Data Structures

lecture 6

30-9-2022

Last Session Quick Revision

Algorithm efficiency.

- There are more than one solutions (algorithms) to any problem
- We need to choose most efficient one.
- Major factor affecting efficiency is repetitive operations
  - Recursion / Loops

- Therefore Loops are very important in efficiency calculation

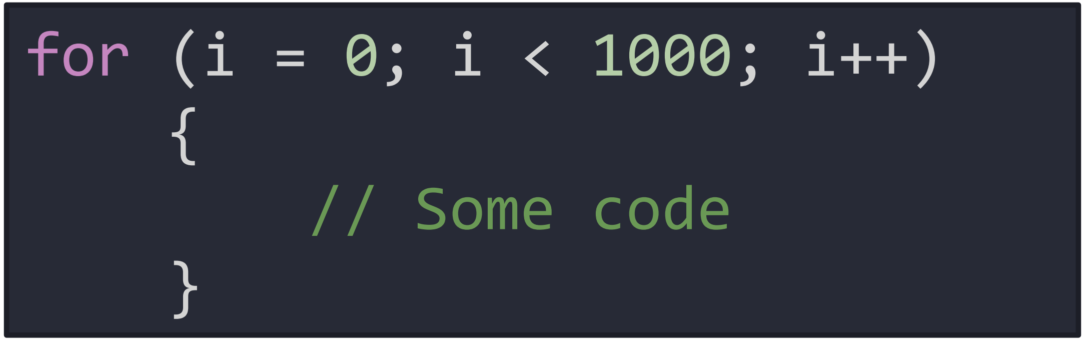
Efficiency

- Efficiency => mathematical function of the number of elements to be processed in loop

- f (n) = efficiency

Linear Loops

- How many times following code will execute



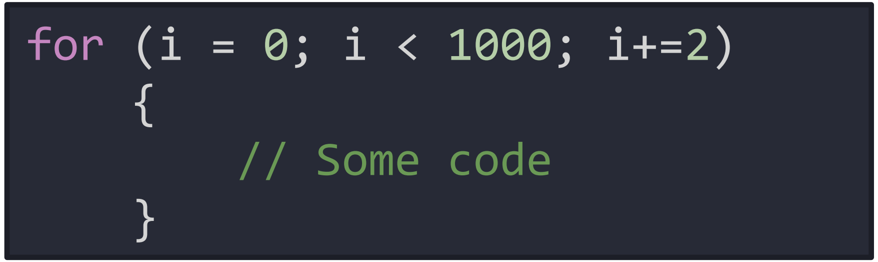
```
for (i = 0; i < 1000; i++)
    {
        // Some code
    }
```

Linear Loops

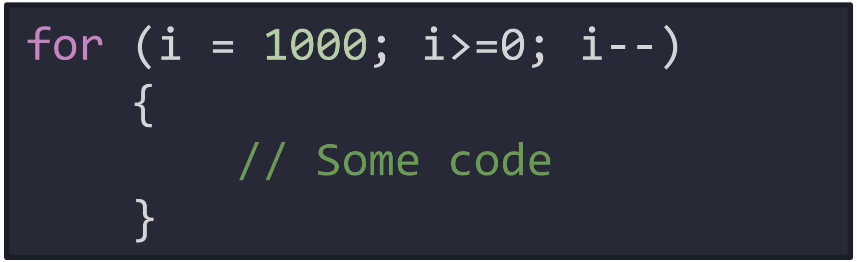
- Answer: 1000
- The number of iterations is directly proportional to the loop factor, 1000
- The higher the factor, the higher the number of loops.
- Because the efficiency is directly proportional to the number of iterations
  - $f(n) = n$

Linear Loops

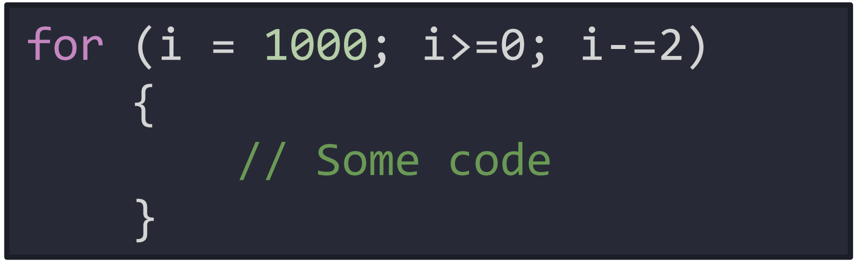
- Some times it is not straight forward



```
for (i = 0; i < 1000; i+=2)
    {
        // Some code
    }
```



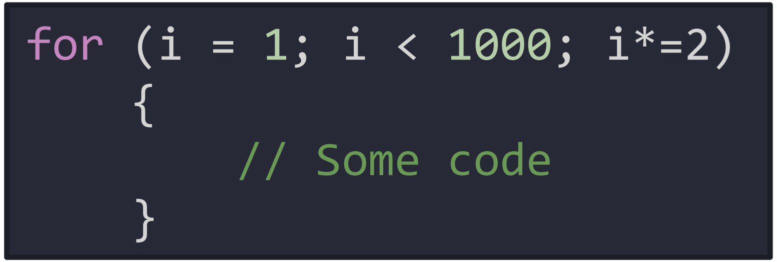
```
for (i = 1000; i>=0; i--)
    {
        // Some code
    }
```



```
for (i = 1000; i>=0; i-=2)
    {
        // Some code
    }
```

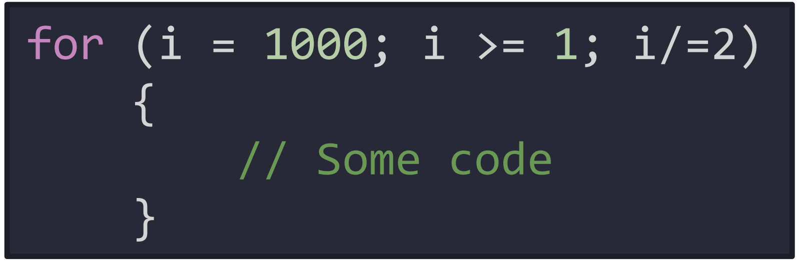
Logarithmic Loops

- How many times following code will execute



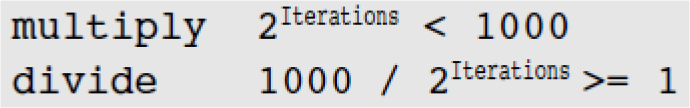
```
for (i = 1; i < 1000; i*=2)
    {
        // Some code
    }
```

Multiply Loops



```
for (i = 1000; i >= 1; i/=2)
    {
        // Some code
    }
```

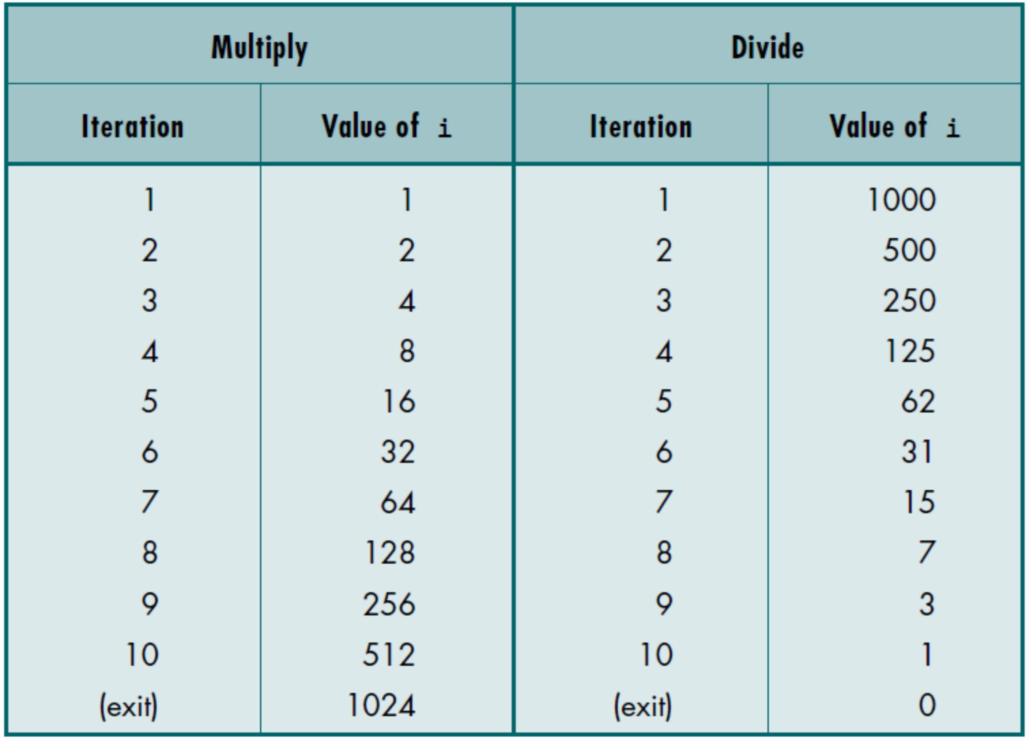
divide Loops



multiply $2^{\text{Iterations}} < 1000$

divide $1000 / 2^{\text{Iterations}} >= 1$

Logarithmic Loops



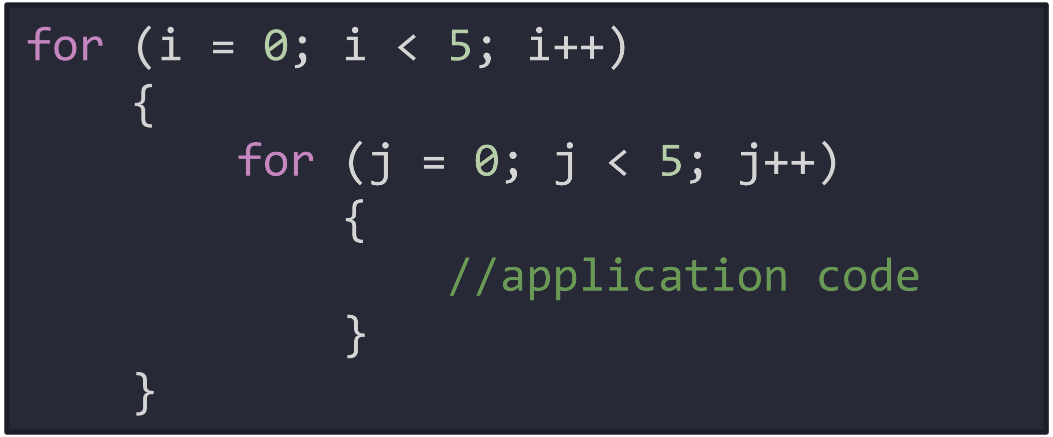
Multiply		Divide	
Iteration	Value of `i`	Iteration	Value of `i`
1	1	1	1000
2	2	2	500
3	4	3	250
4	8	4	125
5	16	5	62
6	32	6	31
7	64	7	15
8	128	8	7
9	256	9	3
10	512	10	1
(exit)	1024	(exit)	0

Logarithmic Loops

- $f(n) = \log n$

Nested Loops : Quadratic Loop

- How many times loop will execute



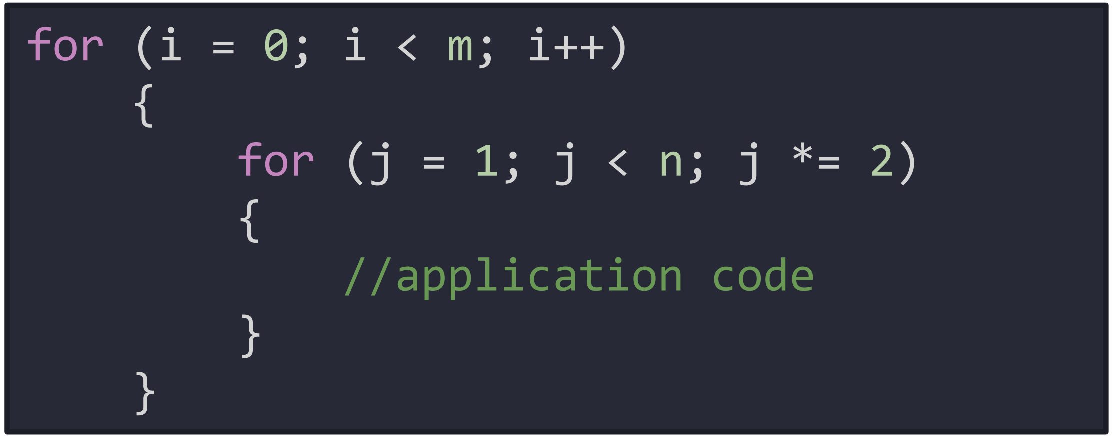
```
for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 5; j++)
            {
                //application code
            }
    }
```

Nested Loop: Quadratic

$$f(n) = n^2$$

Nested Loops

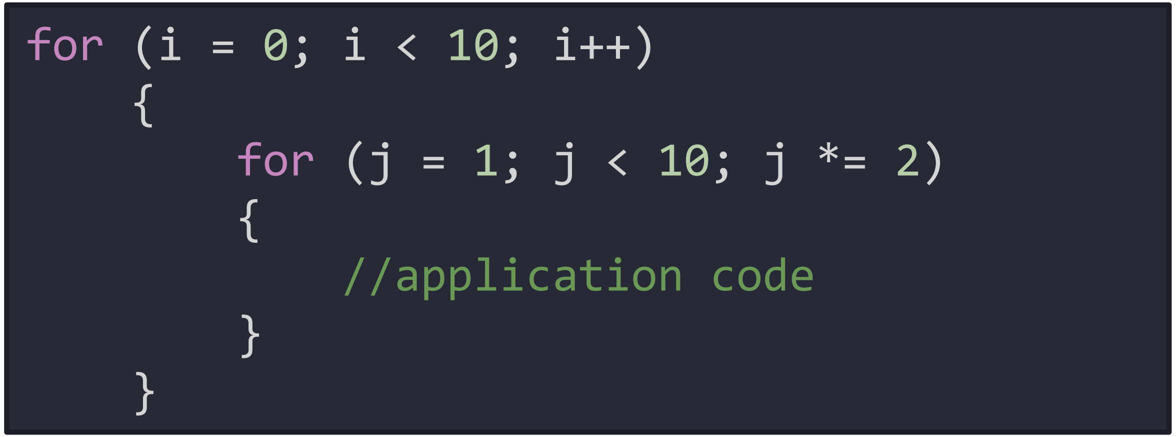
- Generalization



```
for (i = 0; i < m; i++)
    {
        for (j = 1; j < n; j *= 2)
        {
            //application code
        }
    }
```

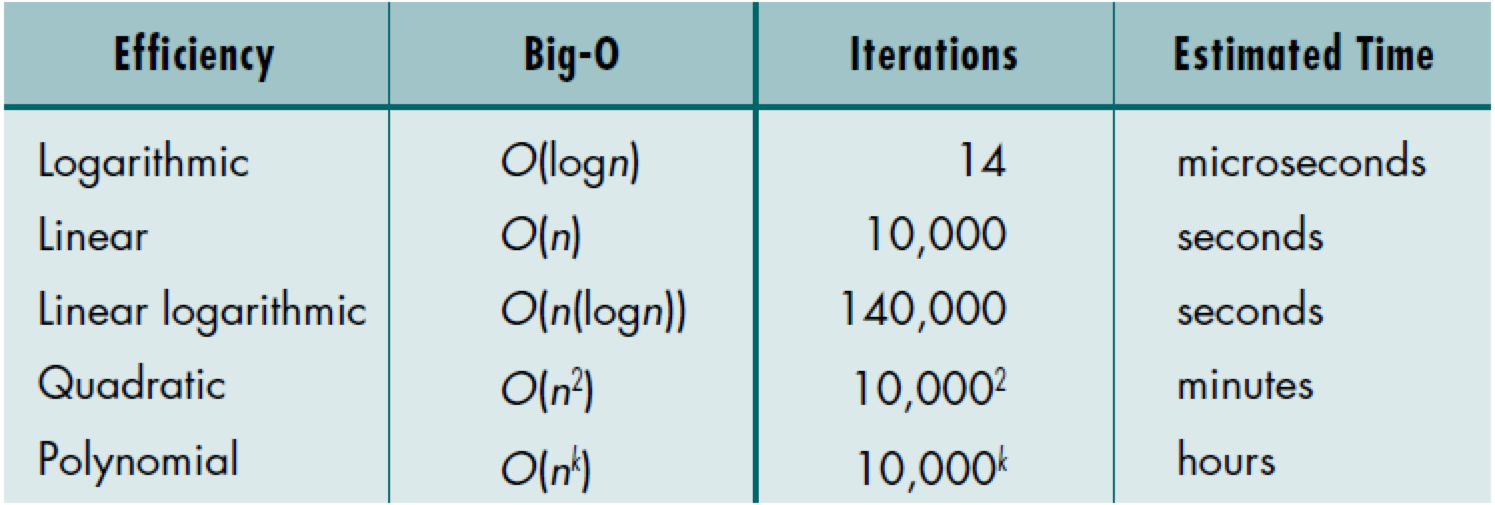
m * n

Nested Loop: Linear Logarithmic



```
for (i = 0; i < 10; i++)
    {
        for (j = 1; j < 10; j *= 2)
        {
            //application code
        }
    }
```

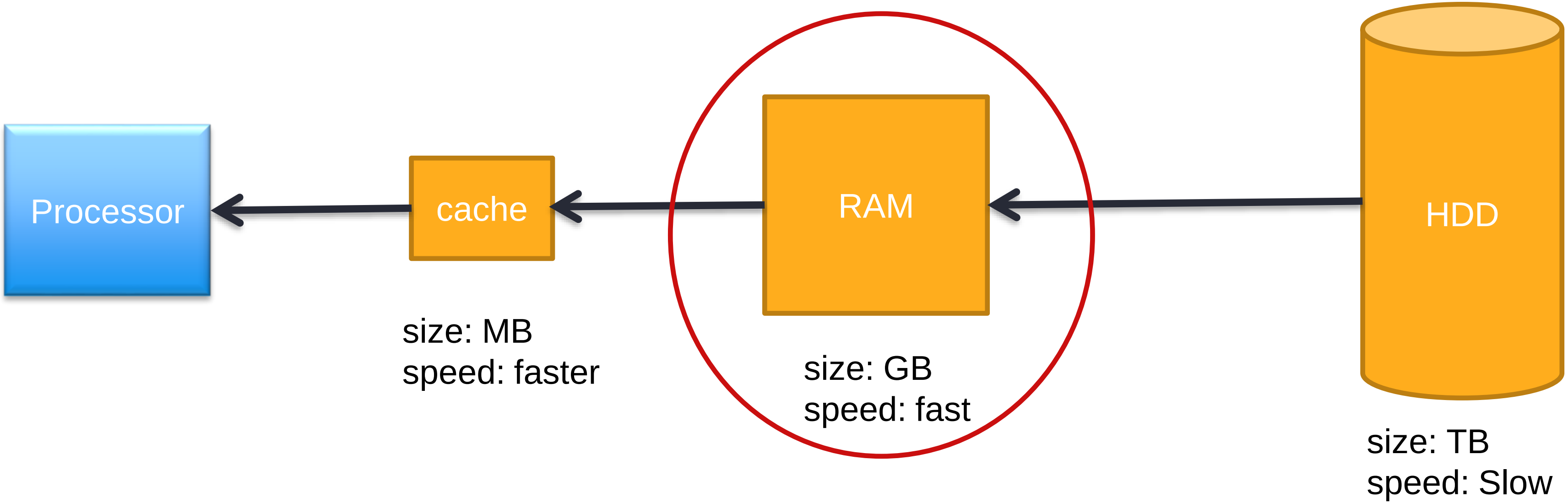
10log10



Efficiency	Big-O	Iterations	Estimated Time
Logarithmic	$O(\log n)$	14	microseconds
Linear	$O(n)$	10,000	seconds
Linear logarithmic	$O(n(\log n))$	140,000	seconds
Quadratic	$O(n^2)$	$10{,}000^2$	minutes
Polynomial	$O(n^k)$	$10{,}000^k$	hours

Basics of Memory Management

Memory Types

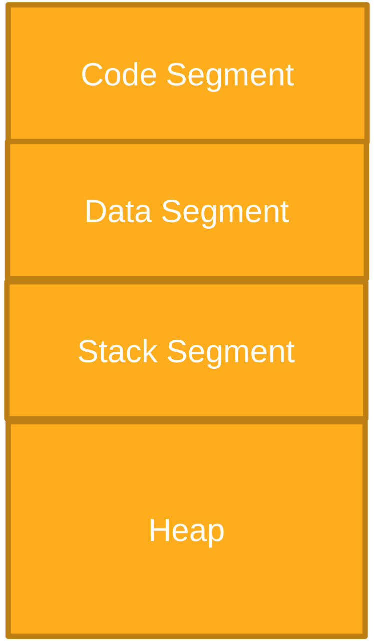


size: MB
speed: faster

size: GB
speed: fast

size: TB
speed: Slow

Closer look to RAM



Code Segment

Data Segment

Stack Segment

Heap

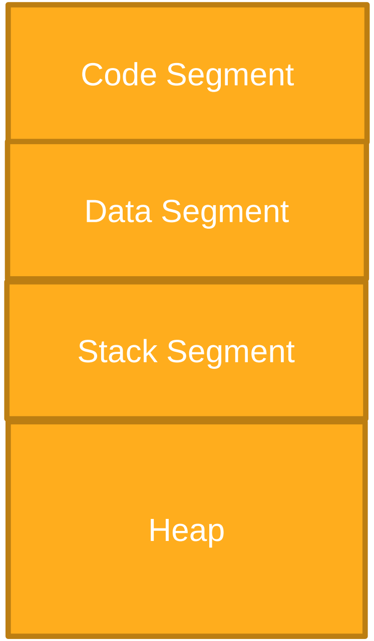
Code Segment:

- Stores plain statements
- Not useful for programmer from storage manipulation perspective

Data Segment

- Stores global and static variables
- Comparatively smaller

Closer look to RAM



Code Segment

Data Segment

Stack Segment

Heap

- Stack Segment (SS)
  - Stores local variables (variables in function)
  - As function called local variables are inserted on ss
  - As function returns (last line of function definition executes) variables are removed from stack.
  - Good Thing:
    - Memory management automatic
  - Bad Thing:
    - Limited in size

A closer look to pointers

- Pointer is a special variable which stores address of other variable.

Regular variable

•Stores value

•data-type var_name;

•int x

Pointer Variable

•Stores address

•data-type* var_name;

•int* p

Dereferencing a Pointer

- Finding out value at the address stored in the pointer
- int v = 10;
- int* ptr = &v;
- printf(“%d”,ptr);
- printf(“%d”,*ptr);