General Notes:

Set, map: native log(n) search

Kth: priority queue, binary search.

Palindrome: hash all the strings, compare with reverse. Trie Tree. KMP

Factor: 参照super ugly number, DP

Template:

Kmp:

vector<int> next(l.size(), -1);

int k = -1;

for(int i = 1; i < l.size(); i++) {

while(k >= 0 && l[k + 1] != l[i]) k = next[k];

if (l[k + 1] == l[i]) k++;

next[i] = k;

}

class Solution {

private:

vector<int> makeNext(string s) {

vector<int> next(s.size(), -1);

int k = -1;

for(int j = 1; j < s.size(); j++) {

while (k >= 0 && s[k + 1] != s[j]) k = next[k];

if (s[k + 1] == s[j]) k++;

next[j] = k;

}

return next;

}

public:

int strStr(string haystack, string needle) {

if (needle.empty()) return 0;

if (haystack.empty()) return -1;

int n = haystack.size(), m = needle.size();

vector<int> next = makeNext(needle);

int j = -1;

for(int i = 0; i < n; i++) {

while(j >=0 && haystack[i] != needle[j + 1]) j = next[j];

if (haystack[i] == needle[j + 1]) j++;

if (j == m - 1) return (i - m + 1);

}

return -1;

}

};

251 [Flatten 2D Vector](https://leetcode.com/problems/flatten-2d-vector/)

1)Int x y 存放当前位置, next() 就更新x y的值

2) 用vector<vector<int>>::iterator 和 vector<int>::iterator 判断y到头: y == x->end();

可以再用一个vector<vector<int>>::iterator 而不用cache整个input

272 [Closest Binary Search Tree Value II](https://leetcode.com/problems/closest-binary-search-tree-value-ii/) (H)

1) In order 遍历 维护k priority O(nlog(k))

2) in order 维护list/deque 右进左出 直道不需要进 O(n)

3) BST implement getSuccessor & getPredecessor

两个栈存放successor和predecessor 的path

注意BST和普通tree存parent的方式

331 [Verify Preorder Serialization of a Binary Tree](https://leetcode.com/problems/verify-preorder-serialization-of-a-binary-tree/) (M)

1) Stack 砍叶子 遇到 A # #就变成# 直道最后看是不是只剩#

2) 一个node 产生未来两个# 本身消耗一个#, (#++) 遇到# (#--) 顺序遍历看#够不够

336 [Palindrome Pairs](https://leetcode.com/problems/palindrome-pairs/)

Hash 所有strings, 遍历所有strings, 取逆序 遍历整个长度看左半或右半是否有hash,并且剩余部分isPalindrome 都成立 则push(hash[left], i) or (i, hash[right]).

Edge case: empty string

313 [Super Ugly Number](https://leetcode.com/problems/super-ugly-number/)

参考Ugly Number II

类似DP对于已有res[1..n] 计算res[n + 1]

res[n + 1] 必然是已有的某个res[i] \* 2 或 res[j] \* 3 或 res[k] \* 5 …

取最小的那个, 然后对应index++. 剩下的index 不变, 为未来potential candidate

除非candidate正好等于当前res, 那么skip, index++;

316 [Remove Duplicate Letters](https://leetcode.com/problems/remove-duplicate-letters/)

public **class** **Solution** {

public String removeDuplicateLetters(String s) {

int[] cnt = new int[26];

int pos = 0; // the position **for** the smallest s[i]

**for** (int i = 0; i < s.length(); i++) cnt[s.charAt(i) - 'a']++;

**for** (int i = 0; i < s.length(); i++) {

**if** (s.charAt(i) < s.charAt(pos)) pos = i;

**if** (--cnt[s.charAt(i) - 'a'] == 0) **break**;

}

**return** s.length() == 0 ? "" : s.charAt(pos) + removeDuplicateLetters(s.substring(pos + 1).replaceAll("" + s.charAt(pos), ""));

}

}

找到第一个即将绝种的字母, 那么,在它之前的所有字母在它之后都会至少出现一次,那么,之前有a肯定拿a,因为a后面的字母后面都会有.

对于比它打的字母没必要让它们出现在之前,但是比它小的字母,肯定选在让它们出现在之前而不是这个位置以后.所有小的里面,有a上a有b上b.因为你至少需要在这个位置上选择这个即将绝种的字母,之前有小的肯定要拿.

309 [Best Time to Buy and Sell Stock with Cooldown](https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/)

4种操作 buy sell cd idle

3 dp array represent max profit end with “buy, sell and cd”

buy[0] = -prices[0], sell[0] = 0, cd[0] = 0;

for(int i = 1; i < n; i++) {

buy[i] = max(cd[i - 1] - prices[i], buy[i - 1]);

sell[i] = max(buy[i - 1] + prices[i], sell[i - 1]);

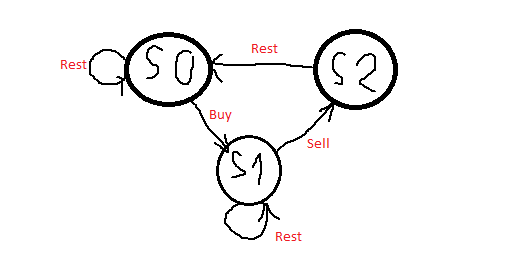
cd[i] = max(cd[i - 1], sell[i - 1]);

}

return max(sell[n - 1], cd[n - 1]);

后续: 转换为3个变量instead of array

2) state machine



s1[0] = -prices[0];

s0[0] = 0;

s2[0] = INT\_MIN;

**for** (**int** i = 1; i < prices.size(); i++) {

s0[i] = max(s0[i - 1], s2[i - 1]);

s1[i] = max(s1[i - 1], s0[i - 1] - prices[i]);

s2[i] = s1[i - 1] + prices[i];

}

**return** max(s0[prices.size() - 1], s2[prices.size() - 1]);

转移方程有一些不同, due to definition. 方法一按definition好理解