August 10, 2018 题目：638,748,313,406,22,822,698,707,282,614,251,477,341

638

1. DFS 本来想用DP做但是要6维。。。

```
class Solution {
    public int shoppingOffers(List<Integer> price, List<List<Integer>> special, List<Integer> needs) {
        return helper(price, special, needs, 0, new HashMap<>());
    }
    public int helper(List<Integer> price, List<List<Integer>> special, List<Integer> needs, int pos, Map<List<Integer>, Integer> map) {
        if (price == null || price.size() == 0 || needs == null || needs.size() == 0) {
            return 0;
        }
        if (map.containsKey(needs)) {
            return map.get(needs);
        }
        int min = 0;
        for (int i = 0; i < needs.size(); i++) {
            min += needs.get(i) * price.get(i);
        }
        for (int i = pos; i < special.size(); i++) {
            List<Integer> tmp = new ArrayList<>(needs);
            List<Integer> s = special.get(i);
            int j = 0;
```

```
            while (j < s.size() - 1) {
                if (s.get(j) > tmp.get(j)) {
                    break;
                }
                tmp.set(j, tmp.get(j) - s.get(j));
                j++;
            }
            if (j == s.size() - 1) {
                min = Math.min(min, s.get(s.size() - 1) + help
er(price, special, tmp, i, map));
            }
        }
        map.put(needs, min);
        return min;
    }
}
```

2. 楼上所说的dp:

```
class Solution {
    int dp[120000];
    int n;
    vector<int> deCode(int stat){
        vector<int> ans;
        for(int i=0; i<n; ++i, stat/=7) ans.push_back(stat%7);
        return ans;
    }
    int enCode(vector<int> needs){
        assert(needs.size()==n);
        int ans = 0;
        for(int i=n-1; i>=0; --i) ans = ans*7 + needs[i];
        return ans;
```

```
    }
    int dfs(int stat, vector<int>& P, vector<vector<int>>& S){
        if(!stat) return 0;
        if(dp[stat] >= 0) return dp[stat];
        vector<int> nd = deCode(stat);
        dp[stat] = 0;
        for(int i=0; i<n; ++i) dp[stat] += nd[i] * P[i];
        for(vector<int> sp: S){
            bool ok = true;
            vector<int> tmp(nd);
            for(int i=0; i<n && ok; ++i){
                tmp[i] -= sp[i];
                if(tmp[i]<0) ok=false;
            }
            if(ok) dp[stat] = min(dp[stat], sp[n] + dfs(enCode
(tmp), P, S));
        }
        return dp[stat];
    }
public:
    int shoppingOffers(vector<int>& price, vector<vector<int>>
& special, vector<int>& needs) {
        n = needs.size();
        memset(dp, -1, sizeof(dp));
        return dfs(enCode(needs), price, special);
    }
};
```

748

1. Easy level:

```
class Solution {
```

```
    typedef pair<int, int> ii;
public:
    string shortestCompletingWord(string L, vector<string>& wo
rds) {
        vector<int> ref(26, 0);
        for(char c: L){
            if(c>='a' && c<='z') ref[int(c-'a')]++;
            if(c>='A' && c<='Z') ref[int(c-'A')]++;
        }
        string ans = "";
        for(string s: words){
            vector<int> tmp(26, 0);
            for(char c: s){
                if(c>='a' && c<='z') tmp[int(c-'a')]++;
                if(c>='A' && c<='Z') tmp[int(c-'A')]++;
            }
            bool ok = true;
            for(int i=0; i<26&&ok; ++i) if(tmp[i] < ref[i]) ok
=false;
            if(ok && (s.size() < ans.size() || ans.empty())) a
ns = s;
        }
        return ans;
    }
};
```

313

1.

```
class Solution {
    public int nthSuperUglyNumber(int n, int[] primes) {
        // 2 7 13 19
```

```
        // 1,2,4,7,8,13,14,16,19,26,28,32
        // 1,2,2*2,7,2*2*2,13,2*7,2*2*2*2,19,2*13,2*2*7,2*2*2*
2*2
        int[] ugly = new int[n];
        int[] indices = new int[primes.length];
        ugly[0] = 1;
        int i = 1;
        while (i < n) {
            ugly[i] = Integer.MAX_VALUE;
            for (int j = 0; j < primes.length; j++) {
                ugly[i] = Math.min(primes[j] * ugly[indices
[j]], ugly[i]);
            }
            for (int j = 0; j < primes.length; j++) {
                if (ugly[i] >= primes[j] * ugly[indices[j]]) i
ndices[j]++;
            }
            i++;
        }
        return ugly[n - 1];
    }
}
```

2. Heap：别想太多就行了，因为 leetcode 给的test case 根本没有到 `1E8`

```
class Solution {
public:
    int nthSuperUglyNumber(int N, vector<int>& P) {
        int n = P.size(), ans = 0;
        priority_queue<int, vector<int>, greater<int>> Q;
        Q.push(1);
        for(; N; --N){
            ans = Q.top();
```

```
            while(!Q.empty() && Q.top()==ans) Q.pop();
            for(int k: P) if(ans < INT_MAX/k) Q.push(ans * k);
        }
        return ans;
    }
};
```

406

1. 就是Brute Force插入，注意插的时候维持 k 的值：

```
class Solution {
public:
    vector<pair<int, int>> reconstructQueue(vector<pair<int, int>>& people) {
        int n = people.size();
        if(n < 2) return people;
        vector<pair<int, int>> ans(n, pair<int, int>(-1, -1));
        sort(people.begin(), people.end());
        for(int cur=people[0].first, k=0, j=0, i=0; i<n; ++i){
            if(people[i].first > cur) {
                k = j = 0;
                cur = people[i].first;
            }
            while(j<n && k<people[i].second) if(ans[j++].second<0) ++k;
            while(j<n && ans[j].first>=0) ++j;
            ans[j] = people[i];
            ++k;
        }
        return ans;
    }
};
```

- [ ] Look at this https://leetcode.com/problems/queue-reconstruction-by-height/discuss/89342/O(nlogn)-Binary-Index-Tree-C++-solution @Zebo L

22

1. DP or recursion

```
class Solution {
public:
    vector<string> generateParenthesis(int n) {
        if(!n) return vector<string>();
        vector<vector<string>> dp(n+1);
        dp[0].push_back("");
        for(int i=1; i<=n; ++i){
            for(int j=0; j<i; ++j){
                for(string s1: dp[j]) for(string s2: dp[i-j-1]) dp[i].push_back("(" + s1 + ")" + s2);
            }
        }
        return dp[n];
    }
};
```

822

1. 小技巧，做一个hash：

```
class Solution {
    const int N = 2018;
public:
    int flipgame(vector<int>& F, vector<int>& B) {
        unordered_set<int> hs;
        int ans = N;
        for(int i=0; i<F.size(); ++i) hs.insert(F[i] * N + B[i]);
        for(int i=0; i<F.size(); ++i) if(F[i] != B[i]){
```

```
            if(!hs.count(F[i]*N + F[i])) ans = min(F[i], ans);
            if(!hs.count(B[i]*N + B[i])) ans = min(B[i], ans);
        }
        return ans<N? ans: 0;
    }
};
```

698

1. Far from optimal:

```
class Solution {
    int N;
    vector<unordered_map<int, bool>> dp;
    bool dfs(int k, int stat, int i, vector<int>&S){
        if(!k) return stat!=N;
        if(i==S.size()) return false;
        if(dp[k].count(stat)) return dp[k][stat];
        if(dfs(k, stat, i+1, S)) return dp[k][stat] = dfs(k, s
tat, i+1, S);
        if(!(stat & S[i])) return dp[k][stat] = dfs(k-1, stat|
S[i], i+1, S);
        return dp[k][stat]=false;
    }
public:
    bool canPartitionKSubsets(vector<int>& nums, int k) {
        int sum = 0, n = nums.size();
        N = (1<<n);
        for(int m: nums) sum += m;
        if(sum % k) return false;
        sum /= k;
        vector<int> states;
        for(int j=1; j<N; ++j) {
```

```
            int tmp = 0;
            for(int i=0; i<n; ++i) if(1&(j>>i)) tmp += nums
[i];
            if(tmp == sum) states.push_back(j);
        }
        dp.resize(k);
        return dfs(k-1, 0, 0, states);
    }
};
```

- [ ] Look at other people's solution @Zebo L https://leetcode.com/problems/partition-to-k-equal-sum-subsets/discuss/155936/c++-dp-with-bitmask.-easy-to-explain-and-impl-in-20mins.-may-not-have-time-for-another-hard-so-sad

707

1. 这种题居然错了两次：

```
class MyLinkedList {
    struct Node{
        int val;
        Node *prev, *next;
        Node(int x=0, Node* p=NULL, Node *n=NULL): val(x), pre
v(p), next(n) {}
    };
    Node *lead, *tail;
    int sz;
public:
    /** Initialize your data structure here. */
    MyLinkedList(): sz(0){
        lead = new Node(0);
        tail = new Node(0);
        lead->next = tail;
        tail->prev = lead;
```

```
    }
    Node *at(int i){
        if(i>=sz) return NULL;
        Node *p=lead->next;
        for(int j=0; j<i; ++j) p=p->next;
        return p;
    }
    int get(int index) {
        Node *p = this->at(index);
        if(p) return p->val;
        return -1;
    }
    void addAtHead(int val) {
        Node *p = new Node(val, lead, lead->next);
        lead->next->prev = p;
        lead->next = p;
        ++sz;
    }
    void addAtTail(int val) {
        Node *p = new Node(val, tail->prev, tail);
        tail->prev->next = p;
        tail->prev = p;
        ++sz;
    }
    void addAtIndex(int index, int val) {
        if(index > sz) return ;
        if(index == sz) this->addAtTail(val);
        else{
            Node *w = this->at(index);
            Node *p = new Node(val, w->prev, w);
```

```
            w->prev->next = p;
            w->prev = p;
            ++sz;
        }
    }
    void deleteAtIndex(int index) {
        if(index < sz){
            Node *p = this->at(index);
            p->prev->next = p->next;
            p->next->prev = p->prev;
            delete p;
            --sz;
        }
    }
};
```

282

- [ ] WTF???? @Zebo L

614

SQL

251

1. Easy 才对：

```
class Vector2D {
    queue<int> Q;
public:
    Vector2D(vector<vector<int>>& vec2d) { for(auto vec: vec2
d) for(auto k: vec) Q.push(k); }
    int next() {
        int ans = Q.front();
        Q.pop();
```

```
        return ans;
    }
    bool hasNext() { return !Q.empty(); }
};
```

477

1. 超时了。。

```
class Solution {
    public int totalHammingDistance(int[] nums) {
        // 4 14 => 0100^1110 =>1010 1010 & 1001 = 1000 => 1000
& 0111 = 0
        int res = 0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                res += hamming(nums[i], nums[j]);
            }
        }
        return res;
    }
    public int hamming(int x, int y) {
        int t = x ^ y; // keep different positions
        int res = 0;
        while (t != 0) {
            t = t & (t - 1);
            res++;
        }
        return res;
    }
}
```

2. improve以后

```
class Solution {
```

```
    public int totalHammingDistance(int[] nums) {
        int res = 0;
        for (int i = 0; i < 32; i++) {
            int bit = 0;
            for (int n : nums) {
                bit += (n >> i) & 1;
            }
            res += bit * (nums.length - bit);
        }
        return res;
    }
}
```

3. 同上：

```
class Solution {
public:
    int totalHammingDistance(vector<int>& nums) {
        vector<vector<int>> cnt(32, vector<int>(2, 0));
        for(int k: nums) for(int j=0; j<32; ++j) ++cnt[j][1&(k
>>j)];
        int ans = 0;
        for(auto vec: cnt) ans += vec[0] * vec[1];
        return ans;
    }
};
```

341

1. Should be easy:

```
class NestedIterator {
    queue<int> Q;
public:
    NestedIterator(vector<NestedInteger> &nl) {
```

```
        stack<NestedInteger> S;
        for(int i=nl.size()-1; i>=0; --i) S.push(nl[i]);
        while(!S.empty()){
            auto r = S.top();
            S.pop();
            while(!r.isInteger() && !r.getList().empty()){
                auto vec = r.getList();
                for(int i=vec.size()-1; i; --i) S.push(vec
[i]);
                            r = vec[0];``````````

           f
        }
            if(r.isInteger()) Q.push(r.getInteger();)
        }
    }
    int next() {
        int ans = Q.front();
        Q.pop();
        return ans;
    }
    bool hasNext() { return !Q.empty(); }
  };`
```