July 13, 2018

132

hard

1. O(n^2) solution:

```
class Solution {
public:
    int minCut(string s) {
        int n = s.size();
        unordered_map<int, unordered_set<int>> pos;
        for(int i=0;i<n;++i) {
            pos[i].insert(i);
        }
        for(int i=0;i<n-1;++i) if(s[i]==s[i+1]) pos[i].insert
(i+1);
        for(int l=2; l<n; ++l) for(int i=0;i<n-l;++i){
            int j = i+l;
            if(s[i] == s[j] && pos[i+1].count(j-1)) pos[i].ins
ert(j);
        }
        vector<int> dp(n, 0);
        for(int j=n-1; j>=0; --j){
            if(pos[j].count(n-1)) dp[j] = 0;
            else{
                dp[j] = n;
                for(int k:pos[j]){
                    dp[j] = min(dp[j], 1 + dp[k+1]);
                }
            }
        }
```

```
        for(int k: dp) cout<<k<<endl;
        return dp[0];
    }
};
```

2. Still O(2), but much faster, use `bool[][]` instead of `unordered_set<int>`:

```
class Solution {
public:
    int minCut(string s) {
        int n = s.size();
        vector<int> dp(n, 0);
        vector<vector<bool>> P(n, vector<bool>(n, false));
        for(int i=0; i<n-1; ++i) P[i+1][i] = P[i][i] = true;
        for(int j=n-2; j>=0; --j){
            dp[j] = 1 + dp[j+1];
            for(int k=j+1; k<n-1; ++k) if(P[j+1][k-1] && s[j]==s[k]){
                P[j][k] = true;
                dp[j] = min(dp[j], 1+dp[k+1]);
            }
            if(P[j+1][n-2] && s[j]==s[n-1]) dp[j] = 0;
        }
        return dp[0];
    }
};
```

784

1. 简单的计数问题

```
class Solution {
public:
    vector<string> letterCasePermutation(string S) {
        vector<string> ans{""};
```

```
        int cnt = 0;
        for(char c: S){
            if(isalpha(c)) {
                ++cnt;
                if(c >= 'A' && c <= 'Z') c += 'a' - 'A';
            }
            ans[0] += c;
        }
        for(int k=1; k<(1<<cnt); ++k){
            int m = k;
            string tmp;
            for(char c: ans[0]){
                if(isalpha(c)){
                    if(m%2) tmp += char(c + 'A' -'a');
                    else tmp += c;
                    m/=2;
                }
                else tmp += c;
            }
            ans.push_back(tmp);
        }
        return ans;
    }
};
```

683

hard

1. 维持一下区间即可：

```
class Solution {
public:
    int kEmptySlots(vector<int>& flowers, int k) {
```

```
        int n = flowers.size();
        set<int> S{-40000, 80000};
        for(int i=0;i<n;++i){
            auto it = S.upper_bound(flowers[i]);
            if(*it - flowers[i] == k+1) return i+1;
            --it;
            if(flowers[i] - *it == k+1) return i+1;
            S.insert(flowers[i]);
        }
        return -1;
    }
};
```

286

1. 简单的BFS

```
class Solution {
    const int INF = (1<<31) - 1;
    int dx[4] = {0, 0, 1, -1}, dy[4] = {1, -1, 0, 0};
public:
    void wallsAndGates(vector<vector<int>>& rooms) {
        if(rooms.empty() || rooms[0].empty()) return;
        int n = rooms.size(), m = rooms[0].size();
        queue<int> Q;
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) if(!rooms[i]
[j]) Q.push(i*m + j);
        while(!Q.empty()){
            int x = Q.front()/m, y = Q.front()%m;
            Q.pop();
            for(int k=0;k<4;++k){
                int x1 = x + dx[k], y1 = y + dy[k];
```

```
                if(x1>=0 && x1<n && y1>=0 && y1<m && rooms[x1]
[y1]==INF){
                    rooms[x1][y1] = rooms[x][y] + 1;
                    Q.push(x1 * m + y1);
                }
            }
        }
        return ;
    }
};
```

667

1. 简单的计数问题

```
class Solution {
public:
    vector<int> constructArray(int n, int k) {
        vector<int> ans;
        int l = 1, r = n, f = 0;
        while(k){
            if(f) ans.push_back(r--);
            else ans.push_back(l++);
            f = 1-f;
            --k;
        }
        if(ans.back() == l-1) for(;l<=r;++l) ans.push_back(l);
        else for(; r>=l;--r) ans.push_back(r);
        return ans;
    }
};
```

800

1. 用set 维持区间

```
class Solution {
    const string ref = "0123456789abcdef";
    map<char, int> xo;
    set<int> po;
    void init(){
        for(int i=0;i<ref.size();++i) xo[ref[i]] = i;
        for(int i=0;i<16;++i) po.insert(i*16 + i);
        po.insert(1000000);
        po.insert(-10000000);
    }
    int hex2int(string s){
        return xo[s[0]] * 16 + xo[s[1]];
    }
    string int2hex(int n){
        string ans;
        ans += ref[n/16];
        ans += ref[n%16];
        return ans;
    }
    string closest(string s){
        int k = hex2int(s), dif, ans;
        auto it = po.lower_bound(k);
        dif = *it - k;
        ans = *it;
        --it;
        if(k - *it <= dif){
            ans = *it;
        }
        return int2hex(ans);
```

```
    }
public:
    string similarRGB(string color) {
        init();
        return "#" + closest(color.substr(1, 2)) + closest(col
or.substr(3, 2)) + closest(color.substr(5, 2));
    }
};
```

141

1. 经典快慢指针

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if(!head || !head->next) return false;
        ListNode *fast = head->next, *slow = head;
        while(fast && fast->next){
            if(fast == slow) return true;
            fast = fast->next->next;
            slow = slow->next;
        }
        return false;
    }
};
```

214

1. `O(n^2)` 的二逼解法

```
class Solution {
public:
    string shortestPalindrome(string s) {
        string tmp(s);
```

```
        int n = s.size();
        reverse(tmp.begin(), tmp.end());
        for(int i=0; i<n; ++i) if(s.substr(0, n-i) == tmp.subs
tr(i)) return tmp.substr(0, i) + s;
        return "";
    }
};
```

2. Using hash

```
class Solution {
    const int Mod = 1E9 + 7;
    const int p = 37;
    int mul(int x, int y){
        return long(x) * long(y) % Mod;
    }
    int add(int x, int y){
        return (x + y) % Mod;
    }
public:
    string shortestPalindrome(string s) {
        int n = s.size();
        string tmp(s);
        reverse(tmp.begin(), tmp.end());
        vector<int> dp(n, 0), re(n, 0);
        for(int i=0; i<n; ++i){
            if(!i) dp[i] = (int)(s[i] - 'a');
            else dp[i] = add((int)(s[i] - 'a'), mul(p, dp[i-
1]));
        }
        for(int j=n-1, m = p; j>=0; --j){
            if(j == n-1) re[j] = (int)(tmp[j] - 'a');
            else{
```

```
                re[j] = add(re[j+1], mul((int)(tmp[j] - 'a'), m));
                m = mul(m, p);
            }
        }
        for(int i=0; i<n; ++i) if(dp[n-i-1] == re[i]) return tmp.substr(0, i) + s;
        return "";
    }
};
```

- [ ] Investigating the following one () @Zebo L

```
class Solution {
public:
    // A utility function to convert string str to the format $#str[0]#str[1]#str[2].....str[str.length()-1]#@
    string preProcess(string str) {
    string res_str(2*str.length(), '#');
    for(int i = 0; i < str.length(); ++i)
        res_str[(i << 1) + 1] = str[i];
    return '$' + res_str + "#@";
    }

    string shortestPalindrome(string str) {
        string str_temp;
        str_temp = preProcess(str);

        int r = 0, c = 0;// initially both right boundary and center are set to 0
        vector<int> p(str_temp.length(), 0);
        for(int i = 1; i < str_temp.length() - 1; ++i) { // notice how the loop is run, we would like to skip the characters
```

```
at both ends
            int mirror = 2 * c - i; // get the mirror of the c
urrent index
            if(r >= i)  // if the character at the current ind
ex is still withing the palindrome than we can
                p[i] = min(p[mirror], r - i);


            // start expanding across the center
            while (str_temp[i + p[i] + 1] == str_temp[i - p[i]
- 1])
                ++p[i]; // keep the center fixed at i and expa
nd at both ends, by incrementing the length


            if(p[i] > r - i) { // if current palindromic strin
g exceeds the right boundary
                c = i;
                r = i + p[i];
            }
        }


        int max_len = 0;
        for(int i = 1; i  < str_temp.length(); ++i) {
            if((i - p[i] - 1) / 2 == 0)
                max_len = max(max_len, p[i]);
        }


        string to_add = str.substr(max_len, str.length() - max
_len);
        reverse(to_add.begin(), to_add.end());
        return  to_add + str;
    }
```

```
};
```

664

1. 三维 dp

```
class Solution {
    string ref;
    unordered_map<char, set<int>> pos;
    int n, dp[30][105][105];
    int dfs(char c, int i, int j){
        if(i>j) return 0;
        if(i==j) return 1;
        int k = int(c - 'a');
        if(dp[k][i][j] >= 0) return dp[k][i][j];
        if(ref[i] == c) return dp[k][i][j] = dfs(c, i+1, j);
        if(ref[j] == c) return dp[k][i][j] = dfs(c, i, j-1);
        char d = ref[i];
        dp[k][i][j] = dfs(c, i+1, j) + 1;
        for(auto it=pos[d].upper_bound(i); it!=pos[d].end() &&
*it<=j; ++it){
            dp[k][i][j] = min(dp[k][i][j], 1 + dfs(d, i+1, *it
-1) + dfs(c, *it+1, j));
        }
        return dp[k][i][j];
    }
public:
    int strangePrinter(string s) {
        if(s.empty()) return 0;
        for(int i=0;i<s.size();++i) if(!i || s[i]!=s[i-1]) ref
+= s[i];
        n = ref.size();
        memset(dp, -1, sizeof(dp));
```

```
        for(int i=0;i<n;++i) pos[ref[i]].insert(i);
        return dfs('z' + 1, 0, n-1);
    }
};
```

2. 谁能解释一下二维数组怎么做 @doc

- [ ] Investigating the following solution:

```
class Solution {
public:
    int strangePrinter(std::string s) {
        if (s.size() < 2) return s.size();
        std::vector<int> x(256, -1);
        std::vector<int> pre;
        for (int i = 0; i < s.size(); i++) {
            pre.push_back(x[s[i]]);
            x[s[i]] = i;
        }
        std::vector<std::vector<int>> aux(100, std::vector<int
>(100, 0));
        for (int d = 0; d < s.size(); d++) {
            for (int b = 0; b < s.size() - d; b++) {
                if (d == 0) aux[b][b + d] = 1;
                else {
                    int cand = aux[b][b + d - 1] + 1;
                    int t = pre[b + d];
                    while (t >= b) {
                        cand = std::min(cand, aux[b][t] + aux
[t + 1][b + d - 1]);
                        t = pre[t];
                    }
                    aux[b][b + d] = cand;
                }
```

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

445:

1. No reverse, No extra space

```
class Solution {
    ListNode *add(ListNode* l1, int n1, ListNode *l2, int n2,
int&n3){
        if(!l1 || !l2) {
            n3 = 0;
            return NULL;
        }
        if(n1 < n2) {
            swap(n1, n2);
            swap(l1, l2);
        }
        int x = l1->val;
        ListNode * l3 = NULL;
        if(n1 > n2) l3 = add(l1->next, n1-1, l2, n2, n3);
        else{
            l3 = add(l1->next, n1-1, l2->next, n2-1, n3);
            x += l2->val;
        }
        if(n3 == n1) {
            x += l3->val;
            l3 = l3->next;
        }
        if(x < 10){
```

```
            ListNode *ans = new ListNode(x);
            ans->next = l3;
            n3 = n1;
            return ans;
        }
        else{
            ListNode *ans = new ListNode(x/10);
            ans->next = new ListNode(x%10);
            ans->next->next = l3;
            n3 = n1+1;
            return ans;
        }
    }
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        int n1=0, n2=0, n3;
        for(auto p=l1;p;p=p->next, ++n1);
        for(auto p=l2;p;p=p->next, ++n2);
        return add(l1, n1, l2, n2, n3);
    }
};
```