September 14, 2018 题目：604,368,418,341,747,69,92,138,87,819,762

604

1.

```
class StringIterator {
    String compressedString;
    int count;
    int index;
    char cur;
    public StringIterator(String compressedString) {
        index = 0;
        this.compressedString = compressedString;
        cur = compressedString.charAt(index++);
        count = getCount();
    }

    public char next() {
        if (hasNext()) {
            if (count == 0) {
                cur = compressedString.charAt(index++);
                count = getCount();
            }
            count--;
            return cur;
        }
        return ' ';
    }
    public boolean hasNext() {
        return !(index == compressedString.length() && count =
= 0);
    }

    private int getCount() {
        int res = 0;
        while (index < compressedString.length() && Character.
isDigit(compressedString.charAt(index))) {
            res = res * 10 + (compressedString.charAt(index) -
'0');
            index++;
        }
        return res == 0 ? 1 : res;
    }
}
```

2. 面试的时候遇到这种题不能慌：

```
class StringIterator {
    string s;
    int i, cnt;
public:
    StringIterator(string compressedString): s(compressedStrin
g), i(0), cnt(stoi(compressedString.substr(1))) {}
    char next() {
        if(i<s.size()){
            char c = s[i];
            --cnt;
            if(!cnt){
                ++i;
                while(i<s.size() && isdigit(s[i])) ++i;
                if(i<s.size()) cnt = stoi(s.substr(i+1));
            }
            return c;
        }
        return ' ';
    }
    bool hasNext() { return i<s.size(); }
};
```

368

1. DP

```
class Solution {
    public List<Integer> largestDivisibleSubset(int[] nums) {
        List<Integer> res = new ArrayList<>();
        if (nums == null || nums.length == 0) {
            return res;
        }
        Arrays.sort(nums);
        int[] dp = new int[nums.length];
        dp[0] = 1;
        for (int i = 1; i < nums.length; i++) {
            for (int j = i - 1; j >= 0; j--) {
                if (nums[i] % nums[j] == 0) {
                    dp[i] = Math.max(dp[i], dp[j] + 1);
                }
            }
        }

        int max = 0;
        for (int i = 1; i < nums.length; i++) {
            max = dp[i] >= dp[max] ? i : max;
        }
        int tmp = nums[max];
        int index = dp[max];
        for (int i = max; i >= 0; i--) {
            if (tmp % nums[i] == 0 && index == dp[i]) {
                res.add(nums[i]);
                tmp = nums[i];
                index--;
            }
        }
        return res;
    }
}37
```

2. n方复杂度居然可以超92% (记住面试很有可能比这难的多，做好心里准备)

```
class Solution {
public:
    vector<int> largestDivisibleSubset(vector<int>& nums) {
        if(nums.empty()) return {};
        int n = nums.size(), idx = 0, res = 1;
        sort(nums.begin(), nums.end());
        vector<int> dp(n, 1), prev(n, -1), ans;
        for(int i=1; i<n; ++i){
            for(int j=0; j<i; ++j) if(nums[i] % nums[j] == 0){
                if(dp[j] + 1 > dp[i]){
                    dp[i] = dp[j] + 1;
                    prev[i] = j;
                }
            }
            if(dp[i] > res){
                idx = i;
                res = dp[i];
            }
        }
        while(idx >= 0){
            ans.push_back(nums[idx]);
            idx = prev[idx];
        }
        return ans;
    }
};
```

418

1. transfer idea :

```
class Solution {
    typedef pair<int, int> ii;
public:
    int wordsTyping(vector<string>& sentence, int rows, int co
ls) {
        int n = sentence.size();
        ++cols;
        vector<int> P(n+1, 0);
        for(int i=1; i<=n; ++i) P[i] = P[i-1] + 1 + sentence[i
-1].size();
        vector<ii> V(n, ii(0, -1));
        for(int i=0; i<n; ++i){
            int k = i, m = cols;
            if(m >= P[n] - P[k]){
                m -= P[n] - P[k];
                ++V[i].first;
                k = 0;
            }
            V[i].first += m / P[n];
            m %= P[n];
            int l = k, r = n;
            while(l < r-1){
                int c = (l + r)/2;
                if(P[c] - P[k]<=m) l = c;
                else r = c;
            }
            V[i].second = l;
        }
        int start = 0, ans = 0;
        for(int i=0; i<rows; ++i){
            ans += V[start].first;
            start = V[start].second;
        }
        return ans;
    }
};
```

341

1. Queue + recursion:

```
class NestedIterator {
    queue<int> Q;
public:
    NestedIterator(vector<NestedInteger> &nl) {
        stack<NestedInteger> S;
        for(int i=nl.size()-1; i>=0; --i) S.push(nl[i]);
        while(!S.empty()){
            auto r = S.top();
            S.pop();
            while(!r.isInteger() && !r.getList().empty()){
                auto vec = r.getList();
                for(int i=vec.size()-1; i; --i) S.push(vec
[i]);
                r = vec[0];
            }
            if(r.isInteger()) Q.push(r.getInteger());
        }
    }
    int next() {
        int ans = Q.front();
        Q.pop();
        return ans;
    }
    bool hasNext() { return !Q.empty(); }
};
```

747

1.

```
class Solution {
    public int dominantIndex(int[] nums) {
        int index = 0;
        int max = nums[0];
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] > max) {
                index = i;
                max = nums[i];
            }
        }
        for (int i = 0; i < nums.length; i++) {
            if (i != index) {
                if (nums[i] * 2 > max) {
                    return -1;
                }
            }
        }
        return index;
    }
}
```

2. 找最大数跟第二大数：

```
class Solution {
public:
    int dominantIndex(vector<int>& nums) {
        int a = nums[0], b = 0, idx = 0, n = nums.size();
        for(int j=1; j<n; ++j){
            if(nums[j] > a){
                b = a;
                a = nums[j];
                idx = j;
            }
            else if(nums[j] > b) b = nums[j];
        }
        return (a>=2*b? idx:-1);
    }
};
```

69

1. binary search

```
class Solution {
    public int mySqrt(int x) {
        if (x <= 1) {
            return x;
        }
        int lo = 1, hi = x / 2;
        while (true) {
            int mid = lo + (hi - lo) / 2;
            if (mid > x / mid) {
                hi = mid - 1;
            } else {
                if (mid + 1 > x / (mid + 1)) {
                    return mid;
                }
                lo = mid + 1;
            }
        }
    }
}
```

2. 去面试的时候别头脑发热，直接brute force了：

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

92

1. 要细心，细心，细心：

```
class Solution {
    ListNode *reverseRange(ListNode *head, int k){
        if(!head || !head->next) return head;
        ListNode *p=head, *p1=head->next;
        while(k && p1){
            ListNode *p2 = p1->next;
            p1->next = p;
            p = p1;
            p1 = p2;
            --k;
        }
        head->next = p1;
        return p;
    }
public:
    ListNode* reverseBetween(ListNode* head, int m, int n) {
        if(m==1) return reverseRange(head, n-m);
        ListNode *p = head;
        for(int i=1; i<m-1; ++i) p = p->next;
        p->next = reverseRange(p->next, n-m);
        return head;
    }
};
```

569

SQL

138

1. Inplace 居然需要三次loop：

```
class Solution {
public:
    RandomListNode *copyRandomList(RandomListNode *head) {
        if(!head) return NULL;
        RandomListNode pre(0);
        for(auto p=&pre, p1=head; p1; ){
            p->next = p1;
            auto tmp = p1->next;
            p1->next = new RandomListNode(p1->label);
            p  = p1->next;
            p1 = tmp;
        }
        for(auto p=pre.next; p; p=p->next->next){
            if(p->random) p->next->random = p->random->next;
        }
        auto head1 = pre.next->next;
        for(auto p=pre.next; p; p=p->next){
            auto q = p->next;
            p->next = q->next;
            if(p->next) q->next = p->next->next;
        }
        return head1;
    }
};
```

87

1. recursion

```
class Solution {
    public boolean isScramble(String s1, String s2) {
        if (s1.equals(s2)) {
            return true;
        }
        if (s1.length() != s2.length()) {
            return false;
        }
        int[] ch = new int[26];
        for (char c : s1.toCharArray()) {
            ch[c - 'a']++;
        }
        for (char c : s2.toCharArray()) {
            ch[c - 'a']--;
            if (ch[c - 'a'] < 0) {
                return false;
            }
        }

        for (int i = 1; i < s1.length(); i++) {
            if (isScramble(s1.substring(0, i), s2.substring(0, i)) && isScramble(s1.substring(i), s2.substring(i))) {
                return true;
            }
            if (isScramble(s1.substring(0, i), s2.substring(s2.length() - i)) && isScramble(s1.substring(i), s2.substring(0, s2.length() - i))) {
                return true;
            }
        }
        return false;
    }
}
```

2. 为啥加层memo才过，而且还速度那么慢：

```
class Solution {
    typedef vector<int> vi;
    vector<vector<vector<vi>>> dp;
    bool isScr(int i1, int j1, int i2, int j2, string&s1, string&s2){
        if(s1.substr(i1, j1-i1) == s2.substr(i2, j2-i2)) return true;
        if(dp[i1][j1][i2][j2] >= 0) return dp[i1][j1][i2][j2];
        for(int k=1; k<j1-i1; ++k){
            int t = j1 - i1 - k;
            if(isScr(i1, i1+k, i2, i2+k, s1, s2) && isScr(i1+k, j1, i2+k, j2, s1, s2)) return dp[i1][j1][i2][j2] = 1;
            if(isScr(i1, i1+k, i2+t, j2, s1, s2) && isScr(i1+k, j1, i2, i2+t, s1, s2)) return dp[i1][j1][i2][j2] = 1;
        }
        return dp[i1][j1][i2][j2] = 0;
    }
public:
    bool isScramble(string s1, string s2) {
        if(s1.size() != s2.size()) return false;
        int n = s1.size();
        dp = vector<vector<vector<vi>>>(n+1, vector<vector<vi>>(n+1, vector<vi>(n+1, vi(n+1, -1))));
        return isScr(0, n, 0, n, s1, s2);
    }
};
```

3. 多重剪枝之后还是这么慢：

```
class Solution {
    #define CI(c) int((c) - 'a')
    typedef vector<int> vi;
    vector<vector<vector<vi>>> dp;
    bool isScr(int i1, int j1, int i2, int j2, string&s1, string&s2){
        if(s1.substr(i1, j1-i1) == s2.substr(i2, j2-i2)) return true;
        if(dp[i1][j1][i2][j2] >= 0) return dp[i1][j1][i2][j2];
        vi l1(26, 0), r1(26, 0), l2(26, 0), r2(26, 0), l3(26, 0), r3(26, 0);
        for(int k=i1; k<j1; ++k) r1[CI(s1[k])]++;
        for(int k=i2; k<j2; ++k) {
            r2[CI(s2[k])]++;
            l3[CI(s2[k])]++;
        }
        for(int k=1; k<j1-i1; ++k){
            int t = j1 - i1 - k;
            l1[CI(s1[i1+k-1])]++;
            r1[CI(s1[i1+k-1])]--;
            l2[CI(s2[i2+k-1])]++;
            r2[CI(s2[i2+k-1])]--;
            l3[CI(s2[j2-k])]--;
            r3[CI(s2[j2-k])]++;
            if(l1==l2 && r1==r2 && isScr(i1, i1+k, i2, i2+k, s1, s2) && isScr(i1+k, j1, i2+k, j2, s1, s2)) return dp[i1][j1][i2][j2] = 1;
            if(l1==r3 && r1==l3 && isScr(i1, i1+k, i2+t, j2, s1, s2) && isScr(i1+k, j1, i2, i2+t, s1, s2)) return dp[i1][j1][i2][j2] = 1;
        }
        return dp[i1][j1][i2][j2] = 0;
    }
public:
    bool isScramble(string s1, string s2) {
        if(s1.size() != s2.size()) return false;
        int n = s1.size();
        vector<int> cnt(26, 0);
        dp = vector<vector<vector<vi>>>(n+1, vector<vector<vi>
>(n+1, vector<vi>(n+1, vi(n+1, -1))));
        return isScr(0, n, 0, n, s1, s2);
    }
};
```

4. 这个超时了：

```
class Solution {
    #define CI(c) int((c) - 'a')
    typedef vector<int> vi;
public:
    bool isScramble(string s1, string s2) {
        if(s1.size() != s2.size()) return false;
        if(s1 == s2) return true;
        int n = s1.size();
        vector<int> cnt(26, 0);
        for(char c: s1) ++cnt[CI(c)];
        for(char c: s2) {
            --cnt[CI(c)];
            if(CI(c) < 0) return false;
        }
        for(int l=1; l<n; ++l){
            int d = n-l;
            if(isScramble(s1.substr(0, l), s2.substr(0, l)) &&
isScramble(s1.substr(l), s2.substr(l))) return true;
            if(isScramble(s1.substr(0, l), s2.substr(d)) && is
Scramble(s1.substr(l), s2.substr(0, d))) return true;
        }
        return false;
    }
};
```

819 前两天做了

1. 前两天用python做的，但是c++做实在太麻烦了。。

762

1.

```
class Solution {
    public int countPrimeSetBits(int L, int R) {
        int res = 0;
        boolean[] dp = new boolean[33];
        dp[1] = true;
        for (int i = 2; i <= 32; i++) {
            for (int j = 2; i * j <= 32; j++) {
                dp[i * j] = true;
            }
        }

        for (int i = L; i <= R; i++) {
            if (!dp[bitCount(i)]) {
                res++;
            }
        }
        return res;
    }

    private int bitCount(int num) {
        int res = 0;
        while (num != 0) {
            num &= (num - 1);
            res++;
        }
        return res;
    }
}
```
2. 只能想到brute force method：
```
class Solution {
public:
    int countPrimeSetBits(int L, int R) {
        vector<bool> prime(32, false);
        for(int n: {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31}) p
rime[n] = true;
        int ans = 0;
        for(int k=L; k<=R; ++k){
            int cnt = 0;
            for(int j=0; j<22; ++j) if(1 & (k>>j)) ++cnt;
            if(prime[cnt]) ++ans;
        }
        return ans;
    }
};
```
- [ ] 神奇的方法： https://leetcode.com/problems/prime-number-of-set-bits-in-binary-representation/discuss/164223/Fast-algorithm-with-explanation-O(logN)-C++-4ms