August 25, 2018 题目：834,668,329,421,340,612,322,195,202,208,592,234,600,126,678

834

1. 这题不难，dfs + 步步推进即可，别被题吓住：

```
class Solution {
    vector<int> res, cnt;
    vector<vector<int>> E;
    int getCount(int from, int k, int level){
        res[0] += level;
        cnt[k] = 1;
        for(int j: E[k]) if(j!=from) cnt[k] += getCount(k, j, level+1);
        return cnt[k];
    }
    void dfs(int from, int k, const int&N){
        for(int j: E[k]) if(j!=from){
            res[j] = res[k] + N - 2 * cnt[j];
            dfs(k, j, N);
        }
    }
public:
    vector<int> sumOfDistancesInTree(int N, vector<vector<int>>& edges) {
        res.resize(N);
        cnt.resize(N);
        E.resize(N);
        for(auto e: edges){
```

```
            E[e[0]].push_back(e[1]);
            E[e[1]].push_back(e[0]);
        }
        getCount(-1, 0, 0);
        dfs(-1, 0, N);
        return res;
    }
};
```

668

1. 昨天刚做过，二分：

```
class Solution {
public:
    int findKthNumber(int m, int n, int k) {
        if(n>m) swap(m, n);
        int l = 0, r = n*m+1;
        while(l < r-1){
            int c = (l+r)/2, cnt=0;
            for(int i=1; i<=n && i<=c-1; ++i) cnt += min(m, (c
-1)/i);
            if(cnt<k) l = c;
            else r = c;
        }
        return l;
    }
};
```

329

1. dfs + memo:

```
class Solution {
    int d[4] = {1, -1, 0, 0}, n, m;
```

```
    vector<vector<int>> dp;
    int dfs(int i, int j, vector<vector<int>>& M){
        if(dp[i][j] >=0 ) return dp[i][j];
        dp[i][j] = 1;
        for(int k=0; k<4; ++k) {
            int i1 = i + d[k], j1 = j + d[3-k];
            if(i1>=0 && i1<n && j1>=0 && j1<m && M[i1][j1]>M
[i][j]){
                dp[i][j] = max(dp[i][j], 1 + dfs(i1, j1, M));
            }
        }
        return dp[i][j];
    }
public:
    int longestIncreasingPath(vector<vector<int>>& matrix) {
        if(matrix.empty() || matrix[0].empty()) return 0;
        n = matrix.size();
        m = matrix[0].size();
        dp = vector<vector<int>>(n, vector<int>(m, -1));
        int ans = 0;
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) ans = ma
x(ans, dfs(i, j, matrix));
        return ans;
    }
};
```

421

1. 蠢的一B之DFS： Complexity should be `O (32 * n)`

```
class Solution {
    int getMax(int j, vector<int>& A, vector<int>&B){
        if(j<0) return 0;
```

```
        vector<int> A0, A1, B0, B1;
        for(int &a: A) {
            if(1&(a>>j)) A1.push_back(a);
            else A0.push_back(a);
        }
        for(int &b: B){
            if(1&(b>>j)) B1.push_back(b);
            else B0.push_back(b);
        }
        if((A0.empty()||B1.empty()) && (A1.empty()||B0.empty
())) return getMax(j-1, A, B);
        if(A0.empty() || B1.empty()) return (1<<j) + getMax(j-
1, A1, B0);
        if(A1.empty() || B0.empty()) return (1<<j) + getMax(j-
1, A0, B1);
        return (1<<j) + max(getMax(j-1, A1, B0), getMax(j-1, A
0, B1));
    }
public:
    int findMaximumXOR(vector<int>& nums) {
        return getMax(31, nums, nums);
    }
};
```

- [ ] 这个解法值得一看：https://leetcode.com/problems/maximum-xor-of-two-numbers-in-an-array/discuss/130427/()-Beats-92 @Zebo L
- [ ] 为什么都用树做啊：https://leetcode.com/problems/maximum-xor-of-two-numbers-in-an-array/discuss/91052/C++-Trie-69ms-beats-85 @Zebo L

340

1. Sliding window:

```
class Solution {
public:
```

```
    int lengthOfLongestSubstringKDistinct(string s, int k) {
        unordered_map<char, int> cnt;
        int i=0, j = 0, ans = 0;
        while(i<s.size() && j<s.size()){
            while(j<s.size() && cnt.size()<=k){
                cnt[s[j]]++;
                ++j;
            }
            if(cnt.size()<=k) ans = max(ans, j-i);
            else ans = max(ans, j-i-1);
            while(i<j && cnt.size()>k){
                cnt[s[i]]--;
                if(!cnt[s[i]]) cnt.erase(s[i]);
                ++i;
            }
        }
        return ans;
    }
};
```

- [ ] Try this one: https://leetcode.com/problems/longest-substring-with-at-most-k-distinct-characters/discuss/80044/Java-O(nlogk)-using-TreeMap-to-keep-last-occurrence-Interview-%22follow-up%22-question! @Zebo L

612

恐怖的SQL

322

1. dp backpack

```
class Solution {
    public int coinChange(int[] coins, int amount) {
        long[] dp = new long[amount + 1];
```

```
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[0] = 0;
        for (int i = 1; i <= amount; i++) {
            for (int j = 0; j < coins.length; j++) {
                if (coins[j] <= i) {
                    dp[i] = Math.min(dp[i], dp[i - coins[j]] +
1);
                }
            }
        }
        return dp[amount] == Integer.MAX_VALUE ? -1 : (int) dp
[amount];
    }
}
```

2. Similar as above (看了楼上的才想出来)，但还是很慢：dfs + memo `600 ms`

```
class Solution {
    unordered_map<int, int> dp;
    const int inf = 1E7;
    int dfs(int tar, vector<int>&C){
        if(!tar) return 0;
        if(dp.count(tar)) return dp[tar];
        dp[tar] = inf;
        for(int k: C) if(tar >= k) dp[tar] = min(dp[tar], 1 +
dfs(tar-k, C));
        return dp[tar];
    }
public:
    int coinChange(vector<int>& coins, int amount) {
        int ans = dfs(amount, coins);
        return (ans == inf? -1: ans);
    }
```

```
};
```

3. BFS: `172 ms`

```
class Solution {
    typedef pair<int, int> ii;
public:
    int coinChange(vector<int>& C, int K) {
        if(!K) return 0;
        unordered_map<int, int> dp;
        dp[0] = 0;
        queue<ii> Q;
        Q.push(ii(0, 0));
        while(!Q.empty()){
            int k = Q.front().first, l = Q.front().second;
            Q.pop();
            for(int m: C) if(k+m<=K && !dp.count(k+m)) {
                if(k+m == K) return l+1;
                Q.push(ii(k+m, l+1));
                dp[k+m] = l+1;
            }
        }
        return -1;
    }
};
```

4. Pure dp: 跟1 完全相同的方法 `32 ms`

```
class Solution {
public:
    int coinChange(vector<int>& C, int K) {
        sort(C.begin(), C.end());
        vector<int> dp(K+1, -1);
        dp[0] = 0;
```

```
        for(int i=0; i<K; ++i) if(dp[i]>=0){
            for(int j=0; j<C.size() && i<=K - C[j]; ++j){
                if(dp[i+C[j]] < 0) dp[i+C[j]] = dp[i] + 1;
                else dp[i+C[j]] = min(dp[i+C[j]], 1+dp[i]);
            }
        }
        return dp[K];
    }
};
```

5. 改进版BFS，其实就是吧`unordered_set/map` 改成`vector<bool>` `32 ms`, 居然快了这么多：

```
class Solution {
    typedef pair<int, int> ii;
public:
    int coinChange(vector<int>& C, int K) {
        if(!K) return 0;
        sort(C.begin(), C.end());
        vector<bool> mark(K+1, false);
        mark[0] = true;
        queue<ii> Q;
        Q.push(ii(0, 0));
        while(!Q.empty()){
            int k = Q.front().first, l = Q.front().second;
            Q.pop();
            for(int j=0; j<C.size() && k<=K-C[j]; ++j) if(!mar
k[k+C[j]]) {
                if(k+C[j] == K) return l+1;
                Q.push(ii(k+C[j], l+1));
                mark[k + C[j]] = true;
            }
        }
```

```
        return -1;
    }
};
```

195

1. Bash 似懂非懂（Linux 还是像坨屎）

```
awk 'NR==10' file.txt
```

202

1.

```
class Solution {
    Set<Integer> set = new HashSet<>();
    public boolean isHappy(int n) {
        if (n == 1) return true;
        if (!set.add(n)) return false;
        return isHappy(sum(n));
    }

    public int sum(int n) {
        int sum = 0;
        while (n != 0) {
            int k = n % 10;

            sum += k * k;
            n /= 10;
        }
        return sum;
    }
}
```

2. 同上，无脑simulation：

```
class Solution {
```

```
public:
    bool isHappy(int n) {
        unordered_set<int> S;
        while(n!=1 && !S.count(n)){
            S.insert(n);
            int cnt = 0;
            while(n){
                cnt += (n%10) * (n%10);
                n /= 10;
            }
            swap(n, cnt);
        }
        return n==1;
    }
};
```

208

1. Tier 实现：

```
class Trie {
    struct T{
        bool W;
        unordered_map<char, T*> C;
        T(): W(false){}
    };
    T * root;
public:
    /** Initialize your data structure here. */
    Trie(): root(new T()){}
    void insert(string word) {
        T* r = root;
```

```
        for(char c: word){
            if(!r->C.count(c)) r->C[c] = new T();
            r = r->C[c];
        }
        r->W = true;
    }
    bool search(string word) {
        T *r = root;
        for(char c: word){
            if(!r->C.count(c)) return false;
            r = r->C[c];
        }
        return r->W;
    }
    bool startsWith(string prefix) {
        T *r = root;
        for(char c: prefix){
            if(!r->C.count(c)) return false;
            r = r->C[c];
        }
        return true;
    }
};
```

592

1. 细心点即可：

```
class Solution {
    typedef pair<int, int> ii;
    int gcd(int x, int y){
        if(x < y) swap(x, y);
```

```
        if(!y) return x;
        return gcd(y, x%y);
    }
public:
    string fractionAddition(string s) {
        ii ans = ii(0, 1);
        while(true){
            int x = stoi(s);
            s = s.substr(s.find('/') + 1);
            int y = stoi(s);
            int n = ans.first * y + ans.second * x, d = ans.se
cond * y;
            int c = gcd(abs(n), abs(d));
            ans = ii(n/c, d/c);
            auto j = s.find_first_of("+-");
            if(j == string::npos) break;
            s = s.substr(j);
        }
        return to_string(ans.first) + "/" + to_string(ans.seco
nd);
    }
};
```

234

1. reverse the middle-right part

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
```

```
 * }
 */
class Solution {
    public boolean isPalindrome(ListNode head) {
        if (head == null) return true;
        ListNode mid = head, fast = head;
        while (fast.next != null && fast.next.next != null) {
            fast = fast.next.next;
            mid = mid.next;
        }
        mid = reverse(mid.next, null);

        while (head != null && mid != null) {
            if (head.val != mid.val) return false;
            head = head.next;
            mid = mid.next;
        }
        return true;
    }

    public ListNode reverse(ListNode head, ListNode pre) {
        if (head == null) return pre;
        ListNode tmp = head.next;
        head.next = pre;
        return reverse(tmp, head);
    }
}
```

2. 同上，别粗心即可：

```
class Solution {
    ListNode *rev(ListNode *head){
```

```
        if(!head || !head->next) return head;
        ListNode *p1 = head->next;
        for(head->next=NULL; p1;){
            ListNode *p2 = p1->next;
            p1->next = head;
            head = p1;
            p1 = p2;
        }
        return head;
    }
public:
    bool isPalindrome(ListNode* head) {
        if(!head || !head->next) return true;
        ListNode *slow=head, *fast=head->next;
        while(fast && fast->next){
            slow = slow->next;
            fast = fast->next->next;
        }
        fast=rev(slow->next);
        for(slow->next=NULL; fast && head; fast=fast->next, he
ad=head->next) if(fast->val != head->val) return false;
        return true;
    }
};
```

600

1. prefix大法，处理好细节就行了：

```
class Solution {
public:
    int findIntegers(int n) {
        string s;
```

```
        while(n){
            s += char(n%2 + '0');
            n /= 2;
        }
        reverse(s.begin(), s.end());
        int l = s.size();
        vector<int> dp(l+1, 0);
        dp[l] = 1;
        dp[l-1] = 2;
        for(int i=l-2; i>=0; --i) dp[i] = dp[i+1] + dp[i+2];
        int ans = 1;
        for(int j=1; j<l && ans; ++j) if(s[j]=='1' && s[j-1]=
='1') ans = 0;
        ans += dp[1];
        for(int i=1; i<l; ++i){
            if(s[i]=='1'){
                if(s[i-1] == '1') {
                    ans += dp[i+1];
                    break;
                }
                else ans += dp[i+1];
            }
        }
        return ans;
    }
};
```

- [ ] 以下方法看着很酷炫，值得研究一下： @Zebo L

```
function findIntegers(num) {
  let res = 1;
  dfs(1);
  return res;

  function dfs(n) {
    if (n > num || (n&3)===3) return;

    res++;
    dfs(n*2);
    dfs(n*2+1);
  }
}
```

126

1. 这道题的test case跟一年前不一样了。。。

```
class Solution {
    public List<List<String>> findLadders(String beginWord, St
ring endWord, List<String> wordList) {
        List<List<String>> res = new ArrayList<>();
        if (!wordList.contains(endWord)) return res;
        Set<String> beginSet = new HashSet<>();
        Set<String> endSet = new HashSet<>();
        Map<String, List<String>> map = new HashMap<>();
        beginSet.add(beginWord);
        endSet.add(endWord);
        boolean flag = true;
        int length = 1;
        int max = Integer.MAX_VALUE;
        while (!beginSet.isEmpty() && !endSet.isEmpty()) {
            if (beginSet.size() > endSet.size()) {
                Set<String> tmp = beginSet;
                beginSet = endSet;
                endSet = tmp;
                flag = !flag;
            }

            wordList.removeAll(beginSet);
```

```
            wordList.removeAll(endSet);
            if (length >= max) break;
            Set<String> tmp = new HashSet<>();
            for (String s : beginSet) {
                char[] ch = s.toCharArray();
                for (int i = 0; i < ch.length; i++) {
                    char t = ch[i];
                    for (char c = 'a'; c <= 'z'; c++) {
                        if (t == c) continue;
                        ch[i] = c;
                        String cur = new String(ch);
                        String key = flag ? s : cur;
                        String value = flag ? cur : s;
                        if (endSet.contains(cur)) {
                            max = length + 1;
                            if (!map.containsKey(key)) {
                                map.put(key, new ArrayList<>
());
                            }
                            map.get(key).add(value);
                        }
                        if (max != length + 1 && wordList.cont
ains(cur)) {
                            tmp.add(cur);
                            if (!map.containsKey(key)) {
                                map.put(key, new ArrayList<>
());
                            }
                            map.get(key).add(value);
                        }
                    }
```

```
                    ch[i] = t;
                }
            }
            beginSet = tmp;
            length++;
        }

        List<String> list = new ArrayList<>();
        list.add(beginWord);
        helper(map, list, beginWord, endWord, res);
        return res;
    }

    public void helper(Map<String, List<String>> map, List<Str
ing> list, String beginWord, String endWord, List<List<String>
> res) {
        if (beginWord.equals(endWord)) {
            res.add(new ArrayList<>(list));
            return;
        }
        if (!map.containsKey(beginWord)) return;
        List<String> tmp = map.get(beginWord);
        for (String s : tmp) {
            list.add(s);
            helper(map, list, s, endWord, res);
            list.remove(list.size() - 1);
        }
    }
}
```

1. dfs + memo:

```
class Solution {
    vector<vector<int>> dp;
    bool dfs(int i, int cnt, string&s){
        if(cnt<0) return false;
        if(i==s.size()) return cnt==0;
        if(dp[i][cnt]>=0) return dp[i][cnt];
        if(s[i]=='(') return dp[i][cnt] = dfs(i+1, cnt+1, s);
        if(s[i]==')') return dp[i][cnt] = dfs(i+1, cnt-1, s);
        return dp[i][cnt] = (dfs(i+1, cnt, s)) || (dfs(i+1, cn
t+1, s)) || (dfs(i+1, cnt-1, s));
    }
public:
    bool checkValidString(string s) {
        dp = vector<vector<int>>(s.size()+1, vector<int>(s.siz
e()+1, -1));
        return dfs(0, 0, s);
    }
};
```