课堂代码

二分：

```
/*************************************************************************
  > File Name: 1.binary_search.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

void output_binary_process(int *arr, int n, int head, int tail, int mid) {
    int p1, p2, p3, len = 0;
    for (int i = 0; i < n; i++) {
        len += printf("%5d", i);
        if (i == head) p1 = len - 1;
        if (i == tail) p2 = len - 1;
        if (i == mid) p3 = len - 1;
    }
    printf("\n");
    for (int i = 0; i < len; i++) printf("-");
    printf("\n");
    for (int i = 0; i < n; i++) {
        printf("%5d", arr[i]);
    }
    printf("\n");
    for (int i = 0; i < len; i++) {
        if (i == p1 || i == p2 || i == p3) {
            printf("^");
        } else {
            printf(" ");
        }
    }
    printf("\n");
    for (int i = 0; i < len; i++) {
        if (i == p1 || i == p2 || i == p3) {
            printf("|");
        } else {
            printf(" ");
        }
    }
```

```c
    printf("\n\n");
    return ;
}

int binary_search(int *arr, int n, int x) {
    int head = 0, tail = n - 1, mid;
    while (head <= tail) {
        mid = (head + tail) >> 1;
        output_binary_process(arr, n, head, tail, mid);
        if (arr[mid] == x) return mid;
        if (arr[mid] < x) head = mid + 1;
        else tail = mid - 1;
    }
    return -1;
}

int *getRandData(int n) {
    int *arr = (int *)malloc(sizeof(int) * n);
    arr[0] = rand() % 10;
    for (int i = 1; i < n; i++) {
        arr[i] = arr[i - 1] + rand() % 5 + 1;
    }
    return arr;
}

void output(int *arr, int n) {
    int len = 0;
    for (int i = 0; i < n; i++) {
        len += printf("%5d", i);
    }
    printf("\n");
    for (int i = 0; i < len; i++) printf("-");
    printf("\n");
    for (int i = 0; i < n; i++) {
        printf("%5d", arr[i]);
    }
    printf("\n");
    return ;
}

int main() {
    srand(time(0));
    int n, x;
    scanf("%d", &n);
    int *arr = getRandData(n);
    output(arr, n);
    while (~scanf("%d", &x)) {
        printf("arr[%d] = %d\n",
            binary_search(arr, 10, x),
            x
        );
    }
    return 0;
}
```

搜索插入位置 (普通二分)

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int head = 0, tail = nums.size(), mid;
        while (head < tail) {
            mid = (head + tail) >> 1;
            if (nums[mid] < target) head = mid + 1;
            else tail = mid;
        }
        return head;
    }
};
```

在排序数组中查找元素的第一个和最后一个位置 (求重复区间)

```
class Solution {
public:
    int last_position(vector<int> &nums, int pos) {
        int head = pos, tail = nums.size() - 1, mid, x = nums[pos];
        while (head < tail) {
            mid = (head + tail + 1) >> 1;
            if (nums[mid] > x) tail = mid - 1;
            else head = mid;
        }
        return head;
    }
    int first_position(vector<int> &nums, int target) {
        if (nums.size() == 0) return -1;
        int head = 0, tail = nums.size() - 1, mid;
        while (head < tail) {
            mid = (head + tail) >> 1;
            if (nums[mid] >= target) tail = mid;
            else head = mid + 1;
        }
        if (nums[head] != target) return -1;
        return head;
    }
    vector<int> searchRange(vector<int>& nums, int target) {
        vector<int> ret(2);
        ret[0] = first_position(nums, target);
        if (ret[0] == -1) {
            ret[1] = -1;
            return ret;
        }
        ret[1] = last_position(nums, ret[0]);
        return ret;
    }
};
```

两数之和

```
class Solution {
public:
    int binary_search(vector<int>& nums, vector<int> &ind, int i, int x) {
        int head = i, tail = nums.size() - 1, mid;
        while (head <= tail) {
            mid = (head + tail) >> 1;
            if (nums[ind[mid]] == x) return mid;
            if (nums[ind[mid]] < x) head = mid + 1;
            else tail = mid - 1;
        }
        return -1;
    }
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> ind(nums.size());
        vector<int> ret(2);
        for (int i = 0; i < nums.size(); i++) ind[i] = i;
        sort(ind.begin(), ind.end(), [&](int i, int j) { return nums[i] < nums[j];});
        for (int i = 0; i < ind.size(); i++) {
            int j = binary_search(nums, ind, i + 1, target - nums[ind[i]]);
            if (j == -1) continue;
            ret[0] = ind[i];
            ret[1] = ind[j];
            if (ret[0] > ret[1]) swap(ret[0], ret[1]);
        }
        return ret;
    }
};
```

x 的平方根 (二分简单应用)

```
class Solution {
public:
    int mySqrt(int x) {
        int l = 0, r = x, mid;
        while (l < r) {
            mid = (r - l) / 2 + l + 1;
            if (mid <= x / mid) l = mid;
            else r = mid - 1;
        }
        return l;
    }
};
```

将 x 减到 0 的最小操作数 (二分前缀和)

```
class Solution {
public:
    int binary_search(vector<int>& nums, int x) {
        int l = 0, r = nums.size() - 1, mid;
        while (l <= r) {
            mid = (l + r) >> 1;
            if (nums[mid] == x) return mid;
            if (nums[mid] < x) l = mid + 1;
            else r = mid - 1;
        }
        return -1;
    }
    int minOperations(vector<int>& nums, int x) {
        vector<int> sum(nums.size() + 1);
        sum[0] = 0;
        for (int i = 1; i < sum.size(); i++) sum[i] = sum[i - 1] + nums[i - 1];
        int ans = -1;
        for (int i = nums.size(), s = 0, j; i >= 0; --i, s += nums[max(i, 0)]) {
            j = binary_search(sum, x - s);
            if (j == -1) continue;
            int cnt = nums.size() - i + j;
            if (cnt > nums.size()) continue;
            if (ans == -1 || nums.size() - i + j < ans) ans = nums.size() - i + j;
        }
        return ans;
    }
};
```

搜索旋转排序数组 II (二分里l,r的含义)

```
class Solution {
public:
    bool search(vector<int>& nums, int target) {
        if (nums.size() == 0) return false;
        if (target == nums[0]) return true;
        int mid, l = 0, r = nums.size() - 1, head, tail;
        while (l < r && nums[l] == nums[0]) ++l;
        while (l < r && nums[r] == nums[0]) --r;
        head = l, tail = r;
        while (l <= r) {
            mid = (l + r) >> 1;
            if (nums[mid] == target) return true;
            if (nums[mid] <= nums[tail]) {
                if (target > nums[mid] && target <= nums[tail]) l = mid + 1;
                else r = mid - 1;
            } else {
                if (target < nums[mid] && target >= nums[head]) r = mid - 1;
                else l = mid + 1;
            }
        }
```

```
            return false;
        }
};
```

供暖器（二分里l,r的含义)

```
class Solution {
public:
    int binary_search(vector<int>& nums, int x) {
        int l = 0, r = nums.size() - 1, mid;
        while (l < r) {
            mid = (l + r) >> 1;
            if (nums[mid] >= x) r = mid;
            else l = mid + 1;
        }
        return l;
    }
    int findRadius(vector<int>& houses, vector<int>& heaters) {
        sort(houses.begin(), houses.end());
        sort(heaters.begin(), heaters.end());
        int ans = 0;
        for (auto x : houses) {
            int ind = binary_search(heaters, x);
            int a = abs(heaters[ind] - x);
            int b = abs(ind ? x - heaters[ind - 1] : a + 1);
            ans = max(ans, min(a, b));
        }
        return ans;
    }
};
```

最长递增子序列

```
class Solution {
public:
    int binary_search(vector<int>& nums, int n, int x) {
        int l = 1, r = n + 1, mid;
        while (l < r) {
            mid = (l + r) >> 1;
            if (nums[mid] >= x) r = mid;
            else l = mid + 1;
        }
        return l;
    }
    int lengthOfLIS(vector<int>& nums) {
        vector<int> len(nums.size() + 1);
        len[1] = nums[0];
        int ans = 1;
        for (int i = 1; i < nums.size(); i++) {
            int l = binary_search(len, ans, nums[i]);
            len[l] = nums[i];
            ans = max(ans, l);
        }
        return ans;
    }
};
```

在 D 天内送达包裹的能力(二分答案)

```
class Solution {
public:
    int check(vector<int>& nums, int x) {
        int cnt = 1, sum = 0;
        for (auto y : nums) {
            if (sum + y > x) {
                cnt += 1;
                sum = y;
            } else {
                sum += y;
            }
        }
        return cnt;
    }
    int shipWithinDays(vector<int>& weights, int days) {
        int l = 0, r = 0, mid;
        for (auto x : weights) r += x, l = max(l, x);
        while (l < r) {
            mid = (l + r) >> 1;
            if (check(weights, mid) <= days) r = mid;
            else l = mid + 1;
        }
        return l;
    }
};
```

寻找两个正序数组的中位数 (经典二分)

```
class Solution {
public:
    double binaryFindMedian(vector<int> &num1, vector<int> &num2, int k1, int k2, int k) {
        if (k1 == num1.size()) return num2[k2 + k - 1];
        if (k2 == num2.size()) return num1[k1 + k - 1];
        if (k == 1) return num1[k1] < num2[k2] ? num1[k1] : num2[k2];
        int a = min(k / 2, int(num1.size() - k1));
        int b = min(k - a, int(num2.size() - k2));
        a = k - b;
        if (num1[k1 + a - 1] < num2[k2 + b - 1]) {
            return binaryFindMedian(num1, num2, k1 + a, k2, k - a);
        } else {
            return binaryFindMedian(num1, num2, k1, k2 + b, k - b);
        }
        return 0;
    }
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int n = nums1.size() + nums2.size();
        double a = binaryFindMedian(nums1, nums2, 0, 0, (n + 1) / 2);
        if (n % 2) return a;
        double b = binaryFindMedian(nums1, nums2, 0, 0, (n + 1) / 2 + 1);
        return (a + b) / 2;
    }
};
```