July 26, 2018 题目：411,657,562,330,402,740,687,102,67,516,536,610

411

1. Trie + backtracking + DFS

```
class Solution {
    class TrieNode {
        TrieNode[] children;
        boolean isWord;
        public TrieNode() {
            children = new TrieNode[26];
            isWord = false;
        }
    }
    TrieNode root = new TrieNode();
    public void build(String s) {
        TrieNode node = root;
        for (char c : s.toCharArray()) {
            if (node.children[c - 'a'] == null) {
                node.children[c - 'a'] = new TrieNode();
            }
            node = node.children[c - 'a'];
        }
        node.isWord = true;
    }

    public void getAbbr(List<String> res, String word, int i,
String cur, int count) {
```

```
        if (i == word.length()) {
            if (count > 0) cur += count;
            res.add(cur);
        } else {
            getAbbr(res, word, i + 1, cur, count + 1);
            getAbbr(res, word, i + 1, cur + (count > 0 ? count
: "") + word.charAt(i), 0);
        }
    }

    public boolean find(String s, TrieNode node, int i, int nu
m) {
        if (node == null) return false;
        if (num != 0) {
            for (int k = 0; k < 26; k++) {
                if (find(s, node.children[k], i, num - 1)) ret
urn true;
            }
            return false;
        }

        if (i == s.length()) {
            return node.isWord;
        }

        if (Character.isDigit(s.charAt(i))) {
            int val = 0;
            while (i < s.length() && Character.isDigit(s.charA
t(i))) {
                val = val * 10 + (s.charAt(i) - '0');
                i++;
```

```
            }
            return find(s, node, i, val);
        } else {
            return find(s, node.children[s.charAt(i) - 'a'], i
+ 1, 0);
        }
    }
    public String minAbbreviation(String target, String[] dict
ionary) {
        if (dictionary == null || dictionary.length == 0) retu
rn target.length() + "";

        Set<String> set = new HashSet<>();
        for (String s : dictionary) {
            if (s.length() == target.length()) set.add(s);
        }
        if (set.size() == 0) return target.length() + "";
        for (String s : dictionary) {
            build(s);
        }

        List<String> list = new ArrayList<>();
        getAbbr(list, target, 0, "", 0);

        Collections.sort(list, new Comparator<String>(){
            public int compare(String a, String b) {
                return a.length() != b.length() ? a.length() -
b.length() : a.charAt(0) - b.charAt(0);
            }
        });
```

```
        for (String s : list) {
            if (!find(s, root, 0, 0)) return s;
        }

        return "";
    }
}
```

1. 暴力枚举：

```
class Solution {
public:
    string minAbbreviation(string target, vector<string>& dict
ionary) {
        int n = target.size();
        if(!n) return target;
        unordered_set<string> P;
        for(string s: dictionary) if(s.size()==n) P.insert(s);
        if(P.empty()) return to_string(n);
        int res = n, cur = (1<<n)-1;
        for(int k=1; k<(1<<n)-1; ++k){
            int l = 0;
            unordered_set<string> tmp(P.begin(), P.end());
            for(int j=0; j<n; ++j){
                if((!j) || (1&(k>>j)) || (1&(k>>(j-1)))) ++l;
                if(1&(k>>j) && !tmp.empty()){
                    for(auto it=tmp.begin(); it!=tmp.end(); ){
                        if((*it)[j] == target[j]) ++it;
                        else it = tmp.erase(it);
                    }
                }
            }
```

```
            if(!(1&&(k>>(n-1)))) ++l;
            if(tmp.empty() && l<res){
                res = l;
                cur = k;
            }
        }
        string ans;
        int cnt = 0;
        for(int j=0; j<n; ++j){
            if(1&(cur>>j)){
                if(cnt){
                    ans += to_string(cnt);
                    cnt = 0;
                }
                ans += target[j];
            }
            else ++cnt;
        }
        if(cnt) ans += to_string(cnt);
        return ans;
    }
};
```

- [x] ~~Investigate solution #1 @Zebo L~~

3. Bit + Trie:

```
#define CI(c) int((c) - 'a')

int L;

struct T{
    vector<T *> S;
```

```
    T() : S(vector<T*>(26, NULL)) {}
};

void inT(T* root, string s){
    for(char c: s){
        if(!root->S[CI(c)]) root->S[CI(c)] = new T();
        root = root->S[CI(c)];
    }
}

bool macH(T *root, int j, const int&stat, const string &s){
    if(!root) return false;
    if(j == L) return true;
    if(1&(stat>>j)) return macH(root->S[CI(s[j])], j+1, stat,
s);
    for(int i=0; i<26; ++i) if(macH(root->S[i], j+1, stat, s))
return true;
    return false;
}

int getLength(int stat){
    int ans = 0;
    for(int j=0; j<L; ++j) if(!j || (1&(stat>>j)) || (1&(stat>
>(j-1)))) ++ans;
    return ans;
}

string getAbbr(int stat, const string&s){
    string ans;
    int cnt = 0;
    for(int j=0; j<L; ++j){
```

```
        if(1&(stat>>j)){
            if(cnt) ans += to_string(cnt);
            cnt = 0;
            ans += s[j];
        }
        else ++cnt;
    }
    if(cnt) ans += to_string(cnt);
    return ans;
}

class Solution {
public:
    string minAbbreviation(string target, vector<string>& dict
ionary) {
        if(target.empty()) return target;
        bool ok = true;
        L = target.size();
        T * root = new T();
        for(string s: dictionary) if(s.size()==L) {
            inT(root, s);
            ok = false;
        }
        if(ok) return to_string(L);
        int state = (1<<L)-1, ml = L, tmpl;
        for(int i=0; i<(1<<L)-1; ++i) if(!macH(root, 0, i, tar
get) && (tmpl = getLength(i))<ml){
            state = i;
            ml = tmpl;
        }
```

```
        return getAbbr(state, target);
    }
};
```

657

1. 算坐标

```
class Solution {
    public boolean judgeCircle(String moves) {
        if (moves == null || moves.length() < 1) return true;
        int x = 0, y = 0;
        for (char c : moves.toCharArray()) {
            if (c == 'U') {
                y++;
            } else if (c == 'D') {
                y--;
            } else if (c == 'L') {
                x--;
            } else if (c == 'R') {
                x++;
            }
        }
        return x == 0 && y == 0;
    }
}
```

2. 向相反方向走的步数相同即可:

```
class Solution {
public:
    bool judgeCircle(string moves) {
        int cnt[26]={0};
        for(char c: moves) cnt[int(c-'A')]++;
```

```
        return cnt[int('L'-'A')]==cnt[int('R'-'A')] && cnt[int
('D'-'A')]==cnt[int('U'-'A')];
    }
};
```

562

1.

```
class Solution {
    public int longestLine(int[][] M) {
        if (M.length == 0 || M[0].length == 0) return 0;
        int max = 0;
        int[] diag = new int[M.length + M[0].length];
        int[] antiDiag = new int[M.length + M[0].length];
        int[] col = new int[M[0].length];

        for (int i = 0; i < M.length; i++) {
            int row = 0;
            for (int j = 0; j < M[0].length; j++) {
                if (M[i][j] == 1) {
                    row++;
                    col[j]++;
                    diag[i + j]++;
                    antiDiag[j - i + M.length]++;
                    max = Math.max(Math.max(Math.max(Math.max
(row, max), col[j]), diag[i + j]), antiDiag[j - i + M.lengt
h]);
                } else {
                    row = 0;
                    col[j] = 0;
                    diag[i + j] = 0;
                    antiDiag[j - i + M.length] = 0;
```

```
                }
            }
        }
        return max;

    }
}
```

2. 扫一遍就行了：

```
class Solution {
public:
    int longestLine(vector<vector<int>>& M) {
        if(M.empty() || M[0].empty()) return 0;
        int ans = 0, n = M.size(), m = M[0].size();
        vector<int> x(n, 0), xy(n+m, 0), yx(n+m, 0);
        for(int j=0; j<m; ++j){
            for(int i=0, y=0;i<n;++i){
                if(M[i][j]) {
                    ++x[i];
                    ++xy[i+j];
                    ++yx[i+m-j];
                    ++y;
                }
                else x[i] = xy[i+j] = yx[i+m-j] = y = 0;
                ans = max(ans, max(max(x[i], y), max(xy[i+j],
yx[i+m-j])));
            }
        }
        return ans;
    }
};
```

330

1. From left to right, add up the numbers. If the sum cannot reach the current number, it needs one more number.

```
class Solution {
    public int minPatches(int[] nums, int n) {
        int res = 0;
        long sum = 0;
        int i = 0;
        while (sum < n) {
            if (i >= nums.length || sum < nums[i] - 1) {
                sum += sum + 1;
                res++;
            } else {
                sum += nums[i];
                i++;
            }
        }

        return res;
    }
}
```

2. 同上: maintain 到 `n` ， check `n+1`是否存在，不存在补上，然后 `n→ n+1` 或 `n→n+nums[j]`

```
class Solution {
public:
    int minPatches(vector<int>& nums, int n) {
        sort(nums.begin(), nums.end());
        long cur = 0, j = 0, ans = 0;
        while(cur<n){
            if(j<nums.size() && cur+1 >= nums[j]) cur += nums
[j++];
```

```
            else{
                cur += cur+1;
                ++ans;
            }
        }
        return ans;
    }
};
```

402

1. Use stack to delete all the elements that are larger than the current number. Use Deque for optimization for string builder.

```
class Solution {
    public String removeKdigits(String num, int k) {
        Deque<Integer> s = new LinkedList<>();
        for (int i = 0; i < num.length(); i++) {
            int n = num.charAt(i) - '0';
            while (k > 0 && !s.isEmpty() && s.peekFirst() > n)
{
                s.removeFirst();
                k--;
            }
            s.addFirst(n);
        }

        while (k > 0) {
            s.removeFirst();
            k--;
        }

        StringBuilder sb = new StringBuilder();
```

```
        while (!s.isEmpty()) {
            if (s.getLast() == 0 && sb.length() == 0) s.removeLast();
            else sb.append(s.pollLast());
        }
        if (sb.length() == 0) return "0";
        return sb.toString();
    }
}
```

2. 每次减第一个递减数对：

```
class Solution {
public:
    string removeKdigits(string num, int k) {
        while(k){
            int j=0;
            while(j<num.size()-1 && num[j]<=num[j+1]) ++j;
            num = num.substr(0, j) + num.substr(j+1);
            j = 0;
            while(j<num.size() && num[j]=='0') ++j;
            num = num.substr(j);
            if(num.empty()) return "0";
            --k;
        }
        return num;
    }
};
```

3. Stack is faster:

```
class Solution {
public:
    string removeKdigits(string num, int k) {
```

```
        if(!k) return num;
        string ans;
        for(char c: num){
            while(!ans.empty() && c<ans.back() && k){
                ans.pop_back();
                --k;
            }
            ans += c;
        }
        while(k && !ans.empty()){
            ans.pop_back();
            --k;
        }
        auto j = ans.find_first_not_of("0");
        return (j==string::npos? "0":ans.substr(j));
    }
};
```

740

1. dp:

```
class Solution {
public:
    int deleteAndEarn(vector<int>& nums) {
        if(nums.empty()) return 0;
        map<int, int> cnt;
        for(int n:nums) cnt[n]++;
        vector<int> dp(cnt.size()+1, 0), A;
        for(auto p: cnt) A.push_back(p.first);
        dp[1] = A[0] * cnt[A[0]];
        for(int j=2; j<=cnt.size(); ++j){
```

```
            if(A[j-1] > A[j-2]+1) dp[j] = dp[j-1] + A[j-1]*cnt
[A[j-1]];
            else dp[j] = max(dp[j-1], dp[j-2] + A[j-1]*cnt[A[j
-1]]);
        }
        return dp[A.size()];
    }
};
```

687

1. Easy DFS:

```
class Solution {
    int res = 0;
    int dfs(TreeNode *root, int v){
        if(!root) return 0;
        int l = dfs(root->left, root->val), r = dfs(root->righ
t, root->val);
        res = max(res, l + r);
        return (root->val==v? 1+max(l,r): 0);
    }
public:
    int longestUnivaluePath(TreeNode* root) {
        dfs(root, 0);
        return res;
    }
};
```

102

1. Easy dfs:

```
class Solution {
    void dfs(vector<vector<int>> &ans, TreeNode *root, int lev
el){
```

```
        if(!root) return;
        if(level>=ans.size()) ans.resize(level+1);
        ans[level].push_back(root->val);
        dfs(ans, root->left, level+1);
        dfs(ans, root->right, level+1);
    }
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> ans;
        dfs(ans, root, 0);
        return ans;
    }
};
```

67

1.

```
class Solution {
    public String addBinary(String a, String b) {
        StringBuilder sbA = new StringBuilder(a).reverse();
        StringBuilder sbB = new StringBuilder(b).reverse();
        StringBuilder sb = new StringBuilder();
        int remain = 0;
        int i = 0;
        while (i < Math.max(sbA.length(), sbB.length())) {
                int sum = (i < sbA.length() ? sbA.charAt(i) -
'0' : 0) + (i < sbB.length() ? sbB.charAt(i) - '0' : 0) + rema
in;
                if (sum % 2 == 0) {
                    sb.append(0);
                } else {
                    sb.append(1);
```

```
                }
            remain = sum / 2;
            i++;
        }
        if (remain != 0) sb.append(1);
        return sb.reverse().toString();
    }
}
```

2. 2 * reverse:

```
class Solution {
    #define CI(c) int((c) - '0')
    #define IC(i) char((i) + '0')
public:
    string addBinary(string a, string b) {
        reverse(a.begin(), a.end());
        reverse(b.begin(), b.end());
        int i = 0, j = 0, d = 0;
        string c;
        while(i<a.size() || j<b.size() || d){
            d += (i<a.size()?CI(a[i++]):0) + (j<b.size()?CI(b
[j++]):0);
            c += IC(d%2);
            d /= 2;
        }
        reverse(c.begin(), c.end());
        return c;
    }
};
```

516

1. DP解法

```
class Solution {
    public int longestPalindromeSubseq(String s) {
        int[][] dp = new int[s.length()][s.length()];
        for (int i = s.length() - 1; i >= 0; i--) {
            dp[i][i] = 1;
            for (int j = i + 1; j < s.length(); j++) {
                if (s.charAt(i) == s.charAt(j)) {
                    dp[i][j] = dp[i + 1][j - 1] + 2;
                } else {
                    dp[i][j] = Math.max(dp[i + 1][j], dp[i][j
- 1]);
                }
            }
        }
        return dp[0][s.length() - 1];
    }
}
```

2. 同上：

```
class Solution {
public:
    int longestPalindromeSubseq(string s) {
        if(s.empty()) return 0;
        int n = s.size();
        vector<vector<int>> dp(n, vector<int>(n, 0));
        for(int i=0;i<n;++i) dp[i][i] = 1;
        for(int l=1; l<n; ++l) for(int i=0; i<n-l;++i){
            int j = i+l;
            if(s[i]==s[j]) dp[i][j] = dp[i+1][j-1] + 2;
            else dp[i][j] = max(dp[i+1][j], dp[i][j-1]);
        }
```

```
        return dp[0][s.size()-1];
    }
};
```

- [ ] Looking for other solutions @Zebo L

536

1. Recursion:

```
class Solution {
public:
    TreeNode* str2tree(string s) {
        if(s.empty()) return NULL;
        int val = stoi(s);
        TreeNode *root = new TreeNode(val);
        s = s.substr(to_string(val).size());
        if(s.empty()) return root;
        int i = 1, cnt = 1, k = 1;
        for(; k<s.size()&&cnt; ++k){
            if(s[k]=='(') ++cnt;
            if(s[k]==')') --cnt;
        }
        root->left = str2tree(s.substr(1, k-2));
        s = s.substr(k);
        if(!s.empty()) root->right = str2tree(s.substr(1, s.si
ze()-2));
        return root;
    }
};
```

- [ ] Check other solutions @Zebo L

610

SQL