July 16, 2018

403

1. 二维DP，依然很慢

```
class Solution {
public:
    bool canCross(vector<int>& stones) {
        int n = stones.size();
        vector<vector<bool>> dp(n, vector<bool>(n, false));
        for(int j=0;j<n;++j) dp[n-1][j] = true;
        for(int i=n-2; i>=0 ; --i) for(int j=i+1; j<n&&stones[j]<=stones[i]+i+1; ++j) for(int k=0; k<3; ++k){
            if(dp[j][stones[j] - stones[i]])
                dp[i][stones[j] - stones[i]] = dp[i][stones[j] - stones[i] - 1] = dp[i][stones[j] - stones[i] + 1] = true;
        }
        return dp[0][0];
    }
};
```

- [ ] See LeetCode discussion @Zebo L

2. Initial idea was a 1d dp with an arraylist at each position but found using hashmap would help speeding up everything.

59

1. 很直接

```
class Solution {
public:
    vector<vector<int>> generateMatrix(int n) {
        vector<vector<int>> ans(n, vector<int>(n, 0));
        for(int i=0,j=0,k=1; k<=n*n; ++k){
            ans[i][j] = k;
            cout<<i<<' '<<j<<endl;
            if(i<(n+1)/2 && i<=j+1 && i < n-j-1) ++j;
            else if(j>=n/2 && n-j-1<=i && n-j-1<n-i-1) ++i;
            else if(i>=(n+1)/2 && n-i-1<=n-j-1 && n-i-1 < j) -
-j;
            else --i;
        }
        return ans;
    }
};
```
2. Iterative or recursive

325

1. 用map来track之前的sum

```
public int maxSubArrayLen(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap<>();
        int max = 0;
        int total = 0;
        for (int i = 0; i < nums.length; i++) {
            total += nums[i];
            if (total == k) max = i + 1;
            if (map.containsKey(total - k)) max = Math.max(ma
x, i - map.get(total - k));
            else if (!map.containsKey(total)) map.put(total,
i);
    }
        return max;
    }
```
2. 思路同上, 只不过是从后往前，多用了一个数组

```
class Solution {
public:
    int maxSubArrayLen(vector<int>& nums, int k) {
        vector<int> P(nums.size()+1, 0);
        for(int i=0; i<nums.size(); ++i) P[i+1] = P[i] + nums[i];
        int ans = 0;
        unordered_map<int, int> pos;
        for(int j=nums.size(); j>=0; --j){
            if(pos.count(P[j] + k)) ans = max(ans, pos[P[j]+k] - j);
            if(!pos.count(P[j])) pos[P[j]] = j;
        }
        return ans;
    }
};
```

1. #1 is the answer.

193

BASH

610

SQL

129

1. 很直接的dfs，每次遇到`NULL`，或`leaf`就`return`， 如果遇到`leaf`，把该边数的累加到`ans`:

```
class Solution {
    int ans;
    bool isLeaf(TreeNode *r){
        return !r->left && !r->right;
    }
    void dfs(TreeNode *root, int res){
        if(!root) return;
        res = 10 * res + root->val;
        if(isLeaf(root)){
            ans += res;
            return;
        }
        dfs(root->left, res);
        dfs(root->right, res);
    }
public:
    int sumNumbers(TreeNode* root) {
        ans = 0;
        dfs(root, 0);
        return ans;
    }
};
```

2. 好像写的有点复杂，不过思路同上

```
class Solution {
    int sum = 0;
    public int sumNumbers(TreeNode root) {
        if (root == null) return 0;
        helper(root, root.val);
        return sum;
    }
    public void helper(TreeNode root, int cur) {
        if (root.left == null && root.right == null) {
            sum += cur;
            return;
        }
        if (root.left != null) helper(root.left, cur * 10 + ro
ot.left.val);
```

```
        if (root.right != null) helper(root.right, cur * 10 + root.right.val);
    }
}
```

769

1. 思路是一旦当前数比之前的都大，就有机会成为一个chunk。因为sorted array规定了从0 ~ len - 1，那只要当前数与index数相同便可以加一个chunk

```
    public int maxChunkksToSorted(int[] arr) {
        int max = 0;
        int res = 0;
        for (int i = 0; i < arr.length; i++) {
            max = Math.max(max, arr[i]);
            if (max == i) res++;
        }
        return res;
    }
```

2. 就是记录两个数组: `max_arr[i]` : `0 ~ i` 最大的数，`min_arr[j]`: `n-1 ~ j` 最小的数，如果 `max_arr[i] < min_arr[i+1]`时，说明可以从 `i` 点断开。(如果数组不是从 `1 ~ n` 也适用) （PS：楼上思路好nb）：

```
class Solution {
public:
    int maxChunksToSorted(vector<int>& arr) {
        vector<int> min_arr(arr.size(), arr[arr.size()-1]), max_arr(arr.size(), arr[0]);
        int ans = 1;
        for(int i=1;i<arr.size();++i) max_arr[i] = max(arr[i], max_arr[i-1]);
        for(int j=arr.size()-1; j; --j){
            if(max_arr[j-1] == min_arr[j]-1) ++ans;
            min_arr[j-1] = min(min_arr[j], arr[j-1]);
        }
        return ans;
    }
};
```
3. finally a top down way is much easier than a bottom up dfs.

825

1. 用二分法计数：注意`age[B] <= 0.5 * age[A] + 7`的判定

```
class Solution {
public:
    int numFriendRequests(vector<int>& ages) {
        int n = ages.size(), ans = 0;
        unordered_map<int, int> cnt;
        sort(ages.begin(), ages.end());
        for(int i=0; i<n; ++i){
            ++cnt[ages[i]];
            int lower = int(0.5 * double(ages[i]) + 8.3);
            int upper = ages[i] - 1;
            if(ages[i] < 100) upper = min(upper, 100);
            if(upper < lower) continue;
            int l1 = -1, r1 = i;
            while(l1 < r1 - 1){
                int c = (l1 + r1)/2;
                if(ages[c] < lower) l1 = c;
                else r1 = c;
            }
            int l2 = -1, r2 = i;
            while(l2 < r2-1){
                int c = (l2 + r2)/2;
                if(ages[c] > upper) r2 = c;
                else l2 = c;
            }
            ans += l2 - r1 + 1;
        }
        for(auto p:cnt){
            int lower = int(0.5 * double(p.first) + 8.3);
            int upper = p.first;
            if(p.first < 100) upper = min(upper, 100);
            if(p.first >= lower && p.first <= upper) ans += p.
second * (p.second-1);
        }
        return ans;
    }
};
```

2. Sliding window:

```
class Solution {
public:
    int numFriendRequests(vector<int>& ages) {
        int n = ages.size(), ans = 0;
        sort(ages.begin(), ages.end());
        int l = 0, r = 0;
        for(int i=0; i<n; ++i){
            int lower = int(0.5 * double(ages[i]) + 8.3);
            int upper = ages[i];
            if(ages[i] < 100) upper = min(upper, 100);
            if(upper < lower) continue;
            while(l<n && ages[l] < lower) ++l;
            while(r<n && ages[r] <= upper) ++r;
            ans += r-l;
            if(ages[i] >= lower && ages[i] <= upper) --ans;
        }
        return ans;
    }
};
```

354

1. LIS的思路

```
public int maxEnvelopes(int[][] envelopes) {
        Arrays.sort(envelopes, new Comparator<int[]>(){
           public int compare(int[] a, int[] b) {
               return a[0] != b[0] ? a[0] - b[0] : b[1] - a
[1];
           }
        });
        int[] tails = new int[envelopes.length];
        int size = 0;
        for (int[] en : envelopes) {
            int i = 0, j = size;
            while (i != j) {
                int mid = i + (j - i) / 2;
                if (en[1] > tails[mid]) {
                    i = mid + 1;
                } else {
                    j = mid;
                }
            }
            tails[i] = en[1];
            if (i == size) size++;
        }
        return size;
    }
```

2. `O(n^2)` dp:

```
class Solution {
```

```
public:
    int maxEnvelopes(vector<pair<int, int>>& envelopes) {
        int n = envelopes.size(), ans = 0;
        sort(envelopes.begin(), envelopes.end());
        vector<int> dp(n, 1);
        for(int i=0;i<n;++i){
            for(int j=i-1;j>=0;--j) if(envelopes[j].first < en
velopes[i].first && envelopes[j].second < envelopes[i].second)
{
                dp[i] = max(dp[i], dp[j] + 1);
            }
            ans = max(ans, dp[i]);
        }
        return ans;
    }
};
```

3. `O(n log(n))`

```
class Solution {
    typedef pair<int, int> ii;
    static bool cmp(const ii&a, const ii&b){
        if(a.first == b.first) return a.second > b.second;
        return a.first < b.first;
    }
public:
    int maxEnvelopes(vector<pair<int, int>>& envelopes) {
        int n = envelopes.size(), ans = 0;
        sort(envelopes.begin(), envelopes.end(), cmp);
        vector<int> dp;
        for(auto p: envelopes){
            int l = -1, r = dp.size();
            while(l < r - 1){
                int c = (l+r)/2;
                if(dp[c] < p.second) l = c;
                else r = c;
            }
            if(r < dp.size()) dp[r] = p.second;
            else dp.push_back(p.second);
        }
        return dp.size();
    }
};
```

- [ ] Review the above solution @Zebo L

273

1. 比较麻烦，注意单词别拼错就行了

```
#include<cassert>
class Solution {
    map<int, string, greater<int>> ref;
    vector<int> nu{(int)1E9, (int)1E6, (int)1E3, 100, 90, 80, 
70, 60, 50, 40, 30, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 1
0, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
    vector<string> eng{"Billion", "Million", "Thousand", "Hund
red", "Ninety", "Eighty", "Seventy", "Sixty", "Fifty", "Fort
y", "Thirty", "Twenty", "Nineteen", "Eighteen", "Seventeen",
"Sixteen", "Fifteen", "Fourteen", "Thirteen", "Twelve", "Eleve
n", "Ten",
"Nine", "Eight", "Seven", "Six", "Five", "Four", "Three", "Tw
o", "One", "Zero"};
    string n2W(int n){
        int m = ref.lower_bound(n)->first;
        string ans;
        if(n >= 100) ans += n2W(n/m) + " ";
        ans += ref[m];
        if(n % m) ans += " " + n2W(n%m);
        return ans;
    }
public:
    string numberToWords(int num) {
        if(!num) return "Zero";
        assert(nu.size() == eng.size());
        for(int i=0;i<nu.size();++i) ref[nu[i]] = eng[i];
        return n2W(num);
    }
};
```