August 2, 2018 题目：274,777,475,212,469,761,644,720,413,737,401,585,83

274

1. binary search O(nlogn) time, O(1) space, this would work when the indices were sorted.

```
class Solution {
    public int hIndex(int[] citations) {
        if (citations == null || citations.length == 0) {
            return 0;
        }
        int start = 0;
        int end = citations.length - 1;
        Arrays.sort(citations);

        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            int num = citations.length - mid;
            if (citations[mid] < num) {
                start = mid;
            } else if (citations[mid] >= num) {
                end = mid;
            }
        }
        if (citations[start] >= citations.length - start) {
            return citations.length - start;
        }
        if (citations[end] >= citations.length - end) {
            return citations.length - end;
        }

        return 0;
    }
}
```
2. bucket sort, O(n) time, O(n) space. Need to understand the problems already.
```
class Solution {
    public int hIndex(int[] citations) {
        if (citations == null || citations.length == 0) {
            return 0;
        }
        int n = citations.length;
        int[] buckets = new int[n + 1];
        for (int c : citations) {
            if (c >= n) {
                buckets[n]++;
            } else {
                buckets[c]++;
            }
        }

        int sum = 0;
        for (int i = n; i >= 0; i--) {
            sum += buckets[i];
            if (sum >= i) {
                return i;
            }
        }
        return 0;
    }
}
```

3. Bisection Search:

```
class Solution {
public:
    int hIndex(vector<int>& citations) {
        if(citations.empty()) return 0;
        sort(citations.begin(), citations.end());
        int n = citations.size(), l=-1, r=citations.size()+1;
        while(l<r-1){
            int c = (l+r)/2;
            int i = -1, j = n;
            while(i<j-1){
                int k = (i+j)/2;
                if(citations[k]>=c) j=k;
                else i=k;
            }
            if(n-j<c) r=c;
            else l=c;
        }
        return l;
    }
};
```

777

1. 昨天刚错过：就是标记`L` 跟`R`分别在两个string中的位置。

```
class Solution {
public:
    bool canTransform(string start, string end) {
        if(start.size() != end.size()) return false;
        vector<int> S, E;
        for(int i=0; i<start.size(); ++i){
            if(start[i]!='X') S.push_back((start[i]=='L'?1:-1)
* (i+1));
            if(end[i]!='X') E.push_back((end[i]=='L'?1:-1) *
(i+1));
        }
        if(S.size() != E.size()) return false;
        for(int i=0; i<S.size(); ++i){
            if(S[i]*E[i] < 0) return false;
            if(S[i]<E[i]) return false;
            if(i<S.size()-1 && S[i]<0 && S[i+1]>0 && abs(E[i])
>=S[i+1]) return false;
        }
        return true;
    }
};
```

475

1. 找出与每个房子最近的heater:

```
class Solution {
public:
    int findRadius(vector<int>& houses, vector<int>& heaters)
{
        sort(houses.begin(), houses.end());
        sort(heaters.begin(), heaters.end());
        int j = 0, dis = 0;
        for(auto k: houses){
            while(j < heaters.size()-1 && heaters[j+1]<k) ++j;
            if(heaters[j] >= k) dis=max(dis, heaters[j] - k);
            else if(j==heaters.size()-1) dis=max(dis, k-heater
s[j]);
```

```
            else dis=max(dis, min(k-heaters[j], heaters[j+1]-
k));
        }
        return dis;
    }
};
```

212

1. 标准的 DFS + Tier:

```
class Solution {
    #define CI(c) int((c) - 'a')
    #define IC(i) char((i) + 'a')
    struct T{
        string W;
        vector<T*> C;
        T(): W(""), C(vector<T*>(26, NULL)) {}
    };
    void insertT(T *root, string s){
        for(char c: s){
            if(!root->C[CI(c)]) root->C[CI(c)] = new T();
            root = root->C[CI(c)];
        }
        root->W = s;
    }
    int d[4] = {1, -1, 0, 0}, n, m;
    vector<vector<bool>> mark;
    set<string> res;
    void dfs(int i, int j, T*root, vector<vector<char>>& B){
        root = root->C[CI(B[i][j])];
        if(!root->W.empty()) res.insert(root->W);
        mark[i][j] = true;
        for(int k=0; k<4; ++k){
            int x=i+d[k], y=j+d[3-k];
            if(x>=0 && x<n && y>=0 && y<m && !mark[x][y] && root->C[CI(B[x][y])]){
                dfs(x, y, root, B);
            }
        }
        mark[i][j] = false;
    }
public:
    vector<string> findWords(vector<vector<char>>& board, vector<string>& words) {
        if(board.empty() || board[0].empty() || words.empty()) return vector<string>();
        T *root = new T();
        for(string s: words) insertT(root, s);
        n = board.size();
        m = board[0].size();
        mark = vector<vector<bool>>(n, vector<bool>(m, false));
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(root->C[CI(board[i][j])])dfs(i, j, root, board);
        return vector<string>(res.begin(), res.end());
    }
};
```

469

1. 一个小错误 Debug 半天，注意要包括顺时针跟逆时针：

```
class Solution {
    int n;
    bool isC(vector<vector<int>>& points){
        for(int i=0; i<n; ++i){
            int x1 = points[(i+1)%n][0] - points[i][0], y1 = points[(i+1)%n][1] - points[i][1];
            int x2 = points[(i+2)%n][0] - points[(i+1)%n][0], y2 = points[(i+2)%n][1] - points[(i+1)%n][1];
            if(x1 * y2 < y1 * x2) return false;
        }
        return true;
    }
public:
    bool isConvex(vector<vector<int>>& points) {
        n = points.size();
        if(isC(points)) return true;
        reverse(points.begin(), points.end());
        if(isC(points)) return true;
        return false;
    }
};
```

761

1. 半递归 + 半贪心， 注意细节，很容易错：

```
class Solution {
public:
    string makeLargestSpecial(string S) {
        if(S.empty()) return S;
        int n = S.size();
        int i = 0, j = n, idx = n, cnt = 0;
        for(i=1, cnt=1; i<n && cnt; ++i){
            if(S[i]=='1') ++cnt;
            else --cnt;
        }
        if(i == n) return S.substr(0, 1) + makeLargestSpecial(S.substr(1, n-2)) + S.substr(n-1);
        S = makeLargestSpecial(S.substr(0, i)) + makeLargestSpecial(S.substr(i));
        string ans = "";
        cnt = 0;
        for(int k=n-1; k>=i; --k){
            if(S[k]=='0') ++cnt;
            else --cnt;
            if(!cnt){
                if(S.substr(k, j-k) > ans){
                    ans = S.substr(k, j-k);
                    idx = k;
                }
                j = k;
            }
        }
        if(ans <= S.substr(0, idx)) return S;
        return S.substr(idx, ans.size()) + makeLargestSpecial(S.substr(0, idx) + S.substr(idx + ans.size()));
    }
};
```

- [ ] Check LeetCode Discussion @Zebo L

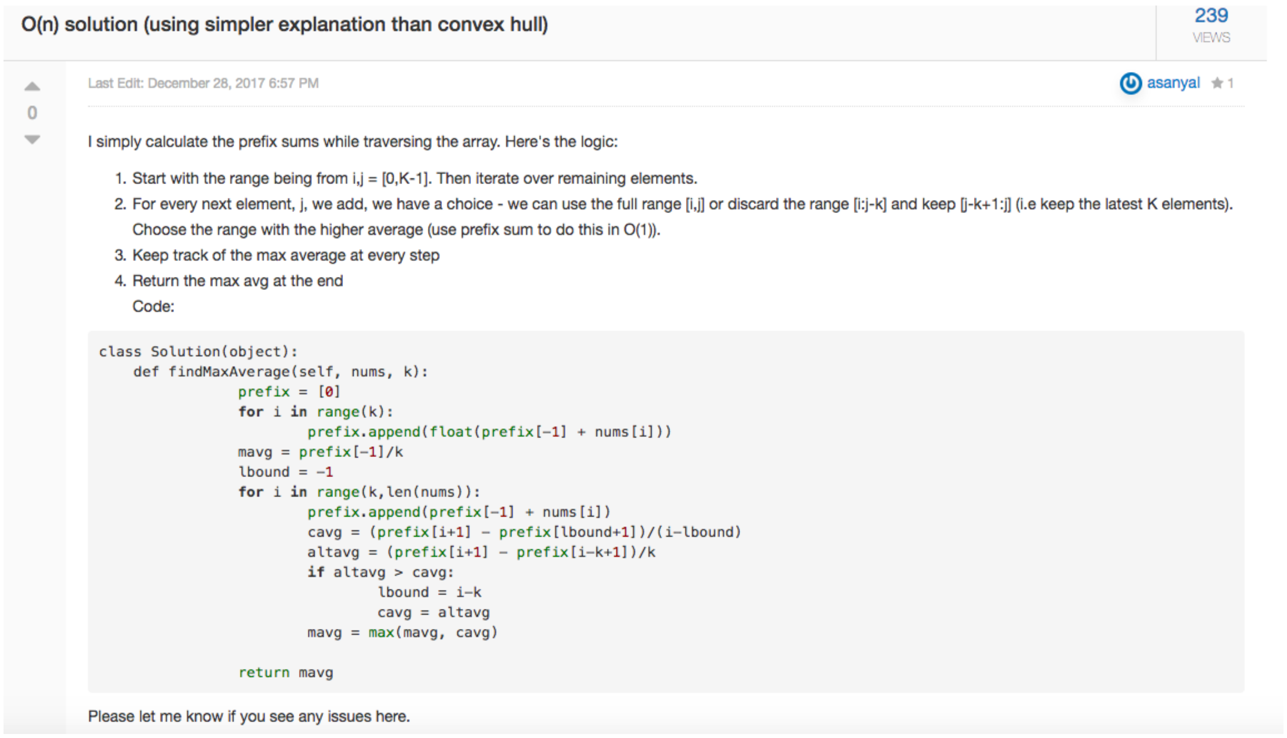
```
https://leetcode.com/problems/special-binary-string/discuss/140400/greedy-Go-0ms

https://leetcode.com/problems/special-binary-string/discuss/113681/7-line-c++

https://leetcode.com/problems/special-binary-string/discuss/114872/O(N2)-greedy-and-asking-for-proof-or-counter-example...
```

644

1. 二分，beat `29.3%`

```
class Solution {
    const double delta = 1.E-7;
    bool ok(vector<int> &A, double avg, int k){
        vector<double> P(A.size()+1, 0.);
        for(int i=1; i<=A.size(); ++i) P[i] = P[i-1] + A[i-1] - avg;
        double m = 0.;
        for(int i=0; i<=A.size()-k; ++i){
            m = min(m, P[i]);
            if(P[i+k] - m >= 0) return true;
        }
        return false;
    }
public:
    double findMaxAverage(vector<int>& nums, int k) {
        double l = -10001, r = 10001;
        while(l < r - delta){
            double c = (l + r)/2.;
            if(ok(nums, c, k)) l = c;
            else r = c;
        }
        return (l+r)/2.;
    }
};
```

- [ ] Investigating the following one @Zebo L


O(n) solution (using simpler explanation than convex hull)

239
VIEWS

0

Last Edit: December 28, 2017 6:57 PM

asanyal ★ 1

I simply calculate the prefix sums while traversing the array. Here's the logic:

1. Start with the range being from i,j = [0,K-1]. Then iterate over remaining elements.
2. For every next element, j, we add, we have a choice - we can use the full range [i,j] or discard the range [i:j-k] and keep [j-k+1:j] (i.e keep the latest K elements). Choose the range with the higher average (use prefix sum to do this in O(1)).
3. Keep track of the max average at every step
4. Return the max avg at the end

   Code:

```
class Solution(object):
    def findMaxAverage(self, nums, k):
                prefix = [0]
                for i in range(k):
                        prefix.append(float(prefix[-1] + nums[i]))
                mavg = prefix[-1]/k
                lbound = -1
                for i in range(k,len(nums)):
                        prefix.append(prefix[-1] + nums[i])
                        cavg = (prefix[i+1] - prefix[lbound+1])/(i-lbound)
                        altavg = (prefix[i+1] - prefix[i-k+1])/k
                        if altavg > cavg:
                                lbound = i-k
                                cavg = altavg
                        mavg = max(mavg, cavg)

                return mavg
```

Please let me know if you see any issues here.

720

1. C++ Functor Usage.

```
class Less{
public:
    bool operator ()(string a, string b) const {
        if(a.size() == b.size()) return a < b;
        return a.size() > b.size();
    }
};
class Solution {
public:
    string longestWord(vector<string>& words) {
        set<string, Less> P(words.begin(), words.end());
        for(string s: P){
            bool ok=true;
            for(int i=s.size()-1; i&&ok; --i) if(!P.count(s.su
bstr(0, i))) ok = false;
```

```
            if(ok) return s;
        }
        return "";
    }
};
```

413

1. Sliding Window

```
class Solution {
public:
    int numberOfArithmeticSlices(vector<int>& A) {
        if(A.size() < 3) return 0;
        int cnt = 1, ans = 0;
        for(int i=1, cur=A[0] - A[1]; i<A.size(); ++i){
            if(A[i] - A[i-1] == cur) ++cnt;
            else{
                ans += (cnt-2) * (cnt-1) / 2;
                cnt = 2;
                cur = A[i] - A[i-1];
            }
        }
        return ans + (cnt-2) * (cnt-1) / 2;
    }
};
```

737

1. 题意没说清楚。

```
class Solution {
public:
    bool areSentencesSimilarTwo(vector<string>& words1, vector
<string>& words2, vector<pair<string, string>> pairs) {
        if(words1.size() != words2.size()) return false;
        unordered_map<string, int> W1, W2;
        for(string s: words1) ++W1[s];
        for(string s: words2) ++W2[s];
        unordered_map<string, vector<string>> E;
        for(auto p: pairs){
            E[p.first].push_back(p.second);
            E[p.second].push_back(p.first);
        }
        while(!W1.empty()){
            unordered_set<string> RM{W1.begin()->first};
            queue<string> Q;
            Q.push(W1.begin()->first);
            while(!Q.empty()){
                string s = Q.front();
                Q.pop();
                for(string k: E[s]) if(!RM.count(k)){
                    RM.insert(k);
                    Q.push(k);
                }
            }
            int cnt1 = 0, cnt2 = 0;
            for(auto it=W1.begin(); it!=W1.end(); ){
                if(RM.count(it->first)) {
                    cnt1 += it->second;
                    it = W1.erase(it);
                }
                else ++it;
            }
            for(auto it=W2.begin(); it!=W2.end(); ){
                if(RM.count(it->first)) {
                    cnt2 += it->second;
                    it = W2.erase(it);
                }
                else ++it;
            }
            if(cnt1 != cnt2) return false;
        }
        return W2.empty();
    }
};
```

401

1. Easy bit manipulation:

```
class Solution {
public:
    vector<string> readBinaryWatch(int num) {
        vector<string> ans;
        for(int k=0; k<64 * 16; ++k){
            int n = k, cnt = 0;
            while(n){
                ++cnt;
                n &= (n-1);
            }
            if(cnt == num && k%64 < 60 && k/64 < 12){
                ans.push_back(to_string(k/64) + ":" + (k%64<1
0?"0"+to_string(k%64):to_string(k%64)));
            }
        }
        return ans;
    }
};
```

585

SQL

83

1. 一次删就行了

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(!head || !head->next) return head;
        for(auto p=head;p->next;){
            if(p->next->val == p->val){
                auto tmp = p->next;
                p->next = p->next->next;
                delete tmp;
            }
            else p = p->next;
        }
        return head;
    }
};
```