# 第二章：二分法（Binary Search）

二分法查找适用于有序序列（数组，链表等），基本思想是每次查询序列中间的那个数，通过这个数与target的比较来缩小搜索区间（每次舍弃一半的区间），从而高效完成搜索。

九章算法给出的二分法模板：

第一步（循环条件）：while (start + 1 < end);

说明：避免死循环。采用start + 1 < end的方式永远不会死循环，缺点则是需要对start和end进行判断。

第二步（求中点）：mid = start + (end – start) / 2;

说明：许多人习惯写成mid = (start + end) / 2; 这样的坏处是当start和end的值过大（Integer.MAX\_VALUE）时会发出溢出。

第三步（比较）：将A[mid]与target进行比较，根据==, >, <来确定

下一步搜索的区间。

第四步（判断）：可以看到，采用上面的方式会造成A[start]和A[end]没有进行比较，所以在最后要判断一下A[start]和A[end]哪个与target相等。

二分查找代码示例：

1. class Solution {
2. /\*\*
3. \* @param nums: The integer array.
4. \* @param target: Target to find.
5. \* @return: The first position of target. Position starts from 0.
6. \*/
7. public int binarySearch(int[] nums, int target) {
8. //write your code here
9. int pos = -1;
10. if (nums == null || nums.length == 0){
11. return pos;
12. }
13. int low = 0;
14. int high = nums.length - 1;
15. while (low + 1 < high){
16. int mid = (low + high) / 2;
17. if (target <= nums[mid]){
18. high = mid;
19. } else {
20. low = mid;
21. }
22. }
23. if (nums[low] == target) {
24. return low;
25. } else if (nums[high] == target) {
26. return high;
27. } else {
28. return -1;
29. }
30. }
31. }