# 第三次算法设计作业

姓名： 学号：

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码（请在关键代码旁加上注释）以及按示例输入得到的输出截图，设计算法时需要考虑边界约束。作业提交时请只保留选做的题，删除空白题。

【题一】 **盛最多水的容器**

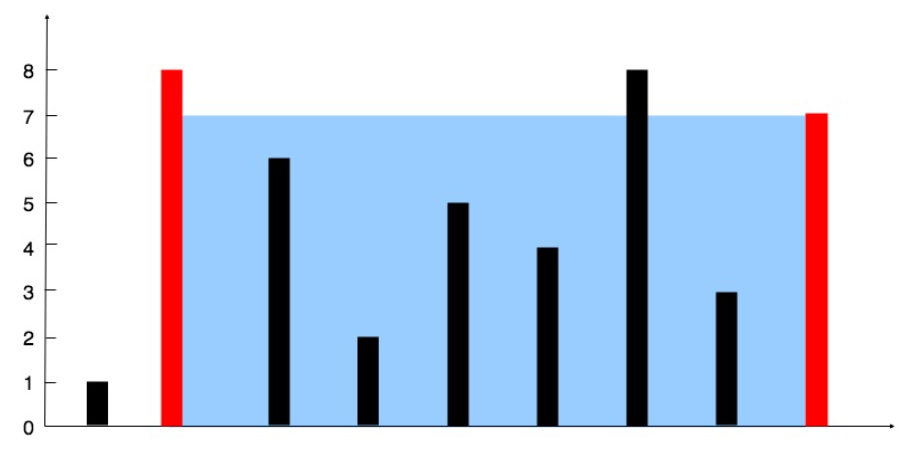
给定一个长度为n的整数数组height。有n条垂线，第i条线的两个端点是(i, 0)和(i, height[i])。

找出其中的两条线，使得它们与x轴共同构成的容器可以容纳最多的水。

返回容器可以储存的最大水量。

**说明**：你不能倾斜容器。

【示例1】



输入：height = [1,8,6,2,5,4,8,3,7]

输出：49

解释：图中垂直线代表输入数组 [1,8,6,2,5,4,8,3,7]。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为49。

【示例2】

输入：height = [1,1]

输出：1

【约束】

n == height.length

2 <= n <= 105

0 <= height[i] <= 104

【解题思路】

首先对输入的数据进行处理，存到一个名为height的vector数组中。随后初始化两个指针left和right。=，用于分别指向height的起始位置和终止位置。同时创建一个int类型的变量maxSize，用于记录当前运算所计算出的最多可盛水量。

又由于“木桶效应”，因此借助min(height[left], height[right])函数求出低的那个边，进而计算出当前位置的两个板子之间可以存储多少水current。如果current大于maxSize则更新maxSize。在更新完maxSize后可以用过移动两个指针的方式来寻找下一个可能的maxSize。遵循的原则是：如果左边的柱子较短，则left++，因为移动较短的柱子可能会找到更高的柱子，从而增加水量；反之则right--。当 left >= right 时，循环结束，此时已经遍历完所有可能的柱子组合。

由于遍历了一遍数组。所以时间复杂度是O(n)。

【代码】

#include <iostream>

#include <vector>

#include <sstream>

#include <algorithm>

using namespace std;

int main()

{

vector<int> height(100000);

string str;

getline(cin, str);

char t = '[';

int i = 1;

int j = 0;

int num = 0;

while (true)

{

if ('0' <= str[i] && str[i] <= '9')

{

num = num \* 10 + str[i] - '0';

}

else if (str[i] == ']')

{

height[j] = num;

num = 0;

break;

}

else

{

height[j] = num;

num = 0;

j++;

i++;

}

i++;

}

int n = height.size();

int left = 0;

int right = n - 1;

int maxSize = 0;

while (left < right)

{

int current = min(height[left], height[right]) \* (right - left);

maxSize = max(current, maxSize);

if (height[left] < height[right])

{

left++;

}

else

{

right--;

}

}

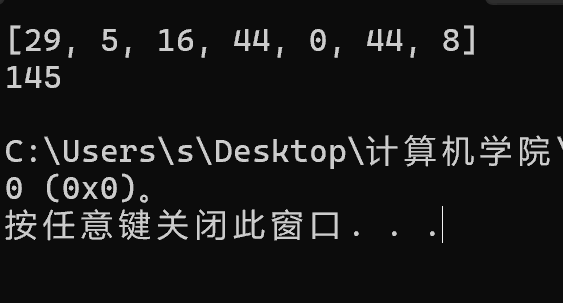
cout << maxSize << endl;

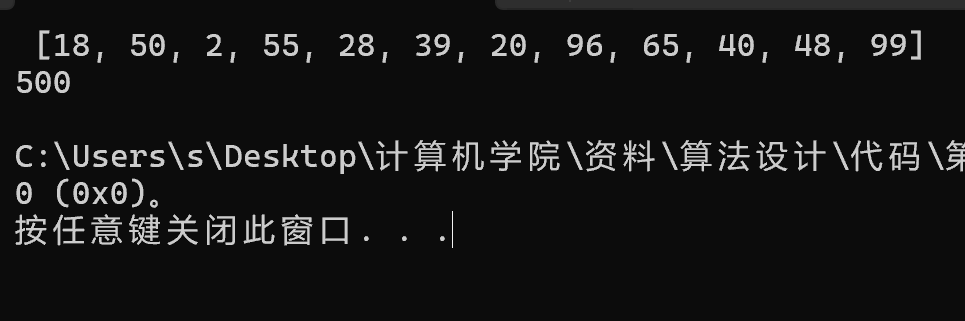
return 0;

}

【输出截图】1、输入：height = [29, 5, 16, 44, 0, 44, 8]

2、输入：height = [18, 50, 2, 55, 28, 39, 20, 96, 65, 40, 48, 99]





【题二】 **划分字母区间**  
给你一个字符串s。我们要把这个字符串划分为尽可能多的片段，同一字母最多出现在一个片段中。例如，字符串"ababcc"能够被分为["abab", "cc"]，但类似["aba", "bcc"]或["ab", "ab", "cc"]的划分是非法的。

注意，划分结果需要满足：将所有划分结果按顺序连接，得到的字符串仍然是s。

返回一个表示每个字符串片段的长度的列表。

【示例1】

输入：s = "ababcbacadefegdehijhklij"

输出：[9,7,8]

解释：划分结果为 "ababcbaca"、"defegde"、"hijhklij"。

每个字母最多出现在一个片段中。

像 "ababcbacadefegde", "hijhklij" 这样的划分是错误的，因为划分的片段数较少。

【示例2】

输入：s = "eccbbbbdec"

输出：[10]

【提示】

1 <= s.length <= 500

s仅由小写英文字母组成

【解题思路】

首先将需要分割的字符串输入到str字符串中。随后创建一个数组arr[26]并借助arr[str[i] - 'a'] = i + 1;语句用来存储每个字母最后出现的位置。在这里需要遍历一次字符串。随后使用 start 和 end 两个指针表示当前片段的范围。遍历字符串时，更新 end 为当前字符的最后出现位置。当遍历到 end 时，说明当前片段结束，记录片段长度，并更新 start 为下一个片段的起始位置。分割位置的结果都保存在result中。

因为遍历了两次str，故时间复杂度是O(n)。

【代码】

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

string str;

cin >> str;

int arr[26];

for (int i = 0; i < str.size(); i++)

{

arr[str[i] - 'a'] = i + 1;

}

int j = 0;

int start = 0, end = 0;

int result[500];

for (int i = 0; i < str.size(); i++)

{

end = max(end, arr[str[i] - 'a']);

if (i + 1 == end)

{

result[j] = end - start;

start = i + 1;

j++;

}

}

cout << "[";

int i = 0;

for (i = 0; i < j - 1; i++)

{

cout << result[i] << ",";

}

cout << result[i];

cout << "]";

return 0;

}

【输出截图】1、s="aabbaabbaazcdcdcdcdghgkhghq"

1. s="abcdbcdbwxyxwzzhijkkpqrspqrsvtuv"

