September 15, 2018 题目：813,485,817,373,651,656,375,434,440,74,215,328,794,154,210

813

1. DFS+ memo :

```
class Solution {
    typedef vector<double> vd;
    vector<vector<vd>> dp;
    double dfs(int i, int j, int k, vector<double>&P){
        if(k > j-i || j<=i) return -10000.0;
        if(k == j-i) return P[j] - P[i];
        if(k==1) return (P[j] - P[i]) / double(j-i);
        if(dp[i][j][k] >= 0) return dp[i][j][k];
        dp[i][j][k] = 0.0;
        for(int z=i+1; z+k-1<=j; ++z){
            dp[i][j][k] = max(dp[i][j][k], (P[z]-P[i])/double
(z-i) + dfs(z, j, k-1, P));
        }
        return dp[i][j][k];
    }
public:
    double largestSumOfAverages(vector<int>& A, int K) {
        int n = A.size();
        vd P(n+1, 0.);
        for(int i=0; i<n; ++i) P[i+1] = P[i] + double(A[i]);
        dp = vector<vector<vd>>(n+1, vector<vd>(n+1, vd(n+1, -
1.)));
        return dfs(0, n, K, P);
    }
};
```
2. 直接dp比memo快好多：
```
class Solution {
    typedef vector<double> vd;
public:
    double largestSumOfAverages(vector<int>& A, int K) {
        int n = A.size();
        vd P(n+1, 0.), res(n+1, 0.);
        for(int i=0; i<n; ++i) {
            P[i+1] = P[i] + double(A[i]);
            res[i+1] = P[i+1]/(i+1);
        }
        for(int k=2; k<=K; ++k){
            vd tmp(n+1, 0.);
            for(int i=k; i<=n; ++i){
                for(int j=k-1; j<i; ++j) tmp[i] = max(tmp[i],
res[j] + (P[i]-P[j])/double(i-j));
            }
            swap(res, tmp);
        }
        return res[n];
    }
};
```
3. 理解起来好烦。。两种做法感觉DP更易懂
```
class Solution {
    public double largestSumOfAverages(int[] A, int K) {
        double[][] dp = new double[A.length][K + 1];
        int[] sum = new int[A.length + 1];
        sum[0] = 0;
        sum[1] = A[0];
        for (int i = 1; i < A.length; i++) {
            sum[i + 1] = sum[i] + A[i];
        }
        for (int i = 1; i <= K; i++) {
            for (int s = 0; s + i <= A.length; s++) {
                if (i == 1) {
                    dp[s][i] = (double) (sum[A.length] - sum
[s]) / (A.length - s);
                    continue;
                }
                for (int e = s; e + i <= A.length; e++) {
                    dp[s][i] = Math.max(dp[s][i], dp[e + 1][i
- 1] + (double) (sum[e + 1] - sum[s]) / (e - s + 1));
                }


            }
        }
        return dp[0][K];
        /*return helper(memo, sum, K, A.length, 0);


    }


    public double helper(double[][] memo, int[] sum, int K, in
t len, int start) {
        if (memo[start][K] > 0) {
            return memo[start][K];
        }
        if (K == 1) {
            memo[start][K] = (double) (sum[len] - sum[start])
/ (len - start);
```

```
            return memo[start][K];
        }

        for (int i = start; i + K <= len; i++) {
            memo[start][K] = Math.max(memo[start][K], (double)
(sum[i + 1] - sum[start]) / (i - start + 1) + helper(memo, su
m, K - 1, len, i + 1));
        }
        return memo[start][K];*/
    }
}
```

485

1. Short is Beauty 系列：

```
class Solution(object):
    def findMaxConsecutiveOnes(self, nums):
        return max([len(s) for s in ''.join([str(x) for x in n
ums]).split('0')])
```

817

1. One pass:

```
class Solution {
public:
    int numComponents(ListNode* head, vector<int>& G) {
        unordered_set<int> S(G.begin(), G.end());
        int ans = 0;
        ListNode lead(0);
        lead.next = head;
        for(auto p=&lead; p->next; p=p->next){
            if(S.count(p->next->val) && (p==&lead || !S.count
(p->val))) ++ans;
        }
        return ans;
    }
};
```

373

1. 用一个set 一直maintain最小即可：

```
class Solution {
    typedef pair<int, int> ii;
public:
    vector<pair<int, int>> kSmallestPairs(vector<int>& nums1, 
vector<int>& nums2, int k) {
        vector<pair<int, int>> ans;
        if(nums1.empty() || nums2.empty()) return ans;
        set<ii> Q;
        int n = nums1.size(), m = nums2.size();
        Q.insert(ii(nums1[0] + nums2[0], 0));
        while(k && !Q.empty()){
            int i = Q.begin()->second/m, j = Q.begin()->second
%m;
            ans.push_back(ii(nums1[i], nums2[j]));
            Q.erase(Q.begin());
            if(i+1 < n) Q.insert(ii(nums1[i+1]+nums2[j], (i+1)
*m + j));
            if(j+1 < m) Q.insert(ii(nums1[i]+nums2[j+1], i*m +
j + 1));
            --k;
        }
        return ans;
    }
};
```

2. heap

```
class Solution {
    public List<int[]> kSmallestPairs(int[] nums1, int[] nums2, int k) {
        List<int[]> list = new ArrayList<>();

        if (nums1 == null || nums2 == null || nums1.length == 0 || nums2.length == 0 || k == 0) {
            return list;
        }
        PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> (a[0] + a[1]) - (b[0] + b[1]));

        for (int i : nums1) {
            for (int j : nums2) {
                pq.offer(new int[]{i, j});
            }
        }
        while (k-- > 0 && !pq.isEmpty()) {
            list.add(pq.poll());
        }
        return list;
    }
}
```

651

1. 为什么做起来这么麻烦：

```
class Solution {
    typedef pair<int, int> ii;
    typedef pair<int, ii> iii;
public:
    int maxA(int N) {
        vector<set<ii>> F(55, set<ii>());
        F[1].insert(ii(1, 0));
        priority_queue<iii, vector<iii>, greater<iii>> Q;
        Q.push(iii(1, ii(1, 0)));
        int res = 0;
        while(!Q.empty()){
            ii stat = Q.top().second;
            int step = Q.top().first;
            Q.pop();
            if(step>N) return res;
            int x = stat.first, y = stat.second;
            res = max(res, x);
            vector<ii> candidates = {ii(x+1, y), ii(x+y, y), i
i(2*x, x)};
            vector<int> delta = {1, 1, 3};
            for(int i=0; i<3; ++i){
                ii s = candidates[i];
                int st = step + delta[i];
                auto it0 = F[st].lower_bound(s);
                if(it0==F[st].end() || s.second > it0->second)
{
                    for(auto it1=F[st].begin(); it1!=it0; ){
                        if(it1->second < s.second) it1 = F[s
t].erase(it1);
                        else ++it1;
                    }
                    F[st].insert(s);
                    Q.push(iii(st, s));
                }
            }
        }
        return res;
    }
};
```

2. 原来应该这么做啊，自己之前实在太蠢了：

```
class Solution {
public:
    int maxA(int N) {
        if(N <= 6) return N;
        vector<int> dp(N+1, 0);
        for(int i=1; i<=6; ++i) dp[i] = i;
        for(int i=7; i<=N; ++i){
            dp[i] = dp[i-1] + 1;
            for(int j=1; j<i-2; ++j) dp[i] = max(dp[i], dp[j]
* (i-j-1));
        }
        return dp[N];
    }
};
```

3. 一个月前自己的做法：

```
class Solution {
public:
    int maxA(int N) {
        vector<int> dp(N+1, 0);
        for(int i=1;i<=N;++i) dp[i] = i;
        for(int i=1;i<=N;++i){
            for(int j=i+3, cnt=2; j<=N; ++j,++cnt) dp[j]=max(d
p[j], cnt*dp[i]);
        }
        return dp[N];
    }
};
```

656

1. 求Lex order最小要从后往前推：

```
class Solution {
    typedef vector<int> vi;
    const int inf = 1E8;
public:
    vector<int> cheapestJump(vector<int>& A, int B) {
        int n = A.size();
        if(A[0]==-1 || A[n-1]==-1) return {};
        vi dp(n, inf), next(n, n);
        dp[n-1] = 0;
        for(int i=n-2; i>=0; --i) if(A[i]!=-1) {
            bool reachable = false;
            for(int j=i+1; j<=min(i+B, n-1); ++j) if(A[j]!=-1)
{
                reachable = true;
                if(dp[j] + A[i] < dp[i]){
                    dp[i] = dp[j] + A[i];
                    next[i] = j;
                }
            }
            if(!reachable) return {};
        }
        vi ans;
        for(int j=0; j<n; j=next[j]){
            ans.push_back(j+1);
        }
        return ans;
    }
};
```

2. O（n）但ms 更慢：

```
class Solution {
    typedef vector<int> vi;
    const int inf = 1E8;
public:
    vector<int> cheapestJump(vector<int>& A, int B) {
        int n = A.size();
        if(A[0]==-1 || A[n-1]==-1) return {};
        vi dp(n, inf), next(n, n), ans;
        deque<int> idx;
        dp[n-1] = 0;
        idx.push_back(n-1);
        for(int i=n-2; i>=0; --i){
            while(!idx.empty() && idx.front() > i+B) idx.pop_f
ront();
            if(idx.empty()) return {};
            if(A[i] != -1){
                next[i] = idx.front();
                dp[i] = dp[next[i]] + A[i];
                while(!idx.empty() && dp[idx.back()]>=dp[i]) i
dx.pop_back();
                idx.push_back(i);
            }
        }
        for(int j=0; j<n; j=next[j]) ans.push_back(j+1);
        return ans;
    }
};
```

375

1. 写代码能力退步了：

```
class Solution {
    const int inf = 1E8;
public:
    int getMoneyAmount(int n) {
        vector<vector<int>> dp(n+1, vector<int>(n+2, 0));
        for(int l=1; l<n; ++l){
            for(int i=1; i+l<=n; ++i){
                dp[i][i+l] = min(i+dp[i+1][i+l], i+l+dp[i][i+l
-1]);
                for(int j=i+1; j<i+l; ++j){
                    dp[i][i+l] = min(dp[i][i+l], j + max(dp[i]
[j-1], dp[j+1][i+l]));
                }
            }
        }
        return dp[1][n];
    }
};
```

2. 感觉dp还是不太熟悉:(

```
class Solution {
    public int getMoneyAmount(int n) {
        int[][] dp = new int[n + 1][n + 1];
        for (int i = 2; i <= n; i++) {
            for (int j = i - 1; j > 0; j--) {
                int max = Integer.MAX_VALUE;
                for (int k = j + 1; k < i; k++) {
                    int local = k + Math.max(dp[j][k - 1], dp
[k + 1][i]);
                    max = Math.min(max, local);
                }
                dp[j][i] = j + 1 == i ? j : max;
            }
        }
        return dp[1][n];
        /**return helper(0, n, new int[n + 1][n + 1]);
    }

    private int helper(int s, int e, int[][] memo) {
        if (s >= e) {
            return 0;
        }
        if (memo[s][e] != 0) {
            return memo[s][e];
        }
        int max = Integer.MAX_VALUE;
        for (int i = s; i <= e; i++) {
            int tmp = i + Math.max(helper(s, i - 1, memo), hel
per(i + 1, e, memo));
            max = Math.min(max, tmp);
        }
        memo[s][e] = max;
        return max;**/
    }
}
```

434

1. Short is Beauty 系列：

```
class Solution(object):
    def countSegments(self, s):
        return len([x for x in s.split(' ') if x])
```

440

1. 这种题一遍bug free简直难于上青天：

```
class Solution {
    int stoi_(string s){
        auto i = s.find_first_not_of('0');
        if(i == string::npos) return 0;
        return stoi(s.substr(i));
    }
    int getCnt(const string&prefix, const string&n){
        if(prefix[0]=='0') return 0;
        int l = prefix.size(), m = n.size(), ans = 0, base=1;
        if(m < l) return 0;
        if(l==m){
            if(prefix <= n) return 1;
            else return 0;
        }
        for(int i=0; i<m-l; ++i){
            ans += base;
            base *= 10;
        }
        if(n.substr(0, l) < prefix) return ans;
        if(n.substr(0, l) == prefix) return ans + stoi_(n.subs
tr(l)) + 1;
        return ans + base;
    }
public:
    int findKthNumber(int n, int k) {
        string s = to_string(n), prefix = "";
        while(k){
            --k;
            for(char c='0'; c<='9'; ++c){
                int cnt = getCnt(prefix + c, s);
                if(k >= cnt) k-=cnt;
                else{
                    prefix += c;
                    break;
                }
            }
        }
        return stoi_(prefix);
    }
};
```

2. 哈哈哈哈哈我用了个collection sort结果超时了

74

1. Bisection search:

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& M, int tar) {
        if(M.empty() || M[0].empty()) return false;
        int n = M.size(), m = M[0].size();
        if(tar > M[n-1][m-1]) return false;
        int l = -1, r = n*m-1;
        while(l < r-1){
            int c = (l+r)/2;
            if(M[c/m][c%m] >= tar) r = c;
            else l = c;
        }
        return M[r/m][r%m] == tar;
    }
};
```

2. binary search

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if (matrix.length == 0 || matrix == null || matrix[0].
length == 0) {
            return false;
        }
        int row = 0;
        int col = 0;
        int m = matrix.length, n = matrix[0].length;

        while (row < m && target > matrix[row][n - 1]) {
            row++;
        }
        if (row == m) {
            return false;
        }
        int i = 0, j = n;
        while (i <= j) {
            int mid = i + (j - i) / 2;
            if (target == matrix[row][mid]) {
                return true;
            }
            if (target > matrix[row][mid]) {
                i = mid + 1;
            } else {
                j = mid - 1;
            }
        }
        return false;
    }
}
```

215

1. Quick Select:

```
class Solution {
    void shaffle(int l, int r, vector<int>&nums){
        for(int i=l; i<r-1; ++i){
            int idx = rand()%(r - i - 1);
            swap(nums[i], nums[i+1+idx]);
        }
    }
    int getKth(int l, int r, int k, vector<int>&nums){
        shaffle(l, r, nums);
        assert(k < r-l);
        if(l == r-1) return nums[l];
        int pivot = nums[l], i=l, j= r-1;
        while(i<j){
            while(j>i && nums[j]<=pivot) --j;
            if(i<j) nums[i++] = nums[j];
            while(i<j && nums[i]>=pivot) ++i;
            if(i<j) nums[j--] = nums[i];
        }
        nums[i] = pivot;
        if(i-l == k) return pivot;
        if(i-l > k) return getKth(l, i, k, nums);
        return getKth(i+1, r, k-i+l-1, nums);
    }
public:
    int findKthLargest(vector<int>& nums, int k) {
        int n = nums.size();
        return getKth(0, n, k-1, nums);
    }
};
```

328

1. 就是On pass：

```
class Solution {
public:
    ListNode* oddEvenList(ListNode* head) {
        if(!head || !head->next) return head;
        auto p1 = head, p2 = head->next, p0 = head->next;
        while(p2 && p2->next){
            p1->next = p2->next;
            p1 = p2->next;
            p2->next = p1->next;
            p2 = p2->next;
        }
        p1->next = p0;
        return head;
    }
};
```

794

1. 一定要细心，注意先后手：
    - 一方取胜的时刻必定是该玩家刚刚下完。

```
class Solution {
    bool win(char c, vector<string>&B, int &cnt){
        cnt = 0;
        for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) cnt += 
(B[i][j] == c);
```

```
        for(int i=0; i<3; ++i) if(B[i][0]==c && B[i][1]==c &&
B[i][2]==c) return true;
        for(int i=0; i<3; ++i) if(B[0][i]==c && B[1][i]==c &&
B[2][i]==c) return true;
        if(B[0][0]==c && B[1][1]==c && B[2][2]==c) return tru
e;
        if(B[2][0]==c && B[1][1]==c && B[0][2]==c) return tru
e;
        return false;
    }
public:
    bool validTicTacToe(vector<string>& board) {
        int cnto, cntx;
        bool o = win('O', board, cnto), x = win('X', board, cn
tx);
        if(o && x) return false;
        if(cnto != cntx && cnto != cntx-1) return false;
        if(o && cnto!=cntx) return false;
        if(x && cnto!=cntx-1) return false;
        return true;
    }
};
```

154

1. Half Bisection search:

```
class Solution {
    int findM(int l, int r, vector<int>& nums){
        if(l>=r-1) return min(nums[l], nums[r]);
        if(nums[l] < nums[r]) return nums[l];
        if(nums[l] > nums[r]){
            int i = l, j = r;
            while(i<j-1){
                int c = (i+j)/2;
                if(nums[c]<nums[l]) j = c;
                else i = c;
            }
            return nums[j];
        }
        int c = (l + r) /2;
        if(nums[c] < nums[l]) return findM(l, c, nums);
        else if(nums[c] > nums[l]) return findM(c, r, nums);
        return min(findM(l, c, nums), findM(c, r, nums));
    }
public:
    int findMin(vector<int>& nums) {
        int n = nums.size();
        return findM(0, n-1, nums);
    }
};
```

210

1. 拓扑排序：

```
class Solution {
public:
    vector<int> findOrder(int n, vector<pair<int, int>>& prere
quisites) {
        vector<vector<int>> E(n);
        vector<int> in(n, 0), ans;
        for(auto v: prerequisites){
            E[v.second].push_back(v.first);
            ++in[v.first];
        }
        unordered_set<int> rest, zeros;
        for(int i=0; i<n; ++i) {
            rest.insert(i);
            if(!in[i]) zeros.insert(i);
        }
        while(!rest.empty()){
            if(zeros.empty()) return {};
            unordered_set<int> tmp;
            for(int k: zeros) {
                for(int j: E[k]){
                    --in[j];
                    if(!in[j]) tmp.insert(j);
                }
                rest.erase(k);
                ans.push_back(k);
            }
            swap(zeros, tmp);
        }
        return ans;
    }
};
```