Use only the dominant term)

→ 1

Values of t: 0,3,6,9,…,t_{last}≤i , use t = 3e with
Values of e: 0,1,2,3,…,p => t_{last}=i=3p => p=i/3 =>
Values of e: 0,1,2,3,…,i/3. (e has consecutive values)
Change of variable: Replace t with e in summation over t

Values of k: 1,2,4,8,…,k_{last}≤M , use k = 2^e with
Values of e: 0,1,2,3,…,p =>
k_{last}=M= 2^p => (apply lg in both sides) => p=lgM =>
Values of e: 0,1,2,3,…,lgM. (e has consecutive values)
Change of variable: Replace k with e in summation over k

We do not need a change of variable for i because values of i are already 'good' (consecutive numbers starting at 1)

You can think that dependent and independent refers to the relation between the variable of a summation and the term of the summation. E.g. $\sum_k (i/3)$ is independent because the summation is over k, but k does not appear in the expression i/3.

If the constant for the dominant term is not needed, we can drop the /3 in i/3 and use only i (use $\sum_k i$ instead of $\sum_k (i/3)$).

Steps for computing the time complexity of loops:

1. Compute the time complexity of the BODY of the loop, T_{body}
2. Write a "loose" summation over the loop variable of the time complexity of the body (e.g. $\sum_k T_{body}$)
3. Summation must be over CONSECUTIVE values. Do a change of variable if needed. Does the loop variable (say k) go in consecutive values?
   1. Yes. Write the summation explicitly: $\sum_{k=1}^{N} T_{body}$
   2. No. DO change of variable: k = f(e) where e goes from 0 (or 1) to p in consecutive values. Use loop condition to solve for p (note that even if you have k<N, you can use k_{last} = f(p)=N since we do not need the exact count). Rewrite the summation using e in sigma, the expression you got for p and replacing k with f(e) in the term. (E.g. if k = 4e and k<N, solve: 4p=N => p = N/4, then do the change of variable: $\sum_k k^2 = \sum_{e=0}^{N/4}(4e)^2$ (Notice that p does not show in the final answer))
4. Solve the summation you got
5. Give Θ. (Keep the dominant term with the multiplication constant if needed)

** If this loop was nested in another, the answer from step 5 will be used to compute the time complexity of the body of that immediately outer loop (step 1 in calculations for that outer loop) , and the process continues.

Time complexity of nested loops

- Time complexity of loops should be calculated using: ***sum over the values* of the loop variable of the *time complexity of the body* of the loop**.

// assume given: $T_{foo}(t,M) = \Theta(t^2M)$

Dominant term with constant: $(N^3M)/12$

```
for(t=0; t<N; t=t+4)
    foo(t,M);
```

→ $\sum_t[t^2M] = \sum_{e=0}^{N/4}[(4e)^2M] = \sum_{e=0}^{N/4} 16Me^2 = 16M\sum_{e=0}^{N/4} e^2 = 16M\frac{\frac{N}{4}(\frac{N}{4}+1)(2\frac{N}{4}+1)}{6} = \Theta(N^3M)$

→ t^2M (Use only the dominant term(s))

Values of t: 0,4,8,12,…,t_{last}<N , use t = 4e with
Values of e: 0,1,2,3,…,p => t_{last}=N=4p => p=N/4 =>
Values of e: 0,1,2,3,…,N/4. (e has consecutive values)
Change of variable: Replace t with e in summation over t
(Note that even though the loop has t<N, we use t_{last} = N to solve for p, since we do not need the exact count)

Useful processing of summation techniques (for Θ or dominant term calculations)

Note that some of these will NOT compute the EXACT solution for the summation

Independent case (term in summation does not have the variable of the summation).

$$\sum_{k=1}^{N} S = S + S + S + \cdots . + S = NS \quad (= S\sum_{k=1}^{N} 1 = SN)$$

Pull constant in front of summation: $\sum_{k=1}^{N}(Sk) = S\sum_{k=1}^{N} k = S\frac{N(N+1)}{2} = \Theta(SN^2)$

Break summation in two summations

$$\sum_{k=1}^{N}(kS + k^2) = \sum_{k=1}^{N} kS + \sum_{k=1}^{N} k^2 = S\sum_{k=1}^{N} k + \sum_{k=1}^{N} k^2 = S\frac{N(N+1)}{2} + \frac{N(N+1)(2N+1)}{6} = \Theta(SN^2 + N^3)$$

Drop lower order term from summation term. E. g. $10k$ *is lower order compared to* k^2:

$$\sum_{k=1}^{N}(10k + k^2) = \sum_{k=1}^{N} k^2 = \frac{N(N+1)(2N+1)}{6} = \Theta(N^3)$$

Use approximation by integrals for increasing or decreasing $f(k)$:

$$\sum_{S}^{N} f(k) = \Theta\big(F(N) - F(S)\big) \ (where\ F\ is\ the\ antiderivative\ of\ f)$$

Formula for values of i and exact calculation of number of loop iterations – Example 1

```
for (i=0; i<=N; i=i+3)
    printf("A");
```

e	i=3e
0	0
1	3
2	6
3	9
…	…
e	3e
…	…
p	3p (i_{last} = **3p**, $i_{last} \le N$ $3p \le N \Rightarrow p = \lfloor N/3 \rfloor$

i takes values: **0,3,6,9,12,…. ≤ N**

We notice that these are consecutive multiples of 3 so we will explicitly show that by writing i as a function of another variable:

i = 3*e

Where **e takes values: 0,1,2,3,….,p**

Here we use p to refer to that last multiple of 3 that is ≤N.

The loop executes (the condition is true) for all i = 3e where e takes values: 0,1,2,3,…,p => 1+p total values (because of the 0) => the loop iterates 1+p times. (A)

Next we will compute the exact formula for p:

Because of how we chose p we have: **3p ≤ N<3(p+1)** (the next multiple of 3 will be strictly larger than N)

We care about 3p because it is the last value of i for which the loop condition is true (3p=i ≤ N).

We know that p is somewhat around N/3, but we need to figure out if it is rounded up or down.

If $\mathbf{N} \in [3\boldsymbol{p}, 3(\boldsymbol{p}+\mathbf{1}))$ *and we divide by* 3 *on both sides* $\Rightarrow \frac{N}{3} \in [\boldsymbol{p}, \boldsymbol{p}+\mathbf{1}) \Rightarrow \mathbf{p} = \left\lfloor \frac{N}{3} \right\rfloor$ (B)

From (A) and (B) it follows that the loop executes $1 + \mathrm{p} = 1 + \left\lfloor \frac{N}{3} \right\rfloor$ times =>

The loop executes exactly: $1 + \left\lfloor \frac{N}{3} \right\rfloor$ times.

As a verification step you should check that the formula does give the exact number of loop iterations for a few values of N: 0,1,2,3,4,15,17

Practice: how would you solve: `for (i=3; i<=N; i=i+3) printf("A");`

Formula for values of i and exact calculation of number of loop iterations – Example 2

```
for (i=2; i<=N; i=i+3)
    printf("A");
```

i takes values: **2,5,8,11,14**,…. ≤ N

We notice that these are consecutive multiples of 3 with an offset of 2. We will explicitly show that by writing i as a function of another variable:

i = 2+(3*e)

Where e takes values: 0,1,2,3,….,p

Here we use p to refer to that last value **2+3p that is ≤ N.**

The loop executes (the condition is true) for all i = 2+3e where e takes values: 0,1,2,3,…,p => 1+p total values (because of the 0) => the loop iterates 1+p times.

2+3p ≤ N => 3p ≤ (N-2) => p ≤ (N-2)/3, but p is an integer and largest with this property => $\mathbf{p} = \left\lfloor \frac{N-2}{3} \right\rfloor \Rightarrow$

FINAL ANSWER: The loop executes exactly: $\mathbf{1} + \left\lfloor \frac{N-2}{3} \right\rfloor$ times.

As a verification step you should check that the formula does give the exact number of loop iterations for a few values of N: **2,3,4,5,6** (Note that you start with the smallest value of N for which the loop iterates at least one time: 2. You do not use 0 or 1 for N in this verification.)

e	i=2+3e
0	2
1	5
2	8
3	11
…	…
e	i = 2+3e
…	…
p	i_{last} <=N (i_{last} =2+3p)

Formula for values of i and exact calculation of number of loop iterations – Example 3

```
for (i=1; i<=N; i=i*5)
    printf("A");
```

i takes values: **1,5,25,125,**….<=N

We notice that these are consecutive multiples powers of 5 so we will explicitly show that by writing i as a function of another variable:

$i = 5^e$

Where e takes values: 0,1,2,3,….,p

Here we use p to refer to that largest value **5^p that is ≤N.**

The loop executes (the condition is true) for all i = **5^e** where e takes values: 0,1,2,3,…,p => 1+p total values (because of the 0) => **the loop iterates 1+p times**. (A)

Next we will compute the exact formula for p.

Because of how we picked p we have: $\mathbf{5^p \le N < 5^{p+1}}$

take $\log_5$ on all sides ⇒ $\boldsymbol{p \le log_5 N < (p+1)}$

⇒ $\boldsymbol{p = \lfloor \log_5 N \rfloor}$ (B)

From (A) and (B) it follows that the loop executes $\mathbf{1 + p = 1 + \lfloor \log_5 N \rfloor}$ times =>

ANSWER: The loop executes exactly: $\mathbf{1 + \lfloor \log_5 N \rfloor}$ times.

As a verification step you should check that the formula does give the exact number of loop iterations for a few values of N: **1,5,25,26,29,30**

e	**$i=5^e$**
0	1
1	5
2	25
3	125
…	…
e	**$i=5^e$**
…	…
p	i_{last} <=N (i_{last} =**5^p**) => $p = \lfloor \log_5 N \rfloor$

Time complexity of nested loops (small variation of the first example)

- Time complexity of loops should be calculated using: ***sum over the values* of the loop variable of the *time complexity of the body* of the loop**. This formulation is general and covers both the dependent (see $\sum_i \left(\frac{i^2}{3}\right) lgM$) and independent cases (see $\sum_t 1$ and $\sum_k (\frac{i^2}{3})$.

Dominant term with constant: $(N^2 lgM)/6$

```
for(i=1; i<=N; i++)
    for(k=1; k<=M; k=k*2)
        for(t=0;t<=i*i; t=t+3)
            printf("C");
```

→ $\sum_i [\left(\frac{i^2}{3}\right) lgM] = \sum_{i=1}^{N} [\left(\frac{i^2}{3}\right) lgM] = \frac{lgM}{3} i^2 = \frac{lgM}{3} \frac{N(N+1)(2N+1)}{6} = \Theta(N^3 lgM)$

→ $\sum_k (i^2/3) = \sum_{e=0}^{lgM} (i^2/3) = \Theta(\left(\frac{i^2}{3}\right) lgM)$ (Use only the dominant term(s))

→ $\sum_t 1 = \sum_{e=0}^{i/3} 1 = \left(\frac{i^2}{3}\right) = \Theta(\frac{i^2}{3})$ (Use only the dominant term)

→ 1

Values of t: 0,3,6,9,…,t_{last}≤i , use t = 3e with
Values of e: 0,1,2,3,…,p => $t_{last}=i^2=3p$ => $p=i^2/3$ =>
Values of e: 0,1,2,3,…,$i^2/3$. (e has consecutive values)
Change of variable: Replace t with e in summation over t

Values of k: 1,2,4,8,…,k_{last}≤M , use $k = 2^e$ with
Values of e: 0,1,2,3,…,p =>
$k_{last}=M= 2^p$ => (apply lg in both sides) => p=lgM =>
Values of e: 0,1,2,3,…,lgM. (e has consecutive values)
Change of variable: Replace k with e in summation over k

We do not need a change of variable for i because values of i are already 'good' (consecutive numbers starting at 1)

You can think that dependent and independent refers to the relation between the variable of a summation and the term of the summation. E.g. $\sum_k (i^2/3)$ is independent because the summation is over k, but k does not appear in the expression $i^2/3$.

If the constant for the dominant term is not needed, we can drop the /3 in $i^2/3$ and use only i (use $\sum_k i$ instead of $\sum_k (i^2/3)$).