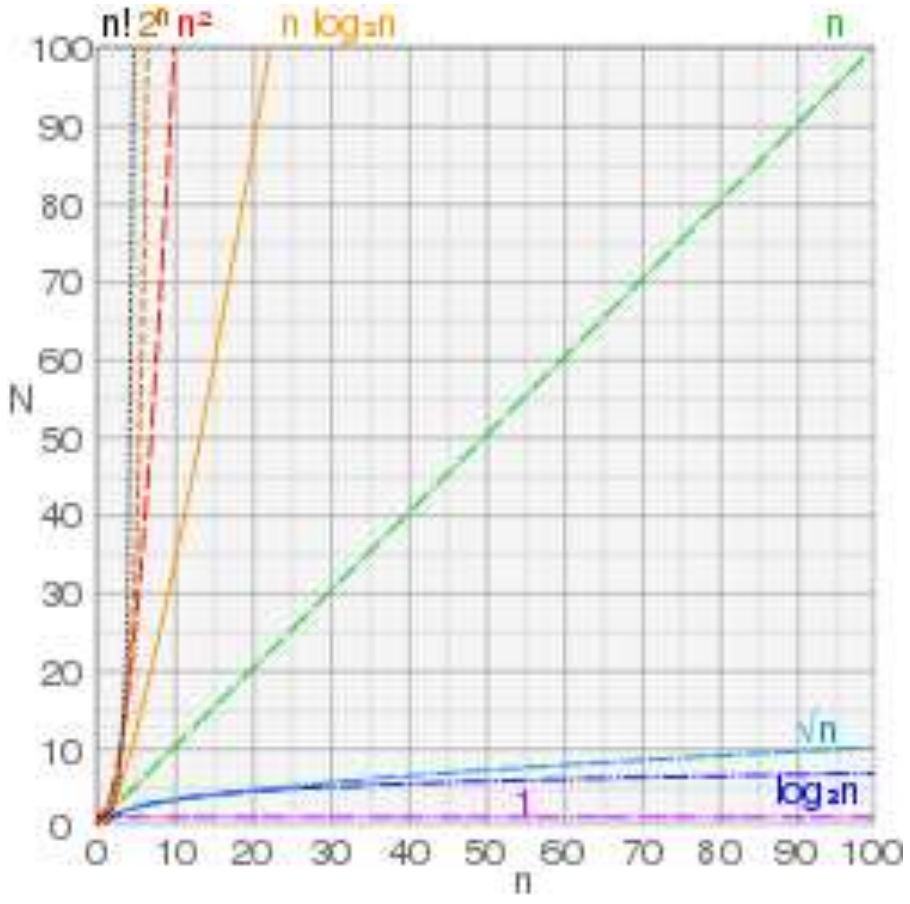
Time Complexity of Algorithms

Graphs of functions

- Show the number of operations *N* versus input size *n* for each function.
- Used in the analysis of algorithms

#Operations



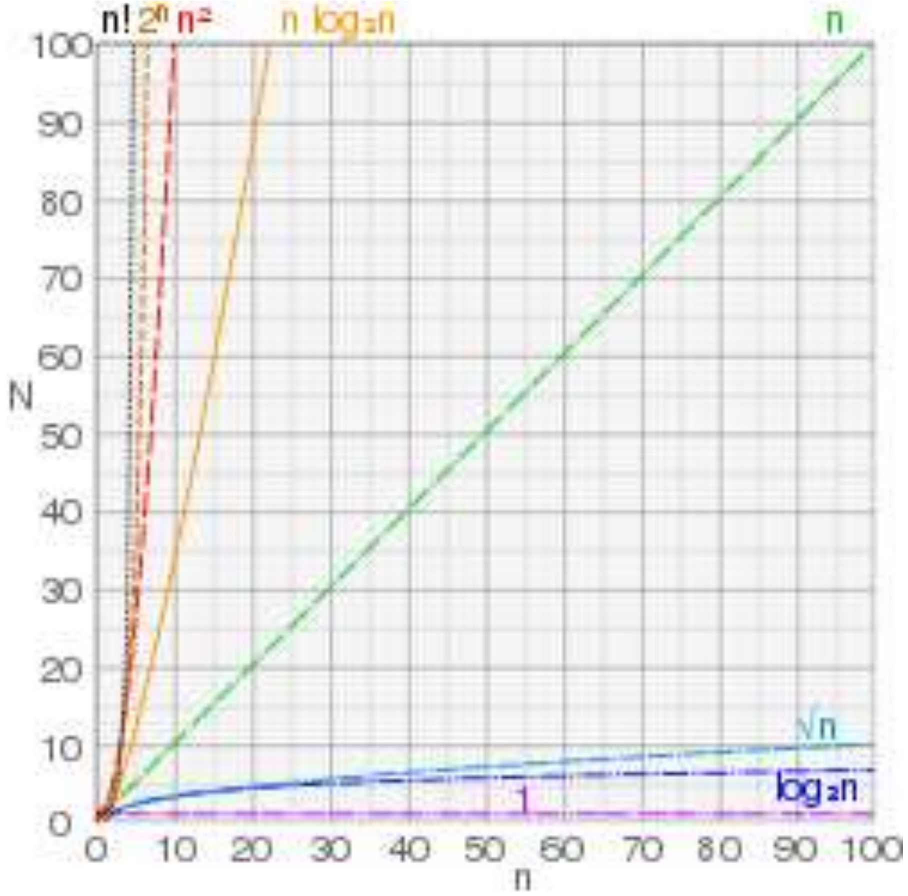
Input size

https://en.wikipedia.org/wiki/Time_complexity

Graphs of functions

#Operations

- Show the number of operations *N* versus input size *n* for each function.
- Used in the analysis of algorithms



Input size

https://en.wikipedia.org/wiki/Time_complexity

Time Complexity

- **Time complexity** of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input.

- The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms.

- **Asymptotical time complexity**: the time complexity of an algorithm as the input size goes to infinity.
- e.g.: if the time required by an algorithm on all inputs of size n is at most $5n^3 + 3n$ for any n (bigger than some n_0), the asymptotic time complexity is $O(n^3)$.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

~~What is the time complexity of the function foo?~~

Let's trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let’s trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let’s trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let’s trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let’s trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let's trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let's trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let’s trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?
Let's trace how this program runs.

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
	Parameter passing and function call	1
	Initialize counter i	1
	i < n	n+1
	++i	n
	Pass parameter to a.push_back	n
	Invoke a.push_back	n
	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
	A1	1
	A2	1
	A3	n+1
	A4	n
	A5	n
	A6	n
	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
	Parameter passing and function call	1
	Initialize counter i	1
	i < n	n+1
	++i	n
	Pass parameter to a.push_back	n
	Invoke a.push_back	n
	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	A1
C1	Initialize counter i	A2
C2	i < n	A3
C3	++i	A4
C4	Pass parameter to a.push_back	A5
C5	Invoke a.push_back	A6
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = = O(n)	

$C6 = C2+C3+C4+C5$

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n)	

C6 = C2+C3+C4+C5

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}

void g() {
        cin >> n;
        foo(n);
}
```

What is the time complexity of the function foo?

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n*C6) = O(n)	

$C6 = C2+C3+C4+C5$

An Example

```
vector a;
int n;
void foo(int n) {
        for (int i = 0; i < n; ++i )
                a.push_back( i );
}
```

Intuitive idea

Assume that a.push_back takes k units of time to execute and k is a fixed value. Also, there is an extra amount of time to set the for-loop, which is c.

Then it will take c + k * n units of time.

Thus:

Time complexity of the algorithm

= c + k*n = O(c + k*n) = O(n).

As n -> infinity, c is ignored. k is a constant, it is eliminated.

Cost term	Item & Purpose	Times
C0	Parameter passing and function call	1
C1	Initialize counter i	1
C2	i < n	n+1
C3	++i	n
C4	Pass parameter to a.push_back	n
C5	Invoke a.push_back	n
C = total cost	C0+C1 +(n+1)*C2+n*C3+n*C4 +n*C5 = C0+C1+C2 + n*(C6) = O(n*C6) = O(n)	

C6 = C2+C3+C4+C5

Time Complexity

Big-O: Computational complexity: refer to the upper bound for the asymptotic computational complexity of an algorithm or a problem, which is usually written in terms of the big-O notation, e.g., $O(n^3)$.

Big Omega $\Omega(n)$: Other types of (asymptotic) computational complexity estimates are lower bounds ("Big Omega" notation; e.g., $\Omega(n)$) and asymptotically tight estimates.

When the asymptotic upper and lower bounds coincide (written using the "big Theta"; e.g., $\Theta(n \log n)$).

asymptotic: approaching a value or curve arbitrary closely

Time Complexity

Time complexity can be one of the followings:

- $O(1)$ constant time
- $O(\log n)$
- $O(n)$ linear
- $O(n \log n)$
- $O(n^2)$ quadratic
- $O(n^3)$
- $O(2^n)$ exponential
- $O(n^{\log n})$, and more

Examples

- Assume k is a constant. n is varying (depends on input size). Use big-O notation to indicate the following functions:

1. 5
2. 5 + k
3. k + k *k
4. n*n+n
5. $n*(k*n+k^5)$
6. n*k + n * log n

Examples

- Assume k is a constant. n is varying (depends on input size). Use big-O notation to indicate the following functions:

1. 5 = A1
2. 5 + k = A2
3. k + k *k = A3
4. n*n+n = A4
5. n*(k*n+k^5) = A5
6. n*k + n * log n = A6

Examples

- Assume k is a constant. n is varying (depends on input size). Use big-O notation to indicate the following functions:

1. $5 = O(1)$
2. $5 + k = O(1)$
3. $k + k * k = O(1)$
4. $n*n+n = O(n^2)$
5. $n*(k*n+k^5) = O(n^2)$
6. $n*k + n * \log n = O(n \log n)$

Example

```
bool find(int key, const vector<int> &a) {
	bool flg = false;
	for (int i = 0; i < a.size(); ++i) {
		if ( a[i] == key) {
			flg = true; break;
		}
	}
	return flg;
}
```

Example

```
bool find(int key, const vector<int> &a) {   // C0
        bool flg = false;                     // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

Example

```
bool find(int key, const vector<int> &a) {  // C0
        bool flg = false;                   // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

Example

```
bool find(int key, const vector<int> &a) {   // C0
        bool flg = false;                     // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

(1, n+1, n), (C2, C3, C4)

Example

```
bool find(int key, const vector<int> &a) {   // C0
        bool flg = false;                     // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

(1, n+1, n), (C2, C3, C4)

n, C5

Example

```
bool find(int key, const vector<int> &a) {  // C0
        bool flg = false;                    // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

(1, n+1, n), (C2, C3, C4)

n, C5

1, C6

Example

```
bool find(int key, const vector<int> &a) {  // C0
        bool flg = false;                    // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
n = a.size( )
```

(1, n+1, n), (C2, C3, C4)

n, C5

1, C6

1, C7

Example

```
bool find(int key, const vector<int> &a) {  // C0
        bool flg = false;                     // C1
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
```

O(n), where n is the size of a.

(1, n+1, n), (C2, C3, C4)

n, C5

1, C6

1, C7

Example

```
bool find(int key, const vector<int> &a) {
        bool flg = false;
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
```

Assume there are m keys. We need to find whether they appear in a. What is the time complexity?

Example

```
bool find(int key, const vector<int> &a) {
        bool flg = false;
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
```

Assume there are m keys. We need to find whether they appear in a. What is the time complexity? m*O(n) = O(m n)

Example

```
bool find(int key, const vector<int> &a) {
    bool flg = false;
    for (int i = 0; i < a.size(); ++i) {
        if ( a[i] == key) {
            flg = true; break;
        }
    }
    return flg;
}
```

Assume that there are m keys to find. **If m is a constant**, what is the time complexity?

Example

```
bool find(int key, const vector<int> &a) {
        bool flg = false;
        for (int i = 0; i < a.size(); ++i) {
                if ( a[i] == key) {
                        flg = true; break;
                }
        }
        return flg;
}
```

Assume that there are m keys to find. **If m is a constant**, what is the time complexity?

$m*O(n) = O(m\ n) = O(n)$

Big-O notation

That is, $f(x) = O(g(x))$ if and only if there exists a positive real number M and a real number x_0 such that

$| f(x) | <= M | g(x) |$, for all $x >= x_0$.

Big-O notation

That is, $f(x) = O(g(x))$ if and only if there exists a positive real number M and a real number x_0 such that

$| f(x) | <= M | g(x) |$, for all $x >= x_0$.

1.	5	= O(1)
2.	5 + k	= O(1)
3.	k + k *k	= O(1)
4.	x*x+x	= $O(x^2)$
5.	$x*(k*x+k^{5000000})$	= $O(x^2)$
6.	x*k + x * log x	= O(x log x)

k is a constant.

Supplemental Materials

Insert an element to a linked list

```
void insert( Node *node ) {
      node->next = head;
    if (!head) tail = node;
      head = node;
}
```

What is the time complexity?

Append an element to a linked list

```
void append( Node *node ) {
	if (tail) {
		tail->next = node;
		tail = node;
	} else {
		head = tail = node;
		node->next = nullptr;
	}
}
```

What is the time complexity?

Find an element in a linked list

```
bool find ( int key ) {
	Node *t = head;
	while (t) {
		if (t->key == key) return true;
		t = t->next;
	}
	return false;
}
```

What is the time complexity?

Bubble sort

```
void bubbleSort( int a[], int n) {
    while (true) {
        bool swap_flg = false;
        for (int i = 0; i < n-1; ++i ) {
            if (a[i] > a[i+1]) {
                swap(a[i], a[i+1]);
                swap_flg = true;
            }
        }
        if ( !swap_flg ) break;
    }
}
```

What is the time complexity?

Bubble sort

```
void bubbleSort( int a[ ], int n) {
    while (true) {
        bool swap_flg = false;
        for (int i = 0; i < n-1; ++i ) {
            if (a[i] >= a[i+1]) {            // Can we use >=?
                swap(a[i], a[i+1]);
                swap_flg = true;
            }
        }
        if ( !swap_flg ) break;
    }
}
```

What is the time complexity?

Big-O notation

Let f and g be two functions defined on some subset of the real numbers. One writes

$f(x) = O(g(x))$, as $x \rightarrow \infty$

if and only if there is a positive constant M such that for all sufficiently large values of x, the absolute value of f(x) is at most M multiplied by the absolute value of g(x).