July 27, 2018 题目：133,351,460,676,766,53,399,349,676,566,641,96

133

1. Easy recursion:

```
class Solution {
    unordered_map<UndirectedGraphNode *, UndirectedGraphNode *> mapping;
public:
    UndirectedGraphNode *cloneGraph(UndirectedGraphNode *node) {
        if(!node) return NULL;
        if(mapping.count(node)) return mapping[node];
        mapping[node] = new UndirectedGraphNode(node->label);
        for(auto n: node->neighbors) mapping[node]->neighbors.push_back(cloneGraph(n));
        return mapping[node];
    }
};
```

351

1. dp + DFS

```
 class Solution {
    public int numberOfPatterns(int m, int n) {
        int[][] dp = new int[10][10];
        dp[1][3] = dp[3][1] = 2;
        dp[4][6] = dp[6][4] = 5;
        dp[7][9] = dp[9][7] = 8;
        dp[1][7] = dp[7][1] = 4;
```

```
        dp[2][8] = dp[8][2] = 5;
        dp[3][9] = dp[9][3] = 6;
        dp[1][9] = dp[9][1] = dp[3][7] = dp[7][3] = 5;
        boolean[] visited = new boolean[10];
        int res = 0;
        res += DFS(0, 1, m, n, 1, visited, dp) * 4;
        res += DFS(0, 1, m, n, 2, visited, dp) * 4;
        res += DFS(0, 1, m, n, 5, visited, dp);
        return res;
    }

     public int DFS(int count, int len, int m, int n, int num,
boolean[] visited, int[][] dp) {
         if (len >= m) count++;
         len++;
         if (len > n) return count;
         visited[num] = true;
         for (int i = 1; i <= 9; i++) {
             int j = dp[num][i];
             if (!visited[i] && (visited[j] || j == 0)) {
                 count = DFS(count, len, m, n, i, visited, d
p);
             }
         }
         visited[num] = false;
         return count;
     }
}
```

2. DFS + DP + 无限看错题：(另外我为什么会做这么复杂)

```
class Solution {
    typedef vector<int> vi;
```

```
    typedef pair<int, int> ii;
    int dp[9][10][520], res, M, N;
    vector<ii> d, v;
    void init(){
        for(int x=-2;x<=2;++x){
            if(!x || abs(x)==2) for(int y=-1;y<=1;y+=2) d.push
_back(ii(x, y));
            else for(int y=-2;y<=2;++y) d.push_back(ii(x, y));
        }
        for(int x=-2;x<=2;x+=2) for(int y=-2;y<=2;y+=2) if(x||
y) v.push_back(ii(x, y));
    }
    int dfs(int where, int step, int stat){
        assert(!(1&(stat>>where)));
        if(step == 1) return 1;
        if(dp[where][step][stat] >= 0) return dp[where][step]
[stat];
        int x = where/3, y = where%3;
        dp[where][step][stat] = 0;
        for(ii p: d){
            int x1 = x + p.first, y1 = y + p.second;
            int w = x1*3 + y1;
            if(x1>=0 && x1<3 && y1>=0 && y1<3 && (1&(stat>>w))
==0){
                dp[where][step][stat] += dfs(w, step-1, stat |
(1<<where));
            }
        }
        for(ii p: v){
            int x1 = x + p.first, y1 = y + p.second, x2 = x +
p.first/2, y2 = y + p.second/2;
```

```
            int w1 = x1 *3 + y1, w2 = x2 * 3 + y2;
            if(x1>=0 && x1<3 && y1>=0 && y1<3 && (1&(stat>>w
1))==0 && (1&(stat>>w2))==0){
                dp[where][step][stat] += dfs(w1, step-1, stat
| (1<<where));
            }
        }
        if(step>=M && step<=N) res += dp[where][step][stat];
        return dp[where][step][stat];
    }
public:
    int numberOfPatterns(int m, int n) {
        init();
        memset(dp, -1, sizeof(dp));
        res = 0;
        M = m;
        N = n;
        for(int i=0;i<9;++i) dfs(i, 9, 0);
        if(M <= 1) res += 9;
        return res;
    }
};
```

- [x] ~~Review this one @Zebo L~~

3. 受到1 的启发: @Tongtong X

```
class Solution {
    void dfs(int x, int y, int l, int status, int &ans, const
int&m, const int&n){
        if(l>=m && l<=n) ++ans;
        if(l==n) return;
        for(int i=0;i<3;++i) for(int j=0;j<3;++j) if(!(1&(stat
us>>(i*3+j)))){
```

```
            if((x+i)%2==0 && (y+j)%2==0) {
                int mid = (x+i)/2*3 + (y+j)/2;
                if(1&(status>>mid)) dfs(i, j, l+1, status|(1<<
(i*3+j)), ans, m, n);
            }
            else dfs(i, j, l+1, status|(1<<(i*3+j)), ans, m,
n);
        }
    }
public:
    int numberOfPatterns(int m, int n) {
        int ans1 = 0, ans2 = 0, ans3 = 0;
        dfs(0, 0, 1, 1<<0, ans1, m, n);
        dfs(0, 1, 1, 1<<1, ans2, m, n);
        dfs(1, 1, 1, 1<<4, ans3, m, n);
        return ans1*4 + ans2*4 + ans3;
    }
};
```

460

1. 前几天做过了，用linked list：自己implement要快一些，因为STL里面有很多没必要的functionality。

```
#include<cassert>
struct Node{
    int key;
    int val;
    int freq;
    Node *next, *prev;
    Node(int k=0, int x=0, int f=0): key(k), val(x), freq(f),
next(NULL), prev(NULL) {}
};
```

```
Node *removeNode(Node *p){
    if(p){
        p->prev->next = p->next;
        if(p->next) p->next->prev = p->prev;
    }
    return p;
}

void insertNode(Node *pos, Node *p){
    if(!p) return;
    p->prev = pos;
    p->next = pos->next;
    if(pos->next) pos->next->prev = p;
    pos->next = p;
}


class LFUCache {
    unordered_map<int, Node*> pos;
    int C, cnt;
    Node lead, *last;
public:
    LFUCache(int capacity) {
        C = capacity;
        cnt = 0;
        last = &lead;
    }

    int get(int key) {
```

```
        if(!pos.count(key) || !C) return -1;
        ++pos[key]->freq;
        Node *p = pos[key]->prev;
        while(p->freq <= pos[key]->freq && p!=&lead) p=p->pre
v;
        if(p->next != pos[key]){
            pos[key] = removeNode(pos[key]);
            insertNode(p, pos[key]);
        }
        while(last->next) last = last->next;
        assert(!last->next);
        return pos[key]->val;
    }

    void put(int key, int value) {
        if(!C) return;
        Node *p;
        if(!pos.count(key)){
            if(cnt < C){
                pos[key] = new Node(key, value, 1);
                ++cnt;
            }
            else{
                pos.erase(last->key);
                pos[key] = removeNode(last);
                pos[key]->key = key;
                pos[key]->val = value;
                pos[key]->freq = 1;
                assert((int)pos.size() == C);
                last = pos[key]->prev;
```

```
            }
            p = last;
        }
        else{
            pos[key] = removeNode(pos[key]);
            pos[key]->val = value;
            ++pos[key]->freq;
            p = pos[key]->prev;
        }
        while(p->freq <= pos[key]->freq && p!=&lead) p=p->pre
v;
        if(p->next != pos[key]) insertNode(p, pos[key]);
        while(last->next) last = last->next;
        assert(!last->next);
        assert((int)pos.size() <= C);
    }
};
```

2. Similar as LRU

```
class LFUCache {

    class Node {
        int key;
        int val;
        int counter;
        Node next;
        Node pre;
        public Node(int key, int val) {
            this.key = key;
            this.val = val;
            this.counter = 0;
```

```
            next = null;
            pre = null;
        }
    }
    Node head;
    Node tail;
    int capacity;
    Map<Integer, Node> map;
    public LFUCache(int capacity) {
        this.capacity = capacity;
        head = new Node(-1, -1);
        tail = new Node(-1, -1);
        map = new HashMap<>();
        head.next = tail;
        tail.pre = head;
    }

    public int get(int key) {
        if (map.containsKey(key)) {
            update(key, map.get(key).val);
            return map.get(key).val;
        } else {
            return -1;
        }
    }

    public void update(int key, int value) {
        Node node = map.get(key);
        node.counter++;
        node.val = value;
```

```
        int count = node.counter;
        if (node.next == null) {
            Node cur = head;
            while (cur.next.counter != 0 && cur.next.counter <
= count) {
                cur = cur.next;
            }
            cur.next.pre = node;
            node.next = cur.next;
            cur.next = node;
            node.pre = cur;
        }
        // When the next node is not tail;
        else if (node.next.counter != 0) {
            Node next = node.next;
            remove(node);
            while (next.counter != 0 && next.counter <= count)
{
                next = next.next;
            }
            next.pre.next = node;
            node.pre = next.pre;
            next.pre = node;
            node.next = next;
        }
        map.put(key, node);
    }

    public void remove(Node node) {
        node.pre.next = node.next;
```

```
        node.next.pre = node.pre;
    }

    public void put(int key, int value) {
        if (map.containsKey(key)) {
            update(key, value);
        } else {
            if (capacity == 0) return;

            if (map.size() == capacity) {
                Node node = head.next;
                map.remove(node.key);
                remove(node);
            }
            map.put(key, new Node(key, value));
            update(key, value);
        }
    }
}

/**
 * Your LFUCache object will be instantiated and called as suc
h:
 * LFUCache obj = new LFUCache(capacity);
 * int param_1 = obj.get(key);
 * obj.put(key,value);
 */
```

676

1. Trie

```
class MagicDictionary {
```

```
class TrieNode {
    TrieNode[] children;
    boolean isWord;
    public TrieNode() {
        children = new TrieNode[26];
        isWord = false;
    }
}
/** Initialize your data structure here. */
TrieNode root;
public MagicDictionary() {
    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;
}
/** Build a dictionary through a list of words */
public void buildDict(String[] dict) {
    for (String s : dict) {
        build(s);
    }
```

```
    }

    /** Returns if there is any word in the trie that equals t
o the given word after modifying exactly one character */
    public boolean search(String word) {
        for (int i = 0; i < word.length(); i++) {
            char[] c = word.toCharArray();
            for (char k = 'a'; k <= 'z'; k++) {
                if (c[i] == k) continue;
                char tmp = c[i];
                c[i] = k;
                String str = new String(c);
                if (find(str)) return true;
                c[i] = tmp;
            }
        }
        return false;
    }

    public boolean find(String s) {
        TrieNode node = root;
        for (char c : s.toCharArray()) {
            if (node.children[c - 'a'] == null) return false;
            node = node.children[c - 'a'];
        }
        return node.isWord;
    }
}

/**
```

```
 * Your MagicDictionary object will be instantiated and called
as such:
 * MagicDictionary obj = new MagicDictionary();
 * obj.buildDict(dict);
 * boolean param_2 = obj.search(word);
 */
```

2. 同上：

```
class MagicDictionary {
    #define CI(c) int((c) - 'a')
    struct T{
        bool isW;
        vector<T *> chl;
        T(): isW(false), chl(vector<T*>(26, NULL)) {}
    };
    void insT(T* root, string s){
        for(char c: s){
            if(!root->chl[CI(c)]) root->chl[CI(c)] = new T();
            root = root->chl[CI(c)];
        }
        root->isW = true;
    }
    bool dfs(T *root, int i, bool modified, const string&s){
        if(!root) return false;
        if(i == s.size()) return root->isW && modified;
        for(char c='a'; c<='z'; ++c) if(!modified || (modified
&&c==s[i])){
            if(dfs(root->chl[CI(c)], i+1, modified||(c!=s[i]),
s)) return true;
        }
        return false;
    }
```

```
public:
    /** Initialize your data structure here. */
    T *root;
    MagicDictionary() : root(new T()) {}
    /** Build a dictionary through a list of words */
    void buildDict(vector<string> dict) {
        for(string s:dict) insT(root, s);
    }
    /** Returns if there is any word in the trie that equals t
o the given word after modifying exactly one character */
    bool search(string word) {
        return dfs(this->root, 0, false, word);
    }
};
```

766

1. 居然差点用了dp：

```
class Solution {
public:
    bool isToeplitzMatrix(vector<vector<int>>& matrix) {
        for(int i=1;i<matrix.size();++i) for(int j=1;j<matrix
[0].size();++j) if(matrix[i][j]!=matrix[i-1][j-1]) return fals
e;
        return true;
    }
};
```

53

1. One pass

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
```

```
        int ans = nums[0];
        for(int i=1, tmp=min(0, nums[0]), sum=nums[0]; i<nums.
size(); ++i){
            sum += nums[i];
            ans = max(ans, sum - tmp);
            tmp = min(tmp, sum);
        }
        return ans;
    }
};
```

399

1. 并查集：另类

```
class Solution {
    string findUnit(string x, unordered_map<string, string>& u
nit, unordered_map<string, double>& coef){
        if(x == unit[x]) return x;
        string y = unit[x];
        unit[x] = findUnit(unit[x], unit, coef);
        coef[x] *= coef[y];
        return unit[x];
    }
public:
    vector<double> calcEquation(vector<pair<string, string>> e
quations, vector<double>& values, vector<pair<string, string>>
queries) {
        unordered_map<string, string> unit;
        unordered_map<string, double> coef;
        for(int i=0; i<equations.size(); ++i){
            string x = equations[i].first, y = equations[i].se
cond;
```

```
            if(!unit.count(x) && !unit.count(y)){
                unit[x] = unit[y] = y;
                coef[y] = 1.;
                coef[x] = values[i];
            }
            else if(unit.count(x) && !unit.count(y)){
                unit[y] = unit[x];
                coef[y] = coef[x] / values[i];
            }
            else if(!unit.count(x) && unit.count(y)){
                unit[x] = unit[y];
                coef[x] = coef[y] * values[i];
            }
            else{
                string c = findUnit(x, unit, coef);
                string d = findUnit(y, unit, coef);
                if(c == d) assert( abs(coef[x]/coef[y] - value
s[i]) < 1.E-6 );
                else{
                    unit[c] = d;
                    coef[c] = values[i] * coef[y] / coef[x];
                }
            }
        }
        vector<double> ans;
        for(auto p: queries){
            string x = p.first, y = p.second;
            if(!unit.count(x) || !unit.count(y)) ans.push_back
(-1.);
            else{
```

```
                string c = findUnit(x, unit, coef);
                string d = findUnit(y, unit, coef);
                if(c != d) ans.push_back(-1.);
                else ans.push_back(coef[x] / coef[y]);
            }
        }
        return ans;
    }
};
```

349

1.

```
class Solution {
    public int[] intersection(int[] nums1, int[] nums2) {
        if (nums1 == null || nums2 == null || nums1.length ==
0 || nums2.length == 0) return new int[]{};
        Set<Integer> set = new HashSet<>();
        Set<Integer> n1 = new HashSet<>();
        for (int n : nums1) {
            n1.add(n);
        }

        for (int n : nums2) {
            if (n1.contains(n)) {
                set.add(n);
            }
        }

        int[] res = new int[set.size()];
        int i = 0;
        for (int k : set) {
```

```
            res[i++] = k;
        }
        return res;
    }
}
```

2. 思路同上，有看错题了，（以为有重复的是正确答案）

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        unordered_set<int> s1(nums1.begin(), nums1.end()), s2(nums2.begin(), nums2.end());
        vector<int> ans;
        for(auto n: s1) if(s2.count(n))  ans.push_back(n);
        return ans;
    }
};
```

566

1. C++ Interpretation 是从右向左

```
class Solution {
public:
    vector<vector<int>> matrixReshape(vector<vector<int>>& nums, int r, int c) {
        if(nums.size() * nums[0].size() != r * c) return nums;
        vector<vector<int>> ans(r, vector<int>(c, 0));
        int j = 0;
        for(auto vec: nums) for(int k: vec) ans[j++/c][j%c] = k;
        return ans;
    }
```

```
};
```

641

1. 直接implement 就行了

```
class MyCircularDeque {
    int size, capa, i, j;
    vector<int> cont;
    inline int getIdx(int x){ return ((x%capa) + capa) % capa;
}
public:
    MyCircularDeque(int k): size(0), capa(k), cont(vector<int>
(k, 0)), i(0), j(0){}
    bool insertFront(int value) {
        if(size == capa) return false;
        ++size;
        --i;
        cont[getIdx(i)] = value;
        return true;
    }
    bool insertLast(int value) {
        if(size == capa) return false;
        ++size;
        cont[getIdx(j)] = value;
        ++j;
        return true;
    }
    bool deleteFront() {
        if(!size) return false;
        --size;
        ++i;
        return true;
```

```
    }
    bool deleteLast() {
        if(!size) return false;
        --size;
        --j;
        return true;
    }
    int getFront() {
        if(!size) return -1;
        return cont[getIdx(i)];
    }
    int getRear() {
        if(!size) return -1;
        return cont[getIdx(j-1)];
    }
    bool isEmpty() { return !size; }
    bool isFull() { return size == capa; }
};
```

96

1. No 难度 dp:

```
class Solution {
    vector<int> dp;
    int dfs(int n){
        if(n<=1) return 1;
        if(dp[n]) return dp[n];
        for(int j=0; j<n; ++j) dp[n] += dfs(j)*dfs(n-j-1);
        return dp[n];
    }
public:
```

```
    int numTrees(int n) {
        if(!n) return 0;
        dp = vector<int>(n+1, 0);
        return dfs(n);
    }
};
```