September 11, 2018 题目 302,415,616,42,239,371,287,817,119,776

302

1. binary search

```
class Solution {
    public int minArea(char[][] image, int x, int y) {
        int m = image.length, n = image[0].length;
        int left = boundaryVertical(image, 0, x, 0, n, true);
        int right = boundaryVertical(image, x + 1, m, 0, n, fa
lse);
        int top = boundaryHorizontal(image, 0, y, left, right,
true);
        int bottom = boundaryHorizontal(image, y + 1, n, left,
right, false);
        return (bottom - top) * (right - left);
    }

    private int boundaryHorizontal(char[][] image, int i, int
j, int top, int bottom, boolean reverse) {
        while (i != j) {
            int mid = i + (j - i) / 2;
            int k = top;
            while (k < bottom && image[k][mid] == '0') {
                k++;
            }

            if (k < bottom == reverse) {
                j = mid;
```

```
            } else {
                i = mid + 1;
            }
        }
        return i;
    }

    private int boundaryVertical(char[][] image, int i, int j,
int left, int right, boolean reverse) {
        while (i != j) {
            int mid = i + (j - i) / 2;
            int k = left;
            while (k < right && image[mid][k] == '0') {
                k++;
            }
            if (k < right == reverse) {
                j = mid;
            } else {
                i = mid + 1;
            }
        }
        return i;
    }
}
```

2. 注意是Char 不是 int :

```
class Solution {
public:
    int minArea(vector<vector<char>>& img, int x, int y) {
        int n = img.size(), m = img[0].size();
        int l = -1, r = n, u = -1, d = m, z = x, w = y;
```

```
        while(z<r-1){
            int c = (z+r)/2;
            bool ok = false;
            for(int j=0; j<m && !ok; ++j) if(img[c][j] == '1')
ok = true;
            if(ok) z = c;
            else r = c;
        }
        z = x;
        while(l < z-1){
            int c = (z+l)/2;
            bool ok = false;
            for(int j=0; j<m && !ok; ++j) if(img[c][j] == '1')
ok = true;
            if(ok) z = c;
            else l = c;
        }
        ++l;
        while(w < d-1){
            int c = (d+w)/2;
            bool ok = false;
            for(int i=0; i<n && !ok; ++i) if(img[i][c] == '1')
ok = true;
            if(ok) w = c;
            else d = c;
        }
        w = y;
        while(u < w-1){
            int c = (u+w)/2;
            bool ok = false;
```

```
            for(int i=0; i<n && !ok; ++i) if(img[i][c] == '1')
ok = true;
            if(ok) w = c;
            else u = c;
        }
        ++u;
        return (r-l) * (d-u);
    }
};
```

415

1. 先Reverse 再加：

```
class Solution {
    #define CI(c) int((c) - '0')
    #define IC(i) char((i) + '0')
public:
    string addStrings(string num1, string num2) {
        reverse(num1.begin(), num1.end());
        reverse(num2.begin(), num2.end());
        string res;
        for(int i = 0, cur = 0; i<num1.size() || i<num2.size()
|| cur; ++i){
            if(i<num1.size()) cur += CI(num1[i]);
            if(i<num2.size()) cur += CI(num2[i]);
            res.push_back(IC(cur%10));
            cur /= 10;
        }
        while(res.back() == '0') res.pop_back();
        if(res.empty()) res = "0";
        reverse(res.begin(), res.end());
        return res;
```

```
    }
};
```

616

1. KMP:

```
class Solution {
public:
    string addBoldTag(string s, vector<string>& dict) {
        int N = s.size();
        vector<int> mark(N, 0);
        for(string t: dict){
            int n = t.size(), i = 0, j = 0;
            vector<int> dp(n, -1);
            for(int k=1; k<n; ++k){
                int l = dp[k-1];
                while(l>=0 && t[l+1] != t[k]) l = dp[l];
                if(t[l+1] == t[k]) dp[k] = l+1;
            }
            while(i<N){
                while(i<N && j<n && s[i]==t[j]){
                    ++i;
                    ++j;
                }
                if(j==n){
                    ++mark[i-n];
                    if(i<N){
                        --mark[i];
                    }
                    j = dp[n-1] + 1;
                }
```

```
                else{
                    while(j > 0 && t[j] != s[i]) j = dp[j-1] +
1;
                    if(t[j] != s[i]) ++i;
                }
            }
        }
        for(int tmp=0, i=0; i<N; ++i){
            tmp += mark[i];
            mark[i] = tmp;
        }
        for(int k: mark) cout<<k;
        string ans;
        for(int i=0; i<N; ++i){
            if(mark[i] && (!i || !mark[i-1])) ans += "<b>";
            ans += s[i];
            if(mark[i] && (i==N-1 || !mark[i+1])) ans += "</b
>";
        }
        return ans;
    }
};
```

42

1. Frontier :

```
class Solution {
public:
    int trap(vector<int>& height) {
        if(height.empty()) return 0;
        int n = height.size(), ans = 0;
```

```
        for(int l=0, r=n-1, lh=height[0], rh=height[n-1]; l<r-1; ){
            if(lh < rh){
                ++l;
                ans += max(0, lh - height[l]);
                lh = max(lh, height[l]);
            }
            else{
                --r;
                ans += max(0, rh - height[r]);
                rh = max(rh, height[r]);
            }
        }
        return ans;
    }
};
```

239

1. 一个deque就搞定了：

```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        vector<int> ans;
        deque<int> Q;
        for(int i=0; i<nums.size(); ++i){
            while(!Q.empty() && nums[i]>=nums[Q.back()]) Q.pop_back();
            Q.push_back(i);
            if(Q.front()<=i-k) Q.pop_front();
            if(i>=k-1) ans.push_back(nums[Q.front()]);
        }
```

```
        return ans;
    }
};
```

371

1. bit

```
class Solution {
    public int getSum(int a, int b) {
        // 1 2 => 0001 0010 => 0011
        // 2 2 => 0010 0010 => 0100
        if (a == 0) return b;
        if (b == 0) return a; // 0010
        return sum(a, b);
    }
    public int sum(int a, int b) {
        while (b != 0) {
            int carry = a & b; // 0010 0000
            a = a ^ b; // 0000 0100
            b = carry << 1; // 0100
        }
        return a;
    }
    public int getSubtract(int a, int b) {
        // a - b => 7 - 2 => 0111 - 0010 = 0101
        while (b != 0) {
            int borrow = ~a & b; //1000 0010 => 0000 0101&0001
=>0001
            a = a ^ b; // 1010 1010^0001=1011
            b = borrow << 1; // 0001
        }
        return a;
```

```
    }
}
```

2. Short is Beauty:

```
class Solution {
public:
    int getSum(int a, int b) {
        return a + b;
    }
};
```

3. Bit,昨天直接懵B了，以为很难:

```
class Solution {
public:
    int getSum(int a, int b) {
        int ans = 0;
        for(int j=0, cur=0; j<32; ++j) {
            int x = 1&(a>>j), y = 1&(b>>j);
            ans |= (x ^ y ^ cur)<<j;
            cur = (x&y)|(y&cur)|(cur&x);
        }
        return ans;
    }
};
```

193

BASH

287

1. 直接sort就行了：

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
```

```
        sort(nums.begin(), nums.end());
        for(int i=1; i<nums.size(); ++i) if(nums[i] == nums[i-1]) return nums[i];
    }
};
```

2. 这个解法太恐怖了 @Tongtong X

- [x] https://leetcode.com/problems/find-the-duplicate-number/discuss/72846/My-easy-understood-solution-with-O(n)-time-and-O(1)-space-without-modifying-the-array.-With-clear-explanation
- [ ] @Zebo L 说实话我不太明白这个解法为啥可以...但是又好像有道理

817

1. One pass:

```
class Solution {
public:
    int numComponents(ListNode* head, vector<int>& G) {
        unordered_set<int> S(G.begin(), G.end());
        int ans = 0;
        ListNode lead(0);
        lead.next = head;
        for(auto p=&lead; p->next; p=p->next){
            if(S.count(p->next->val) && (p==&lead || !S.count(p->val))) ++ans;
        }
        return ans;
    }
};
```

119

1. 折半optimization：

```
class Solution {
public:
```

```
    vector<int> getRow(int r) {
        if(r <= 1) return vector<int>(r+1, 1);
        vector<int> res(2, 1);
        for(int i=2; i<=r; ++i){
            vector<int> tmp(i+1, 1);
            for(int j=1; j<=i/2; ++j) tmp[j] = tmp[i-j] = res
[j-1] + res[j];
            swap(tmp, res);
        }
        return res;
    }
};
```

776

1. 一路向下，依次append即可：

```
class Solution {
public:
    vector<TreeNode*> splitBST(TreeNode* root, int V) {
        vector<TreeNode *> res(2, NULL);
        TreeNode *resl = NULL, *resr = NULL;
        while(root){
            if(root->val > V){
                if(!resr) res[1] = resr = root;
                else{
                    resr->left = root;
                    resr = root;
                }
                root = root->left;
            }
            else{
                if(!resl) res[0] = resl = root;
```

```
                else{
                    resl->right = root;
                    resl = root;
                }
                root = root->right;
            }
        }
        if(resl) resl->right = NULL;
        if(resr) resr->left = NULL;
        return res;
    }
};
```