

running time:

```
Stopwatch stopwatch = new Stopwatch();
(client code)
double time = stopwatch.elapsedTime();
```

二分查找:

```
public static int binarySearch (int[] a, int key) {
    int lo = 0, hi = a.length-1;

    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        if (key < a[mid]) hi = mid - 1;
        else if (key > a[mid]) lo = mid + 1;
        else return mid;
    }
    return -1;
}
```

order of growth	name	typical code framework	description	example	$T(2N) / T(N)$
1	constant	$a = b + c;$	statement	add two numbers	1
$\log N$	logarithmic	<pre>while (N > 1) { N = N / 2; ... }</pre>	divide in half	binary search	~ 1
N	linear	<pre>for (int i = 0; i < N; i++) { ... }</pre>	loop	find the maximum	2
$N \log N$	linearithmic	[see mergesort lecture]	divide and conquer	mergesort	~ 2
N^2	quadratic	<pre>for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) { ... }</pre>	double loop	check all pairs	4
N^3	cubic	<pre>for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) for (int k = 0; k < N; k++) { ... }</pre>	triple loop	check all triples	8
2^N	exponential	[see combinatorial search lecture]	exhaustive search	check all subsets	$T(N)$

三和问题:

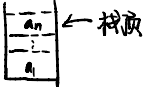
```
public class ThreeSum {
    public static int count (int[] a) {
        int N = a.length;
        int count = 0;
        for (int i = 0; i < N; i++)
            for (int j = i+1; j < N; j++)
                for (int k = j+1; k < N; k++)
                    if (a[i] + a[j] + a[k] == 0)
                        count++;
        return count;
    }

    public static void main (String[] args) {
        In in = new In( args[0]);
        int[] a = in.readAllInts();
        StdOut.println( count( a));
    }
}
```

Ex 1. Array accesses for brute-force 3-SUM. Ex 2. Compares for binary search.

Best: $\sim \frac{1}{2} N^3$ Worst: $\sim \frac{1}{2} N^3$
Average: $\sim \frac{1}{2} N^3$ Worst: $\sim \lg N$

栈 创建栈: `Stack<Integer> stack = new Stack<>();`



`push()`: 在栈顶插入元素
`pop()`: 弹出栈顶元素并删除
`peek()`: 获取栈顶元素但不删除
`isEmpty()`: 判断是否为空

队列 `Queue<Integer> queue = new Queue<>();`

	抛出异常		返回特殊值
插入	<code>add(e)</code>	超容量	返回 false \rightarrow <code>offer(e)</code>
删除	<code>remove()</code>	容量为0	返回 false \rightarrow <code>poll()</code>
检查	<code>element()</code>	容量为0	返回 null \rightarrow <code>peek()</code>

```
public class LinkedStackOfStrings {
    private Node first = null;

    private class Node {
        String item;
        Node next;
    }

    public boolean isEmpty() {
        return first == null;
    }

    public void push (String item) {
        Node oldfirst = first;
        first = new Node();
        first.item = item;
        first.next = oldfirst;
    }

    public String pop() {
        String item = first.item;
        first = first.next;
        return item;
    }
}
```

```
public class LinkedQueueOfStrings {
    private Node first, last;

    private class Node
    { /* same as in LinkedStackOfStrings */ }

    public boolean isEmpty()
    { return first == null; }

    public void enqueue (String item) {
        Node oldlast = last;
        last = new Node();
        last.item = item;
        last.next = null;
        if (isEmpty()) first = last;
        else oldlast.next = last;
    }

    public String dequeue() {
        String item = first.item;
        first = first.next;
        if (isEmpty()) last = null;
        return item;
    }
}
```