July 10, 2018

@Mingze X @Chong T @Zhaorui D @Yu S

84

1. 经典stack解法：

```
class Solution {
public:
    int largestRectangleArea(vector<int>& heights) {
        heights.insert(heights.begin(), -2);
        heights.push_back(-1);
        stack<int> S;
        int ans = 0, h;
        S.push(0);
        for(int i=1;i<heights.size();++i){
            while(!S.empty() && (h=heights[S.top()]) >= heights[i]){
                S.pop();
                ans = max(ans, h * (i-1-S.top()));
            }
            S.push(i);
        }
        return ans;
    }
};
```

2. 单调栈，单调增序，pop出来的就确定了左右边界。栈存index

```
class Solution {
    public int largestRectangleArea(int[] heights) {
        if (heights == null || heights.length == 0) {
            return 0;
        }
        Stack<Integer> st = new Stack<>();
        st.push(-1);
        int res = 0;
        for (int i = 0; i < heights.length; i++) {
            while (st.peek() != -1 && heights[st.peek()] >= heights[i]) {
                res = Math.max(res, heights[st.pop()] * (i - st.peek() - 1));
            }
            st.push(i);
        }
        while (st.peek() != -1) {
            res = Math.max(res, heights[st.pop()] * (heights.length - st.peek() - 1));
        }
        return res;
    }
}
```

686

1. 简单brute force 一下即可:

```
class Solution {
public:
    int repeatedStringMatch(string A, string B) {
        int n = A.size(), m = B.size();
        int k = (m + n-1)/n;
        string tmp = A;
        for(int i=1;i<k;++i) tmp += A;
        for(int j=0; j<=1; ++j){
            if(tmp.find(B) != string::npos) return k+j;
            tmp += A;
        }
        return -1;
    }
};
```

2.

```
class Solution {
    public int repeatedStringMatch(String A, String B) {
        StringBuffer sb = new StringBuffer();
        int res = 0;
        while(sb.length() < B.length()) {
            sb.append(A);
            res++;
        }
        if (sb.toString().indexOf(B) >= 0) {
            return res;
        }
        if (sb.append(A).toString().indexOf(B) >= 0) {
            return res + 1;
        }
        return -1;
    }
}
```

627

SQL

313

1. 二分法：史上最慢solution：(1700 ms)

```
class Solution {
    int N;
    int dfs(int i, int k, vector<int>& primes){
        if(i==N) return 1;
        int ans = 0;
        while(k){
            ans += dfs(i+1, k, primes);
            k /= primes[i];
        }
        return ans;
    }
public:
    int nthSuperUglyNumber(int n, vector<int>& primes) {
        if(n==1) return 1;
        reverse(primes.begin(), primes.end());
        N = (int)primes.size();
        long l = 1, r = INT_MAX;
        while(l<r-1){
            int m = (int)((l+r)/2);
            int cnt = dfs(0, m, primes);
            if(cnt >= n) r = m;
            else l = m;
        }
        return r;
    }
};
```

- □ See Leetcode discussion @Zebo L

2. Priority queue, complexity `O(k * log(k) * n)`:

```
class Solution {
    typedef pair<int, int> ii;
public:
    int nthSuperUglyNumber(int n, vector<int>& primes) {
        vector<int> dp(n, 1), idx(primes.size(), 1);
        priority_queue<ii, vector<ii>, greater<ii>> Q;
        for(int i=0;i<primes.size(); ++i) Q.push(ii(primes[i], i));
        for(int i=1; i<n; ++i){
            dp[i] = Q.top().first;
            while(Q.top().first == dp[i]){
                int k = Q.top().second, val;
                Q.pop();
                while((val=dp[idx[k]] * primes[k]) <= dp[i]) ++idx[k];
                Q.push(ii(val, k));
            }
        }
        return dp[n-1];
    }
};
```

3. Dijkstra. TLE at case 81.

```
class Solution {
    public int nthSuperUglyNumber(int n, int[] primes) {
        if (n == 1) {
            return 1;
        }
        Set<Integer> visited = new HashSet<>();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.offer(1);
        visited.add(1);
        n--;
        while (n > 0) {
            int cur = pq.remove();
            visited.add(cur);
            for (int p : primes) {
                if (!visited.contains(cur * p)) {
                    pq.offer(cur * p);
                    visited.add(cur * p);
                }
            }
            n--;
        }
        return pq.remove();
    }
}
```

178

SQL

□ Urgently need to learn db. @Zebo L

38

1. Easy Recursion:

```
class Solution {
public:
    string countAndSay(int n) {
        if(n==1) return "1";
        string s = countAndSay(n-1) + '.', ans;
        char c = s[0];
        for(int i=1, cnt=1; i<s.size(); ++i){
            if(s[i] == c) ++cnt;
            else{
                ans += to_string(cnt) + c;
                c = s[i];
                cnt = 1;
            }
        }
        return ans;
    }
};
```

2. simulation.

```
class Solution {
    public String countAndSay(int n) {
        StringBuilder cur = new StringBuilder("1");
        StringBuilder prev;
        int count = 1;
        char say = ' ';
        for (int i = 1; i < n; i++) {
            prev = cur;
            cur = new StringBuilder();
            say = prev.charAt(0);
            count = 1;

            for (int j = 1; j < prev.length(); j++) {

                if (prev.charAt(j) != say) {
                    cur.append(count).append(say);
                    say = prev.charAt(j);
                    count = 1;
                } else {
                    count++;
                }
            }
            cur.append(count).append(say);
        }
        return cur.toString();
    }
}
```

433

1. 典型的BFS

```
class Solution {
    typedef pair<string, int> si;
public:
    int minMutation(string start, string end, vector<string>& bank) {
        if(start == end) return 0;
        unordered_set<string> pool(bank.begin(), bank.end());
        if(!pool.count(end)) return -1;
        if(pool.count(start)) pool.erase(start);
        queue<si> Q;
        Q.push(si(start, 0));
        while(!Q.empty()){
            string s = Q.front().first;
            int step = Q.front().second;
            if(s == end) return step;
            Q.pop();
            for(int i=0; i<s.size(); ++i){
                string tmp = s;
                for(char c: "ACGT") if(c!=s[i]){
                    tmp[i] = c;
                    if(pool.count(tmp)){
                        pool.erase(tmp);
                        Q.push(si(tmp, step+1));
                    }
                }
            }
        }
        return -1;
    }
};
```

394

1. stack，细心点就行了

```
#include<cassert>
class Solution {
    const string num = "0123456789";
public:
    string decodeString(string s) {
        int i=0, n=s.size(), cnt=1;
        string ans="";
        stack<string> SS;
        stack<int> SI;
        while(i<n){
            if(s[i]>='0' && s[i]<='9') {
                SI.push(cnt);
                SS.push(ans);
                cnt = stoi(s.substr(i));
                ans = "";
                i = s.find_first_not_of(num, i);
                assert(s[i] =='[');
                ++i;
            }
            else if(s[i] == ']'){
                string tmp = "";
                for(int j=0;j<cnt;++j) tmp+=ans;
                ans = SS.top() + tmp;
                cnt = SI.top();
                SS.pop();
                SI.pop();
                ++i;
            }
            else ans += s[i++];
        }
        return ans;
    }
};
```

481

1. Simulation:

```
class Solution {
public:
    int magicalString(int n) {
        string ans = "12211";
        for(int i=3; ans.size()<n; ++i){
            int l = ans.size();
            char c = (ans[l-1]=='1'? '2':'1');
            ans += c;
            if(ans[i] == '2'){
                ans += c;
            }
        }
        int res = 0;
        for(int i;i<n;++i) if(ans[i]=='1') ++res;
        return res;
    }
};
```

2.上面的答案很好，为啥这个magic string是唯一的呢？？？

639

1. dp, 细心一点就行了

```
class Solution {
    const int M = 1E9+7;
    void add(int &x, int y){
        x += y;
        if(x >= M) x -= M;
    }
    int mul(int x, int y){
        return long(x)*long(y)%M;
    }
public:
    int numDecodings(string s) {
        int n = s.size();
        vector<int> dp(n+1, 0);
        dp[n] = 1;
        if(s[n-1]=='0') dp[n-1] = 0;
        else if(s[n-1]=='*') dp[n-1] = 9;
        else dp[n-1] = 1;
        for(int j=n-2;j>=0;--j){
            if(s[j]=='0') dp[j] = 0;
            else{
                if(s[j]=='*'){
                    add(dp[j], mul(9, dp[j+1]));
                    int cnt = 1;
                    if(s[j+1] == '*') cnt = 15;
                    else if(s[j+1] <= '6') cnt = 2;
                    add(dp[j], mul(cnt, dp[j+2]));
                }
                else{
                    add(dp[j], dp[j+1]);
                    int cnt = 0;
                    if(s[j]=='1') cnt = (s[j+1]=='*'? 9:1);
                    else if(s[j]=='2') cnt = (s[j+1]=='*'? 6:int(s[j+1]<='6'));
                    add(dp[j], mul(cnt, dp[j+2]));
                }
            }
        }
        return dp[0];
    }
};
```

2. 注意与Decode the ways 1的区别，1是加法，2是乘法（对于*）

```
class Solution {
    public int numDecodings(String s) {
        if (s == null || s.length() == 0) {
            return 0;
        }
        int mod = 1000000007;

        long[] dp = new long[s.length() + 1];

        dp[0] = 1;
        if (s.charAt(0) != '0') {
            if (s.charAt(0) == '*') {
                dp[1] = 9;
            } else {
                dp[1] = 1;
            }
        }

        for (int i = 2; i < s.length() + 1; i++) {
            if (s.charAt(i - 1) == '*') {
                dp[i] = 9 * dp[i - 1];
                if (s.charAt(i - 2) == '2') {
                    dp[i] = (dp[i] + 6 * dp[i - 2]) % mod;
                } else if (s.charAt(i - 2) == '1') {
                    dp[i] = (dp[i] + 9 * dp[i - 2]) % mod;
                } else if (s.charAt(i - 2) == '*') {
                    dp[i] = (dp[i] + 15 * dp[i - 2]) % mod;
                }

            } else {
                dp[i] = s.charAt(i - 1) != '0' ? dp[i - 1] : 0;
                if (s.charAt(i - 2) == '1')
                    dp[i] = (dp[i] + dp[i - 2]) % mod;
                else if (s.charAt(i - 2) == '2' && s.charAt(i - 1) <= '6')
                    dp[i] = (dp[i] + dp[i - 2]) % mod;
                else if (s.charAt(i - 2) == '*')
                    dp[i] = (dp[i] + (s.charAt(i - 1) <= '6' ? 2 : 1) * dp[i - 2]) % mod;
            }
        }
        return (int)dp[s.length()];
    }
}
```