参考代码

0. 课上实现代码

```
#include <cstdio>
#include <cstdlib>

using namespace std;

#define low16(a) ((a) & 0xffff)
#define __high16(a) (((a) & 0xffff0000) >> 16)
#define high16(a) (__high16(a) > 32767 ? (__high16(a) - 32768) : (__high16(a) + 32768))

void radix_sort(int *arr, int n) {
    int cnt[65536] = {0};
    int *temp = (int *) malloc(sizeof(int) * n);
    // low 16 bit sort
    for (int i = 0; i < n; i++) cnt[low16(arr[i])] += 1; // count
    for (int i = 1; i < 65536; i++) cnt[i] += cnt[i - 1];  // prefix sum
    for (int i = n - 1; i >= 0; --i) temp[--cnt[low16(arr[i])]] = arr[i]; // placement
    // init cnt
    for (int i = 0; i < 65536; i++) cnt[i] = 0;
    // high 16 bit sort
    for (int i = 0; i < n; i++) cnt[high16(temp[i])] += 1; // count
    for (int i = 1; i < 65536; i++) cnt[i] += cnt[i - 1];  // prefix sum
    for (int i = n - 1; i >= 0; --i) arr[--cnt[high16(temp[i])]] = temp[i];
    free(temp);
    return;
}

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

int *getRandData(int n) {
    int *temp = (int *) malloc(sizeof(int) * n);
    for (int i = 0; i < n; i++) temp[i] = (rand() % 2 ? -1 : 1) * (rand());
    return temp;
}

int main() {
#define MAX_N 20
    int *arr = getRandData(MAX_N);
    output(arr, MAX_N);
    radix_sort(arr, MAX_N);
    output(arr, MAX_N);
    free(arr);
    return 0;
}
```

1. 课程表

```
class Solution {
public:
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
        vector<int> indeg(numCourses);
        vector<vector<int>> g(numCourses);
        queue<int> q;
        for (auto x : prerequisites) {
            indeg[x[1]] += 1;
            g[x[0]].push_back(x[1]);
        }
        for (int i = 0; i < numCourses; i++) {
            if (indeg[i] == 0) q.push(i);
        }
        int ans = 0;
        while (!q.empty()) {
            ans += 1;
            int ind = q.front();
            q.pop();
            for (auto to : g[ind]) {
                indeg[to] -= 1;
                if (indeg[to] == 0) q.push(to);
            }
        }
        return ans == numCourses;
    }
};
```

2. 课程表 II

```
class Solution {
public:
    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
        vector<int> indeg(numCourses);
        vector<vector<int>> g(numCourses);
        queue<int> q;
        vector<int> ans;
        for (auto x : prerequisites) {
            indeg[x[0]] += 1;
            g[x[1]].push_back(x[0]);
        }
        for (int i = 0; i < numCourses; i++) {
            if (indeg[i] == 0) q.push(i);
        }
        while (!q.empty()) {
            int ind = q.front();
            ans.push_back(ind);
            q.pop();
            for (auto to : g[ind]) {
                indeg[to] -= 1;
                if (indeg[to] == 0) q.push(to);
            }
        }
        if (ans.size() - numCourses) ans.clear();
        return ans;
    }
};
```

3. 数组的相对排序

```
class Solution {
public:
    vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {
        int cnt[1005] = {0};
        for (auto x : arr1) cnt[x] += 1;
        int k = 0;
        for (auto x : arr2) {
            while (cnt[x]--) arr1[k++] = x;
        }
        for (int i = 0; i <= 1000; i++) {
            if (cnt[i] <= 0) continue;
            while (cnt[i]--) arr1[k++] = i;
        }
        return arr1;
    }
};
```

4. 最大间距

```
class Solution {
public:
    int maximumGap(vector<int>& nums) {
        vector<int> temp(nums.size());
        int cnt[65536] = {0};
        for (auto x : nums) cnt[x & 0xffff] += 1;
        for (int i = 1; i < 65536; i++) cnt[i] += cnt[i - 1];
```

```cpp
        for (int i = nums.size() - 1; i >= 0; i--) {
            temp[--cnt[nums[i] & 0xffff]] = nums[i];
        }
        memset(cnt, 0, sizeof(cnt));
        for (auto x : temp) cnt[(x & 0xffff0000) >> 16] += 1;
        for (int i = 1; i < 65536; i++) cnt[i] += cnt[i - 1];
        for (int i = nums.size() - 1; i >= 0; i--) {
            nums[--cnt[(temp[i] & 0xffff0000) >> 16]] = temp[i];
        }
        int ans = 0;
        for (int i =  1; i < nums.size(); i++) {
            ans = max(ans, nums[i] - nums[i - 1]);
        }
        return ans;
    }
};
```

5. H 指数

```cpp
class Solution {
public:
    int hIndex(vector<int>& citations) {
        sort(citations.begin(), citations.end());
        int i;
        for (i = citations.size() - 1; i > 0 && citations[i - 1] >= citations.size() - i + 1; i--) ;
        if (citations[i] < citations.size() - i) return 0;
        return citations.size() - i;
    }
};
```

6. 合并区间

```cpp
class Solution {
public:
    struct Data {
        Data(int pos, int c) : pos(pos), c(c) {}
        bool operator<(const Data &a) {
            if (pos - a.pos) return pos < a.pos;
            return c > a.c;
        }
        int pos, c;
    };
    vector<vector<int>> merge(vector<vector<int>>& intervals) {
        vector<Data> arr;
        for (auto x : intervals) {
            arr.push_back(Data{x[0], 1});
            arr.push_back(Data{x[1], -1});
        }
        vector<vector<int>> ret;
        sort(arr.begin(), arr.end());
        for (int i = 0, pre = -1, cnt = 0; i < arr.size(); i++) {
            if (pre == -1) pre = arr[i].pos;
            cnt += arr[i].c;
            if (cnt == 0) {
                vector<int> temp(2);
                temp[0] = pre;
                temp[1] = arr[i].pos;
                ret.push_back(temp);
                pre = -1;
            }
        }
        return ret;
    }
};
```

7. 删除被覆盖区间

```
class Solution {
public:
    int removeCoveredIntervals(vector<vector<int>>& intervals) {
        sort(intervals.begin(), intervals.end(),
            [](const vector<int> &a, const vector<int> &b) -> bool {
                if (a[0] - b[0]) return a[0] < b[0];
                return a[1] > b[1];
            }
        );
        int cnt = 0, pre = -1;
        for (auto x : intervals) {
            if (pre >= x[1]) cnt += 1;
            pre = max(x[1], pre);
        }
        return intervals.size() - cnt;
    }
};
```

8. 递增子序列

```
class Solution {
public:
    void getResult(vector<int> &nums, int ind, int k, vector<int> buff, vector<vector<int>> &ret) {
        if (buff.size() > 1) ret.push_back(buff);
        if (ind == nums.size()) return ;
        buff.push_back(0);
        unordered_map<int, int> can;
        for (int i = ind; i < nums.size(); i++) {
            if (k == 0 || buff[k - 1] <= nums[i]) {
                if (can.find(nums[i]) != can.end()) continue;
                can[nums[i]] = 1;
                buff[k] = nums[i];
                getResult(nums, i + 1, k + 1, buff, ret);
            }
        }
        return ;
    }
    vector<vector<int>> findSubsequences(vector<int>& nums) {
        vector<vector<int>> ret;
        getResult(nums, 0, 0, vector<int>(), ret);
        return ret;
    }
};
```

9. 求和路径

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    int pathSumContinuation(TreeNode *root, int sum) {
        if (root == NULL) return 0;
        sum -= root->val;
        return (sum == 0) + pathSumContinuation(root->left, sum) + pathSumContinuation(root->right, sum);
    }
    int pathSum(TreeNode* root, int sum) {
        if (root == NULL) return 0;
        int a = pathSum(root->left, sum);
        int b = pathSum(root->right, sum);
        return a + b + pathSumContinuation(root, sum);
    }
};
```