# 题目

实现int sqrt(int x)函数。

计算并返回x的平方根，其中x是非负整数。

由于返回类型是整数，结果只保留整数的部分，小数部分将被舍去。

**示例 1:**

输入: 4

输出: 2

**示例 2:**

输入: 8

输出: 2

说明: 8 的平方根是2.82842...,

由于返回类型是整数，小数部分将被舍去。

# 分析

## 方法一：二分查找

思路：

我们想找一个最大整数 ans，使得：

ans \* ans <= x

用二分搜索 1 到 x（或者更优化一点是 1 到 x / 2）来逼近这个最大 ans。

代码：

class Solution {

public:

int mySqrt(int x) {

if (x == 0 || x == 1) return x;

int left = 1, right = x / 2; // 优化边界

int ans = 0;

while (left <= right) {

int mid = left + (right - left) / 2;

// 使用 long long 避免 mid \* mid 溢出

long long square = (long long)mid \* mid;

if (square == x) {

return mid;

} else if (square < x) {

ans = mid; // 记录当前可能的答案

left = mid + 1; // 去右边找更大的

} else {

right = mid - 1; // 去左边找

}

}

return ans;

}

};

**思路：**

基本思想是在区间 [1, x] 上进行二分查找，每次取中间值 mid，并比较 mid 和 x/mid 的大小关系，如果 mid 小于等于 x/mid，则说明 mid 是可能的结果，更新 ans，并在右半部分继续查找；如果 mid 大于 x/mid，则在左半部分继续查找。直到找到最后一个满足条件的 mid，即为所求的平方根。

**代码：**

class Solution {

public:

int mySqrt(int x) {

if (x == 0) return 0;

int left = 1, right = x, ans = 0;

while (left <= right) {

int mid = left + (right - left) / 2;

if (mid <= x / mid) {

ans = mid;

left = mid + 1;

} else {

right = mid - 1;

}

}

return ans;

}

};

或：

class Solution {

public:

int mySqrt(int x) {

int low=0,hight=x,ret=-1;

while(low<=hight)

{

int mid = low + (hight-low)/2;

if((long long)mid\*mid<=x)

{

ret = mid;

low = mid+1;

}

else

{

hight = mid-1;

}

}

return ret;

}

};

**复杂度：**

时间复杂度：O(logx)，即为二分查找需要的次数。

空间复杂度：O(1)。