July 31, 2018 题目：489,293,818,739,397,7,364,705,150,386,777,24,719

489

1. Track coordinates:

```
class Solution {
    typedef pair<int, int> ii;
    set<ii> pool;
    void dfs(int x, int y, int dx, int dy, Robot& robot){
        robot.clean();
        pool.insert(ii(x, y));
        int dx1 = dy, dy1 = -dx;
        robot.turnRight();
        for(int i=0; i<3; ++i){
            if(!pool.count(ii(x+dx1, y+dy1)) && robot.move()){
                dfs(x+dx1, y+dy1, dx1, dy1, robot);
                robot.turnRight();
            }
            else robot.turnLeft();
            int tmp = -dy1;
            dy1 = dx1;
            dx1 = tmp;
        }
        robot.move();
    }
public:
    void cleanRoom(Robot& robot) {
        int x = 0 , y = 0, dx = 0, dy = 1;
```

```
        robot.clean();
        pool.insert(ii(0, 0));
        for(int i=0; i<4; ++i){
            if(!pool.count(ii(x+dx, y+dy)) && robot.move()){
                dfs(x+dx, y+dy, dx, dy, robot);
                robot.turnRight();
            }
            else robot.turnLeft();
            int tmp = -dy;
            dy = dx;
            dx = tmp;
        }
    }
};
```

2. dfs暴力解，慢极了，memo没写好 ---------- 他妹这是486，我看错题了

```
class Solution {
    public boolean PredictTheWinner(int[] nums) {
        if (nums == null || nums.length == 1) {
            return true;
        }
        // int[][] memo = new int[nums.length][nums.length];
        return dfs(nums, 0, nums.length - 1,  0, 0, 1) == 1;
    }
    int dfs(int[] nums, int l, int r,  int p1, int p2, int pla
yer) {

        if (l > r) {
            return p1 >= p2 ? 1 : 2;
        }
        //         System.out.println("l is " + l + ", r is "
+ r + ", p1 is " + p1 + ", p2 is " + p2 + ", player is " + pla
```

```
yer + ", memo is " + memo[l][r]) ;
        // if (memo[l][r] > 0) {
        //     return memo[l][r];
        // }
        if (player == 1) {
            int left = dfs(nums, l + 1, r, p1 + nums[l], p2,
2);
            int right = dfs(nums, l, r - 1, p1 + nums[r], p2,
2);
            if (left == 1 || right == 1) {
                // memo[l][r] = 1;
                return 1;
            }
            // memo[l][r] = 2;
            return 2;
        } else {
            int left = 0;
            int right = 0;
            // if (nums[l] > nums[r]) {
                left = dfs (nums, l + 1, r,  p1, p2 + nums[l],
1);
            // } else {
                right = dfs (nums, l, r - 1, p1, p2 + nums[r],
1);
            // }
            if (left == 2 || right == 2) {
                // memo[l][r] = 2;
                return 2;
            } else {
                // memo[l][r] = 1;
                return 1;
```

```
            }
            // return memo[l][r];
        }

    }
}
```

3. 486 是应该用dp，就像 palindrome一样的dp

```
class Solution {
    public boolean PredictTheWinner(int[] nums) {
        int[][] dp = new int[nums.length][nums.length];
        for (int i = 0; i < nums.length; i++) {
            dp[i][i] = nums[i];
        }
        for (int i = nums.length - 2; i >= 0; i--) {
            for (int j = i + 1; j < nums.length; j++) {
                int left = nums[i] - dp[i + 1][j];
                int right = nums[j] - dp[i][j - 1];
                dp[i][j] = Math.max(left, right);
            }
        }
        return dp[0][nums.length - 1] >= 0;
    }
}
```

293

1. One pass:

```
class Solution {
public:
    vector<string> generatePossibleNextMoves(string s) {
        vector<string> ans;
```

```
        if(s.size()<=1) return ans;
        for(int i=0; i<s.size()-1; ++i){
            if(s[i]=='+' && s[i+1]=='+'){
                s[i] = s[i+1] = '-';
                ans.push_back(s);
                s[i] = s[i+1] = '+';
            }
        }
        return ans;
    }
};
```

2. use some python language sugar lol

```
class Solution:
    def generatePossibleNextMoves(self, s):
        """
        :type s: str
        :rtype: List[str]
        """
        if s is None:
            return []
        res = []
        i = 0
        while i < len(s):
            if (s[i:i+2]== '++'):
                res.append(s[0:i]+'--'+s[i+2:])
            i+=1
        return res
```

818

1. Slow BFS `768 ms`:

```
class Solution {
```

```
    typedef pair<int, int> ii;
    typedef vector<int> vi;
    int sgn(int x){
        return (x>=0?1:-1);
    }
public:
    int racecar(int target) {
        queue<vi> Q;
        set<ii> S;
        Q.push(vi{0, 1, 0});
        S.insert(ii(0, 1));
        while(!Q.empty()){
            auto v = Q.front();
            Q.pop();
            if(v[0] == target) return v[2];
            if(v[0] + v[1] < 2*target && v[0] + v[1] > -1 * ta
rget && !S.count(ii(v[0]+v[1], 2*v[1]))){
                S.insert(ii(v[0]+v[1], 2*v[1]));
                Q.push(vi{v[0]+v[1], 2*v[1], v[2]+1});
            }
            if(!S.count(ii(v[0], -sgn(v[1])))){
                S.insert(ii(v[0], -sgn(v[1])));
                Q.push(vi{v[0], -sgn(v[1]), v[2]+1});
            }
        }
        return -1;
    }
};
```

- [ ] Check other ppl's solution @Zebo L

739

1. Solved using a stack to keep track of previous temperatures, all things in the stack haven't found a temperature hotter than them yet. Because they are removed as soon as a hotter temperature is found, the final temperature is always the coldest in the stack.

```
class Solution:
    def dailyTemperatures(self, temperatures):
        """
        :type temperatures: List[int]
        :rtype: List[int]
        """
        #We implement this algorithm in O(n) using a stack
        #Python lists act as stacks, using append and pop

        stack = []

        #initialise the answer to 0
        answer = [0 for i in range(len(temperatures))]

        for i in range(len(temperatures)):
            #Due to the construction of the stack (and our rem
oval from the stack) the coldest temp on the stack is last.
            if stack != [] and temperatures[i] > stack[-1][1]:
                #When we find a warmer temperature, add everyt
hing colder than the current temp to the answer, removing from
the stack
                while stack != [] and stack[-1][1] < temperatu
res[i]:
                    index = stack.pop()[0]
                    answer[index] = i - index
            #Add a temperature to the stack
            stack.append((i,temperatures[i]))
```

```
        return answer
```

2. 同上：

```
class Solution {
public:
    vector<int> dailyTemperatures(vector<int>& T) {
        stack<int> S;
        vector<int> ans(T.size(), 0);
        for(int i=0; i<T.size(); ++i){
            while(!S.empty() && T[S.top()] < T[i]){
                ans[S.top()] = i - S.top();
                S.pop();
            }
            S.push(i);
        }
        return ans;
    }
};
```

397

1. Greedy: Overflow 很烦

```
class Solution {
public:
    int integerReplacement(int N) {
        int ans = 0;
        long n = long(N);
        while(n>1){
            if(!(n%2)) n/=2;
            else{
                if(n>3){
                    if((n+1)%4 == 0) ++n;
                    else --n;
```

```
                }
                else --n;
            }
            ++ans;
        }
        return ans;
    }
};
```

7

1. Overflow 很烦

```
class Solution {
public:
    int reverse(int X) {
        long ans = 0, sign = (X>=0? 1:-1), x= abs(long(X));
        while(x){
            ans = ans * 10 + x % 10;
            x /= 10;
        }
        ans *= sign;
        if(ans>long(INT_MAX) || ans<long(INT_MIN)) return 0;
        return ans;
    }
};
```

364

1. 正常DFS :

```
class Solution {
    int sum, rsum, depth;
    void dfs(NestedInteger I, int level){
        depth = max(depth, level);
```

```
        if(I.isInteger()){
            rsum += level * I.getInteger();
            sum += I.getInteger();
        }
        else{
            for(auto i: I.getList()){
                dfs(i, level + 1);
            }
        }
    }
public:
    int depthSumInverse(vector<NestedInteger>& nestedList) {
        sum = rsum = depth = 0;
        for(auto I: nestedList){
            dfs(I, 1);
        }
        return (depth + 1) * sum - rsum;
    }
};
```

705

1. 这种大array 都能过。。

```
class MyHashSet {
public:
    /** Initialize your data structure here. */
    vector<bool> A;
    MyHashSet(): A(vector<bool>(1000001, false)) {}
    void add(int key) { A[key] = true; }
    void remove(int key) { A[key] = false; }
    bool contains(int key) { return A[key]; }
};
```

150

1. 原理是stack，但用 `array` 更好写一些，因为要去最末端两个元素进行运算。

```
class Solution {
    int oper(int x, int y, char c){
        if(c=='+') return x+y;
        if(c=='-') return x-y;
        if(c=='*') return x*y;
        return x/y;
    }
public:
    int evalRPN(vector<string>& tokens) {
        vector<int> A(tokens.size(), 0);
        int i = -1;
        for(string s: tokens){
            if(s.size()==1 && (s[0]>'9' || s[0]<'0')){
                --i;
                A[i] = oper(A[i], A[i+1], s[0]);
            }
            else{
                ++i;
                A[i] = stoi(s);
            }
        }
        assert(!i);
        return A[i];
    }
};
```

2. Relatively simple but very slow O(n^2) iterative solution using python lists

```
class Solution:
```

```
    def evalRPN(self, tokens):
        """
        :type tokens: List[str]
        :rtype: int
        """
        #This is an interative solution - each iteration will evauluate only one expression in the RPN string
        #It then updates the tokens list and starts again
        while len(tokens) > 1:
            #We loop through until we find an operator (There must be at least one operator if the RPN is valid)
            i = 0
            while tokens[i] not in '+-*/':
                i += 1

            #evaluate the statement (SECURITY FLAW IF RPN NOT VALID!!!)
            ans = eval(tokens[i-2] + tokens[i] + tokens[i-1])
            ans = int(ans) #In case of division, round towards 0

            #update the tokens list
            tokens = tokens[:i-2] + [str(ans)] + tokens[i+1:]
        return int(tokens[0])
```

386

1. Cannot believe the following `O(n)` approach only beats 6`0%`!

```
class Solution {
public:
    vector<int> lexicalOrder(int n) {
        vector<int> ans;
```

```
        for(int i=1; i;){
            ans.push_back(i);
            if(i * 10 <= n) i*=10;
            else{
                while(i && (i==n || i%10==9)) i/=10;
                if(i) ++i;
            }
        }
        return ans;
    }
};
```

777

1. 记录并比较两个字符串中 `R` 跟 `L` 的位置即可：

```
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;
    }
};
```

24

1. Somewhat of a confusing/inelegant solution, it iterates through the list keeping track of the previous two nodes, swapping when the two nodes are defined and then undefining those nodes. Any tips or an alternate solution would be welcome.

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def swapPairs(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        #We keep track of the current node, and the two previo
us nodes
        cur = head
        prev2 = prev = None

        #We also monitor the new head
        newHead = None

        while cur != None:
            '''
```

```
            i = cur
            print('[', end='')
            while(i != None):
                print(i.val, end = ',')
                i = i.next
            print(']')
            '''
            if (prev2 == None and newHead != None) or prev == None: #This signifies no swap needs to take place
                prev2 = prev
                prev = cur
                cur = cur.next
            else: #We will swap the nodes, then set prev2/prev to None to continue to the next pair
                if newHead == None: #Special case to deal with the first swap
                    tmp = cur.next
                    cur.next = prev
                    prev.next = tmp
                    newHead = cur
                else:
                    tmp = cur.next
                    prev2.next = cur
                    cur.next = prev
                    prev.next = tmp

                cur = prev
                prev2 = prev = None

        if newHead == None: #Edge case when no swaps occurred
            return head
```

```
        return newHead
```

2. 跟楼上差不多把，应该：

```
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {
        ListNode lead(0);
        lead.next = head;
        for(auto p=&lead;p->next && p->next->next; p=p->next->
next){
            auto tmp = p->next->next->next;
            p->next->next->next = p->next;
            p->next = p->next->next;
            p->next->next->next = tmp;
        }
        return lead.next;
    }
};
```

3. recursive solution

```
class Solution {
    public ListNode swapPairs(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }

        ListNode first = head;
        ListNode second = head.next;
        ListNode p = swapPairs(second.next);



        second.next = first;
```

```
        first.next = p;
        return second;
    }
}
```

719

1. 二分：

```
class Solution {
public:
    int smallestDistancePair(vector<int>& nums, int k) {
        int n = nums.size();
        sort(nums.begin(), nums.end());
        int l = 0, r = nums[n-1] - nums[0] + 1;
        while(l < r-1){
            int c = (l+r)/2, cnt = 0;
            for(int i=0; i<n-1;++i){
                int j = i, x = n, target=nums[i]+c;
                while(j<x-1){
                    int m = (x+j)/2;
                    if(nums[m] < target) j=m;
                    else x = m;
                }
                cnt += x - i - 1;
            }
            cout << c << ' '<<cnt<<endl;
            if(cnt < k) l = c;
            else r = c;
        }
        return l;
    }
```

```
};
```

- [ ] Check other people's solution @Zebo L

2. bucket sort. Since the range of the sum is limited, use it as the bucket range. And store all the numbers of distances and sum them. Whenever we met the correct sum, it is the answer. nums[i] < 1000001 is the key. So the difference is less than 1000001.

```
class Solution {
    public int smallestDistancePair(int[] nums, int k) {
        int[] dp = new int[1000001];
        for (int i = 0; i < nums.length - 1; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                int dis = Math.abs(nums[i] - nums[j]);
                dp[dis]++;
            }
        }
        int sum = 0;
        for (int i = 0; i < 1000001; i++) {
            sum += dp[i];
            if (sum >= k) {
                return i;
            }
        }
        return -1;
    }
}
```