# August 7, 2018 题目：835,765,140,727,444,582,661,675,128,42,183,534,159

## 835

1. 最直接的Brute Force：`O(n^4)`

```
class Solution {
public:
    int largestOverlap(vector<vector<int>>& A, vector<vector<int>>& B) {
        int n = A.size(), ans = 0;
        for(int k=-n+1; k<n; ++k) for(int l=-n+1; l<n; ++l){
            int tmp = 0;
            for(int i=max(0, -k); i<min(n, n-k); ++i) for(int j=max(0, -l); j<min(n, n-l); ++j){
                if(A[i][j] && B[i+k][j+l]) ++tmp;
            }
            ans = max(tmp, ans);
        }
        return ans;
    }
};
```

2. Bit manipulation: `O(n^3)`

```
class Solution {
public:
    int largestOverlap(vector<vector<int>>& A, vector<vector<int>>& B) {
        int n = A.size(), ans = 0;
        vector<int> a(n, 0), b(n, 0);
        for(int i=0; i<n; ++i) for(int j=0; j<n; ++j) {
```

```
            a[i] |= (A[i][j]<<j);
            b[i] |= (B[i][j]<<j);
        }
        for(int k=-n+1; k<n; ++k) for(int l=0; l<n; ++l){
            int tmp1 = 0, tmp2 = 0;
            for(int i=max(0, -k); i<min(n, n-k); ++i){
                int x1 = (a[i]>>l) & b[i+k], x2 = (b[i+k]>>l)
& a[i];
                while(x1){
                    ++tmp1;
                    x1 &= (x1-1);
                }
                while(x2){
                    ++tmp2;
                    x2 &= (x2-1);
                }
            }
            ans = max(ans, max(tmp1, tmp2));
        }
        return ans;
    }
};
```

## 765

1. UnionFind

```
class Solution {
    Map<Integer, Integer> map;
    int count;
    public int minSwapsCouples(int[] row) {
        map = new HashMap<>();
        count = row.length;
```

```
        for (int i = 0; i < row.length / 2; i++) {
            map.put(i, i);
        }
        for (int i = 0; i < row.length / 2; i++) {
            int u = row[i * 2];
            int v = row[i * 2 + 1];
            union(u / 2, v / 2);
        }
        return row.length - count;
    }

    public void union(int u, int v) {
        int uu = find(u);
        int vv = find(v);
        if (uu != vv) {
            map.put(uu, vv);
            count--;
        }
    }

    public int find(int u) {
        while (u != map.get(u)) {
            u = map.get(u);
        }
        return u;
    }
}
```

2. 抽象代数？找一个环：

```
class Solution {
    int nei(int x){
```

```
        if(x%2) return x-1;
        return x+1;
    }
public:
    int minSwapsCouples(vector<int>& row) {
        int n = row.size()/2, ans = 0;
        vector<int> pos(2*n, 0);
        for(int i=0; i<2*n; ++i) pos[row[i]] = i;
        unordered_set<int> rest;
        for(int i=0; i<n; ++i) rest.insert(i);
        while(!rest.empty()){
            int k = *rest.begin();
            int x = 2*k, cnt = 0;
            while(rest.count(x/2)){
                rest.erase(x/2);
                ++cnt;
                x = row[nei(pos[nei(x)])];
            }
            ans += cnt - 1;
        }
        return ans;
    }
};
```

- [ ] Investigate 1 @Zebo L

## 140

1. DFS, 并且利用 dp 剪枝

```
class Solution {
    #define CI(c) int((c) - 'a')
    struct T{
        bool W;
```

```
        vector<T*> C;
        T(): W(false), C(vector<T*>(26, NULL)) {}
    };
    void inT(T *r, string s){
        for(char c: s){
            if(!r->C[CI(c)]) r->C[CI(c)] = new T();
            r = r->C[CI(c)];
        }
        r->W = true;
    }
    unordered_map<int, bool> dp;
    bool dfs(vector<string>& ans, string cur, T *root, int i,
string &s){
        if(i == s.size()){
            cur = cur.substr(1);
            ans.push_back(cur);
            return true;
        }
        if(dp.count(i) && !dp[i]) return false;
        T *r = root;
        dp[i] = false;
        cur += " ";
        for(int j=i; j<s.size() && r->C[CI(s[j])]; ++j){
            r = r->C[CI(s[j])];
            cur += s[j];
            if(r->W) dp[i] = dfs(ans, cur, root, j+1, s)||dp
[i];
        }
        return dp[i];
    }
```

```
public:
    vector<string> wordBreak(string s, vector<string>& wordDic
t) {
        vector<string> ans;
        if(s.empty()||wordDict.empty()) return ans;
        T *root = new T();
        for(string s: wordDict) inT(root, s);
        dfs(ans, "", root, 0, s);
        return ans;
    }
};
```

## 727

1. 用map做一个跳板：only beat 16%

```
class Solution {
public:
    string minWindow(string S, string T) {
        unordered_map<char, set<int>> pos;
        if(S.size()<T.size()) return "";
        for(int i=0; i<S.size(); ++i) pos[S[i]].insert(i);
        if(!pos.count(T[0])) return "";
        int L = S.size() + 1;
        string cur = S;
        for(int k0: pos[T[0]]){
            int k = k0;
            bool ok = true;
            for(int j=1; j<T.size()&&ok; ++j){
                if(!pos.count(T[j])){
                    ok = false;
                }
                else{
```

```
                    auto it = pos[T[j]].upper_bound(k);
                    if(it == pos[T[j]].end()) ok = false;
                    else k = *it;
                }
            }
            if(!ok) break;
            if(k-k0+1 < L){
                L = k-k0+1;
                cur = S.substr(k0, k-k0+1);
            }
        }
        if(L==S.size()+1) return "";
        return cur;
    }
};
```

- [ ] Look at other people's solution @Zebo L

## 444

1. 已做（拓扑排序）

```
class Solution {
public:
    bool sequenceReconstruction(vector<int>& org, vector<vecto
r<int>>& seqs) {
        int n = org.size(), judge = 0;
        vector<unordered_set<int>> E(n), P(n);
        for(auto vec: seqs) {
            if(vec.empty()) continue;
            judge = 1;
            if(vec[0] > n || vec[0]<0) return false;
            for(int i=1; i<vec.size(); ++i) {
                if(vec[i] > n || vec[i] < 0) return false;
```

```
                E[vec[i]-1].insert(vec[i-1]-1);
                P[vec[i-1]-1].insert(vec[i]-1);
            }
        }
        if(!judge) return false;
        vector<int> root;
        for(int i=0; i<n; ++i) if(E[i].empty()) root.push_back
(i);
        for(int i=0; i<n; ++i){
            if(root.size()!=1 || root[0]!=org[i]-1) return fal
se;
            vector<int> new_root;
            for(int j: P[org[i]-1]) if(E[j].count(org[i]-1)) {
                E[j].erase(org[i]-1);
                if(E[j].empty()) new_root.push_back(j);
            }
            swap(root, new_root);
        }
        return true;
    }
};
```

## 582

1. BFS

```
class Solution {
public:
    vector<int> killProcess(vector<int>& pid, vector<int>& ppi
d, int kill) {
        unordered_map<int, vector<int>> E;
        for(int i=0; i<pid.size(); ++i) E[ppid[i]].push_back(p
id[i]);
```

```
        vector<int> ans{kill};
        queue<int> Q;
        Q.push(kill);
        unordered_set<int> S{kill};
        while(!Q.empty()){
            int k = Q.front();
            Q.pop();
            for(int i: E[k]) if(!S.count(i)){
                ans.push_back(i);
                Q.push(i);
                S.insert(i);
            }
        }
        return ans;
    }
};
```

- [x] ~~Look at other people's solution @Zebo L~~

2. Queue

```
class Solution {
    public List<Integer> killProcess(List<Integer> pid, List<I
nteger> ppid, int kill) {
        Map<Integer, Set<Integer>> map = new HashMap<>();
        for (int i = 0; i < pid.size(); i++) {
            int pidi = pid.get(i);
            int ppidi = ppid.get(i);
            if (ppidi == 0) continue;
            if (!map.containsKey(ppidi)) {
                map.put(ppidi, new HashSet<>());
            }
            map.get(ppidi).add(pidi);
        }
```

```
        Queue<Integer> q = new LinkedList<>();
        q.offer(kill);
        List<Integer> res = new ArrayList<>();
        while (!q.isEmpty()) {
            int cur = q.poll();
            res.add(cur);
            if (!map.containsKey(cur)) continue;
            Set<Integer> tmp = map.get(cur);
            for (int i : tmp) {
                q.offer(i);
            }
        }
        return res;
    }
}
```

## 661

1. One pass :

```
class Solution {
public:
    vector<vector<int>> imageSmoother(vector<vector<int>>& M)
{
        int n = M.size(), m = M[0].size();
        vector<vector<int>> ans(n, vector<int>(m, 0));
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) {
            int cnt = 0, sum = 0;
            for(int k=-1; k<=1; ++k) for(int l=-1; l<=1; ++l){
                int x = i + k, y = j + l;
                if(x>=0 && x<n && y>=0 && y<m){
                    ++cnt;
```

```
                    sum += M[x][y];
                }
            }
            ans[i][j] = sum/cnt;
        }
        return ans;
    }
};
```

## 675

1. Naive Brute Force:

```
class Solution {
    typedef pair<int, int> ii;
    int N, M, d[4]={1, -1, 0, 0};
    int bfs(int s, int e, vector<vector<int>>& F){
        queue<ii> Q;
        unordered_set<int> S{s};
        Q.push(ii(s, 0));
        while(!Q.empty()){
            auto p = Q.front();
            Q.pop();
            if(p.first == e) return p.second;
            int i = p.first/M, j = p.first%M;
            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 && F[x][y] && !
S.count(x*M + y)){
                    Q.push(ii(x*M+y, p.second+1));
                    S.insert(x*M+y);
                }
            }
```

```
        }
        return -1;
    }
public:
    int cutOffTree(vector<vector<int>>& F) {
        if(F.empty() || F[0].empty()) return 0;
        if(!F[0][0]) return -1;
        N = F.size();
        M = F[0].size();
        vector<ii> trees{ii(-1, 0)};
        for(int i=0; i<N; ++i) for(int j=0; j<M; ++j) if(F[i]
[j]>1) trees.push_back(ii(F[i][j], i*M+j));
        sort(trees.begin(), trees.end());
        int ans = 0;
        for(int i=0; i<trees.size()-1; ++i){
            int l = bfs(trees[i].second, trees[i+1].second,
F);
            if(l==-1) return -1;
            ans += l;
        }
        return ans;
    }
};
```

- [ ] Look at other people's solution @Zebo L
  - [ ] Maybe use dp

## 128

1. 区间：

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

```
        unordered_set<int> S(nums.begin(), nums.end());
        int ans = 0;
        while(!S.empty()){
            int k = *S.begin();
            S.erase(S.begin());
            int l = k - 1, r = k + 1;
            while(S.count(l)) S.erase(l--);
            while(S.count(r)) S.erase(r++);
            ans = max(ans, r-l-1);
        }
        return ans;
    }
};
```

## 42

1. Track frontier:

```
class Solution {
public:
    int trap(vector<int>& height) {
        if(height.empty()) return 0;
        int n = height.size(), ans = 0;
        for(int l=0, r=n-1, lh=height[0], rh=height[n-1]; l<r-
1; ){
            if(lh < rh){
                ++l;
                ans += max(0, lh - height[l]);
                lh = max(lh, height[l]);
            }
            else{
                --r;
                ans += max(0, rh - height[r]);
```

```
                rh = max(rh, height[r]);
            }
        }
        return ans;
    }
};
```

- [ ] Think about other method. @Zebo L
  - [ ] There should be a stack solution @Zebo L

## 183

SQL

## 534

没有

## 159

1. Sliding Window:

```
class Solution {
public:
    int lengthOfLongestSubstringTwoDistinct(string s) {
        unordered_map<char, int> cnt;
        int i=0, j=0, n = s.size(), ans = 0;
        while(i<n && j<n){
            while(j<n && cnt.size()<=2) cnt[s[j++]]++;
            if(cnt.size()<=2) return max(ans, j-i);
            ans = max(ans, j-i-1);
            while(cnt.size()>2){
                cnt[s[i]]--;
                if(!cnt[s[i]]) cnt.erase(s[i]);
                ++i;
            }
        }
```

```
        return ans;
    }
};
```