## Analysis of Algorithms

## Outline

(1) Performance

2 Time Complexity
(3) Space Complexity

Performance
Permance
路
$\square$
 $\square$ $\square$
 $\square$
 $\square$
 $\square$ $\square$ $\square$
 － $\square$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$ $>$



Algorithms are methods for solving computational problems

Algorithms are methods for solving computational problems
Data structures are schemes for arranging data, amenable to efficient processing by algorithms

Algorithms are methods for solving computational problems
Data structures are schemes for arranging data, amenable to efficient processing by algorithms

The performance characteristics of a program is determined by

- its time complexity, ie, how long it takes; and
- its space complexity, ie, how much memory it needs

Algorithms are methods for solving computational problems
Data structures are schemes for arranging data, amenable to efficient processing by algorithms

The performance characteristics of a program is determined by

- its time complexity, ie, how long it takes; and
- its space complexity, ie, how much memory it needs

The execution time of a program of size $n$ is a function $f(n)$ determined from the cost of executing each statement, and the frequency of execution of each statement

Algorithms are methods for solving computational problems
Data structures are schemes for arranging data, amenable to efficient processing by algorithms

The performance characteristics of a program is determined by

- its time complexity, ie, how long it takes; and
- its space complexity, ie, how much memory it needs

The execution time of a program of size $n$ is a function $f(n)$ determined from the cost of executing each statement, and the frequency of execution of each statement

The running time $T(n)$ of the program is an approximation of $f(n)$ obtained by ignoring any lower-order terms and constant coefficients

Algorithms are methods for solving computational problems
Data structures are schemes for arranging data, amenable to efficient processing by algorithms

The performance characteristics of a program is determined by

- its time complexity, ie, how long it takes; and
- its space complexity, ie, how much memory it needs

The execution time of a program of size $n$ is a function $f(n)$ determined from the cost of executing each statement, and the frequency of execution of each statement

The running time $T(n)$ of the program is an approximation of $f(n)$ obtained by ignoring any lower-order terms and constant coefficients

For example, if $f(n)=31 n^{2}+78 n+42$, then $T(n)=n^{2}$

- Command-line input: filename (String)
- Command-line input: filename (String)
- Standard output: the number of unordered triples $(x, y, z)$ from the file such that $x+y+z=0$


## Time Complexity

## Program: triplesum.py

- Command-line input: filename (String)
- Standard output: the number of unordered triples $(x, y, z)$ from the file such that $x+y+z=0$

```
>_ -/workspace/dsa/programs
$ cat ../data/1Kints.txt
    324110
-442472
    745942
$/usr/bin/time --format=%%e seconds' python3 triplesum.py ../data/1Kints.txt
70
0.7 seconds
$ /usr/bin/time --format='%e seconds' python3 triplesum.py ../data/2Kints.txt
528
5.9 seconds
```


## triplesum.py

```
from instream import InStream
import stdio
import sys
def main():
    inStream = InStream(sys.argv [1])
    a = inStream.readAllInts()
    stdio.writeln(count(a))
def count(a):
    n = len(a)
    count = 0
    for i in range(0, n):
        for j in range(i + 1, n):
            for k in range(j + 1, n):
                    if a[i] + a[j] + a[k] == 0:
                count += 1
    return count
if __name__ == '__main__':
    main()
```

Time Complexity

| $n$ | $f(n)$ |
| :--- | ---: |
| 1 K | 0.28 s |
| 2 K | 1.8 s |
| 4 K | 14.06 s |
| 8 K | 111.83 s |
| 16 K | 892.19 s |

Time Complexity

| $n$ | $f(n)$ |
| :--- | ---: |
| 1 K | 0.28 s |
| 2 K | 1.8 s |
| 4 K | 14.06 s |
| 8 K | 111.83 s |
| 16 K | 892.19 s |

$$
f(n)=0.2273121 n^{3}+0.007625303 n^{2}+0.006868505 n+0.01817256
$$

Time Complexity

| $n$ | $f(n)$ |
| :--- | ---: |
| 1 K | 0.28 s |
| 2 K | 1.8 s |
| 4 K | 14.06 s |
| 8 K | 111.83 s |
| 16 K | 892.19 s |

$f(n)=0.2273121 n^{3}+0.007625303 n^{2}+0.006868505 n+0.01817256$

$$
T(n)=n^{3}
$$

```
def count(a):
    n = len(a)
    count = 0
    for i in range (0, n):
    for j in range(i + 1, n)
        for k in range(j +
            if a[i] + a[j] + a[k] == 0:
                a[i] + a[j] + a[k] == 0:
    return count
```

Time Complexity

```
def count(a):
    n = len(a)
    count = 0
    for i in range(0, n):
        for j in range(i + 1, n):
            for k in range(j + 1, n)
            in range(j + 1, n): 
            if a[i] + a[j] + a[k]== 0:
                count += 1
                [A]
                LulE
    return count
```

| Statement Block | Time | Frequency | Total Time |
| :--- | :--- | :--- | :--- |
| $[A]$ | $t_{4}$ | 1 | $t_{4}$ |
| $[B]$ | $t_{3}$ | $n$ | $t_{3} n$ |
| $[C]$ | $t_{2}$ | $\binom{n}{2}^{1}=n^{2} / 2-n / 2$ | $t_{2}\left(n^{2} / 2-n / 2\right)$ |
| $[D]$ | $t_{1}$ | $\binom{n}{3}=n^{3} / 6-n^{2} / 2+n / 3$ | $t_{1}\left(n^{3} / 6-n^{2} / 2+n / 3\right)$ |
| $[E]$ | $t_{0}$ | $x$ (depends on input) | $t_{0} \times$ |

[^0]Time Complexity

```
def count(a):
    n = len(a)
    count = 0
    for i in range(0, n):
        for j in range(i + 1, n):
            for k in range(j + 1, n)
```



```
                f a[i] + a[j] +a[k] == 0:
    return count
```

| Statement Block | Time | Frequency | Total Time |
| :--- | :--- | :--- | :--- |
| $[A]$ | $t_{4}$ | 1 | $t_{4}$ |
| $[B]$ | $t_{3}$ | $n$ | $t_{3} n$ |
| $[C]$ | $t_{2}$ | $\binom{n}{2}^{1}=n^{2} / 2-n / 2$ | $t_{2}\left(n^{2} / 2-n / 2\right)$ |
| $[D]$ | $t_{1}$ | $\binom{n}{3}=n^{3} / 6-n^{2} / 2+n / 3$ | $t_{1}\left(n^{3} / 6-n^{2} / 2+n / 3\right)$ |
| $[E]$ | $t_{0}$ | $x$ (depends on input) | $t_{0} \times$ |

Grand total: $f(n)=\left(t_{1} / 6\right) n^{3}+\left(t_{2} / 2-t_{1} / 2\right) n^{2}+\left(t_{1} / 3-t_{2} / 2+t_{3}\right) n+t_{4}+t_{0} x$

$$
1\binom{n}{k}=\frac{n!}{k!(n-k)!}
$$

Time Complexity

```
def count(a):
    n = len(a)
    count = 0
    for i in range(0, n):
        for j in range(i + 1, n):
            for k in range(j + 1, n)
```



```
            \begin{array} { l } { \text { if a[i] + a[j] + a[k] == 0:} } \\ { \text { count += = [E]} } \end{array}
                [A]
    return count
```

| Statement Block | Time | Frequency | Total Time |
| :--- | :--- | :--- | :--- |
| $[A]$ | $t_{4}$ | 1 | $t_{4}$ |
| $[B]$ | $t_{3}$ | $n$ | $t_{3} n$ |
| $[C]$ | $t_{2}$ | $\binom{n}{2}^{1}=n^{2} / 2-n / 2$ | $t_{2}\left(n^{2} / 2-n / 2\right)$ |
| $[D]$ | $t_{1}$ | $\binom{n}{3}=n^{3} / 6-n^{2} / 2+n / 3$ | $t_{1}\left(n^{3} / 6-n^{2} / 2+n / 3\right)$ |
| $[E]$ | $t_{0}$ | $x$ (depends on input) | $t_{0} \times$ |

Grand total: $f(n)=\left(t_{1} / 6\right) n^{3}+\left(t_{2} / 2-t_{1} / 2\right) n^{2}+\left(t_{1} / 3-t_{2} / 2+t_{3}\right) n+t_{4}+t_{0} x$ Running time: $T(n)=n^{3}$

$$
1\binom{n}{k}=\frac{n!}{k!(n-k)!}
$$

## Running time classifications

| Name | $T(n)$ | Code Description | Example |
| :--- | :--- | :--- | :--- |
| constant | 1 | statement | increment the ith element in an array |
| logarithmic | $\log n$ | divide and discard | binary search |
| linear | $n$ | loop | find the maximum |
| linearithmic | $n \log n$ | divide and conquer | merge sort |
| quadratic | $n^{2}$ | double loop | check all ordered pairs |
| cubic | $n^{3}$ | triple loop | check all ordered triples |
| exponential | $2^{n}$ | exhaustive search | check all subsets |

## Space Complexity <br> Space <br> $\qquad$

The sizes of objects of built-in types differ from system to system, so the sizes of data types that we create also differ accordingly

## Space Complexity

The sizes of objects of built-in types differ from system to system, so the sizes of data types that we create also differ accordingly

The function call sys.getsizeof(x) returns the number of bytes that a built-in object x consumes on a particular system

## Space Complexity

The sizes of objects of built-in types differ from system to system, so the sizes of data types that we create also differ accordingly

The function call sys.getsizeof $(\mathrm{x})$ returns the number of bytes that a built-in object x consumes on a particular system
Sizes of built-in objects on a typical system

| Object | Size in Bytes |
| :--- | :--- |
| integer | 24 |
| float | 24 |
| boolean | 24 |
| string of $n$ characters | $40+n$ |
| list of $n$ integers | $72+8 n+24 n=72+32 n$ |
| $m$-by- $n$ list of integers | $72+8 m+m(72+32 n)=72+80 m+32 m n$ |
| user-defined | hundreds of bytes, at least |


[^0]:    $1\binom{n}{k}=\frac{n!}{k!(n-k)!}$

