https://leetcode.com/discuss/interview-question/344650/Amazon-Online-Assessment-Questions
https://aonecode.com/amazon-online-assessment-questions
https://rooooobin.github.io/2020/06/03/Amazon-OA-Questions/

https://medium.com/@scarletinked/are-you-the-leader-were-looking-for-interviewing-at-amazon-8301d787815d

目录

Maximum Units

https://leetcode.com/discuss/interview-question/793606/
O(NlogN) runtime where N = number of boxes or unitSize

```
static long getMaxUnit(int num, List<Integer> boxes, int unitSize, List<Integer>
unitsPerBox, long truckSize) {
   PriorityQueue<int[]> maxHeap = new PriorityQueue<>((b1, b2) -> b2[1] - b1[1]);
   for (int i = 0; i < boxes.size(); i++) {
      maxHeap.add(new int[] { boxes.get(i), unitsPerBox.get(i)});
   }
   long count = 0;
   while (truckSize > 0 && !maxHeap.isEmpty()) {
      int[] curr = maxHeap.poll();
      long boxesTruckCanHold = Math.min(curr[0], truckSize);
      count = count + (boxesTruckCanHold * curr[1]);
      truckSize = truckSize - boxesTruckCanHold;
   }
   return count;
}
```

Subtree with Maximum Average

https://leetcode.com/discuss/interview-question/349617
Java bottom-up naive recursion solution with complexity O(N).

```
public class Node {
   public int val;
   public List<Node> children;
   public Node() {}
   public Node(int _val) { val = _val; }
   public Node(int _val,List<Node> _children) {
      val = _val;
      children = _children;
   }
};
double max = 0;
Node res = null;
public int[] computeAvg(Node root){
    if(root == null) return new int[]{0, 0};
    if(root.children == null) return new int[]{root.val, 1};
    int val = root.val, count = 1;
    for(Node child: root.children){
        int[] arr = computeAvg(child);
        val += arr[0]; count += arr[1];
    }
    if(count > 1 && (res == null || val / (0.0 + count) > max)){
        res = root;
        max = val / (0.0 + count);
    }
    return new int[]{val, count};
}
public Node subtreeWithMaximumAverage(Node root){
    if(root == null) return res;
    computeAvg(root);
    return res;
}
```

Disk Space Analysis

https://leetcode.com/discuss/interview-question/808348/
Time Complexity : O(n) & space complexity : O(segmentLength) as we will be remove if its is greater than segment Length

```
int MaxInMinimal(int numComputer, List<Integer> hardDiskSpace, int segmentLength) {
    int ans = Integer.MIN_VALUE;;
    Deque<Integer> q = new ArrayDeque<>();

    for (int i=0; i<hardDiskSpace.size(); i++) {
        int current = hardDiskSpace.get(i);
        while (!q.isEmpty() && current<hardDiskSpace.get(q.peekLast())) q.pollLast();

        if (!q.isEmpty() && q.peekFirst() <= (i - segmentLength))  q.removeFirst();
        q.addLast(i);
        if (i >= segmentLength-1) ans = Math.max(ans, hardDiskSpace.get(q.peekFirst()));
    }
    return ans;
}
```

Nearest City

https://leetcode.com/discuss/interview-question/808374/
time: O(numOfCities * numOfQueries) space: O(numOfCities)

```
private String[] solve(String[] cities, int[] xs, int[] ys, String[] queries) {
    String[] res = new String[queries.length];
    Map<Integer, TreeMap<Integer, String>> xMap = new HashMap<>();
    Map<Integer, TreeMap<Integer, String>> yMap = new HashMap<>();
    Map<String, int[]> nodeMap = new HashMap<>();
    for(int i=0;i<cities.length;i++) {
        nodeMap.put(cities[i], new int[] {xs[i], ys[i]});
        xMap.putIfAbsent(xs[i], new TreeMap<>());
        yMap.putIfAbsent(ys[i], new TreeMap<>());
        xMap.get(xs[i]).put(ys[i], cities[i]);
        yMap.get(ys[i]).put(xs[i], cities[i]);
    }
    for(int i=0;i<queries.length;i++) {
        int[] node = nodeMap.get(queries[i]);
        TreeMap<Integer, String> ym = xMap.getOrDefault(node[0], new TreeMap<>());
        TreeMap<Integer, String> xm = yMap.getOrDefault(node[1], new TreeMap<>());
        Integer yl = ym.lowerKey(node[1]), yh = ym.higherKey(node[1]);
        Integer xl = xm.lowerKey(node[0]), xh = xm.higherKey(node[0]);
        int dist = Integer.MAX_VALUE;
        if(yl != null && Math.abs(yl - node[1]) <= dist) {
            dist = Math.abs(yl - node[1]);
            res[i] = res[i] == null ? ym.get(yl) : res[i].compareTo(ym.get(yl)) > 0 ?
ym.get(yl) : res[i];
        }
        if(yh != null && Math.abs(yh - node[1]) <= dist) {
            dist = Math.abs(yh - node[1]);
            res[i] = res[i] == null ? ym.get(yh) : res[i].compareTo(ym.get(yh)) > 0 ?
ym.get(yh) : res[i];
        }
        if(xl != null && Math.abs(xl - node[0]) <= dist) {
            dist = Math.abs(xl - node[0]);
            res[i] = res[i] == null ? xm.get(xl) : res[i].compareTo(xm.get(xl)) > 0 ?
xm.get(xl) : res[i];
        }
        if(xh != null && Math.abs(xh - node[1]) <= dist) {
            dist = Math.abs(xh - node[1]);
            res[i] = res[i] == null ? xm.get(xh) : res[i].compareTo(xm.get(xh)) > 0 ?
xm.get(xh) : res[i];
        }
        if(res[i] == null)
            res[i] = "None";
    }
    return res;
}
```

Fetch Items To Display

第二道 fetch display items 用 pq 会超时 一种方法可以用 treemap 注意 lc 讨论区的几个答案有错的也有会超时的 另外 map 的 key 是 string value， value 是 pairInt ， pairint 通过 first second 两个属性访问值
https://leetcode.com/discuss/interview-question/823159/amazon-oa-aug-2020-fetch-items-to-display
Time - O(nlog(n)), Space - O(n), n = number of items.

```
    public List fetchItemsToDisplay(int numOfItems, HashMap<String, int[]> items, int
sortParameter, int sortOrder, int itemsPerPage, int pageNumber) {
        PriorityQueue<DisplayItems> pq = new PriorityQueue<>();
        if (sortOrder == 1)
            pq = new PriorityQueue<>(Collections.reverseOrder());

        for (Map.Entry<String, int[]> map : items.entrySet()) {
            if(sortParameter == 0) pq.add(new DisplayItems(-1, map.getKey()));
            else pq.add(new DisplayItems(map.getValue()[sortParameter - 1],
map.getKey()));
        }

        List<String> result = new ArrayList<>();
        while (!pq.isEmpty()) {
            result.add(pq.peek().itemName);
            pq.poll();
        }

        int startIndex = pageNumber * itemsPerPage;
        int endIndex = (startIndex + itemsPerPage) > result.size() ? result.size() :
startIndex + itemsPerPage;

        return result.subList(startIndex, endIndex);
    }

public class DisplayItems implements Comparable<DisplayItems> {
    private String itemName;
    private Integer value;

    public DisplayItems(Integer value, String itemName) {
        this.itemName = itemName;
        this.value = value;
    }

    public String getItemName() {
        return itemName;
    }

    public Integer getValue() {
        return value;
    }

    @Override
    public int compareTo(DisplayItems o) {
        if(this.value == -1) return this.getItemName().compareTo(o.itemName);
        return this.getValue().compareTo(o.value);
    }
}
```

Count Teams

https://aonecode.com/amazon-online-assessment-create-teams

Time - O(m*m), Space - O(m*m), m = count.

```
    public int countTeams(int num, int[] skills, int minAssociates, int minLevel, int
maxLevel) {
        int count = 0;
        for(int i = 0; i < num; i++){
            if(skills[i] >= minLevel && skills[i] <= maxLevel) count++;
        }
        int res = 0;
        for(int i = minAssociates; i <= count; i++){
            res += comb(count, i);
        }
        return res;
    }

    Map<String,Integer> map= new HashMap<String, Integer>();
    private int comb(int m,int n){
        String key= m+","+n;
        if(n==0)
            return 1;
        if (n==1)
            return m;
        if(n>m/2)
            return comb(m,m-n);
        if(n>1){
            if(!map.containsKey(key))
                map.put(key, comb(m-1,n-1)+comb(m-1,n));
            return map.get(key);
        }
        return 0;
    }
```

Critical Routers

https://leetcode.com/discuss/interview-question/436073/

```
    private static List<Integer> getCriticalNodes(int[][] links, int numLinks, int
numRouters) {
        time = 0;
        Map<Integer, Set<Integer>> map = new HashMap<>();
        for(int i=0;i<numRouters;i++) {
            map.put(i, new HashSet<>());
        }
        for(int[] link : links) {
            map.get(link[0]).add(link[1]);
            map.get(link[1]).add(link[0]);
        }
        Set<Integer> set = new HashSet<>();
        int[] low = new int[numRouters];
        int[] ids = new int[numRouters];
        int parent[] = new int[numRouters];
        Arrays.fill(ids, -1);
        Arrays.fill(parent, -1);
        for(int i=0;i<numRouters;i++) {
            if(ids[i] == -1)
                dfs(map, low, ids, parent, i, set);
        }
        return new ArrayList<>(set);
    }

    private static void dfs(Map<Integer, Set<Integer>> map, int[] low, int[] ids, int[]
parent, int cur, Set<Integer> res) {
        int children = 0;
        ids[cur] = low[cur]= ++time;
        for(int nei : map.get(cur)) {
            if(ids[nei] == -1) {
                children++;
                parent[nei] = cur;
                dfs(map, low, ids, parent,nei, res);
                low[cur] = Math.min(low[cur], low[nei]);
                if((parent[cur] == -1 && children > 1) || (parent[cur] != -1 && low[nei] >=
ids[cur]))
                    res.add(cur);
            }
            else if(nei != parent[cur])
                low[cur] = Math.min(low[cur], ids[nei]);
        }
    }
```

Product Suggestions

https://leetcode.com/problems/search-suggestions-system/

```
    public List<List<String>> suggestedProducts(String[] products, String searchWord) {
        List<List<String>> res = new ArrayList<>();
        TreeMap<String, Integer> map = new TreeMap<>();
        Arrays.sort(products);
        List<String> productsList = Arrays.asList(products);

        for (int i = 0; i < products.length; i++) {
            map.put(products[i], i);
        }

        String key = "";
        for (char c : searchWord.toCharArray()) {
            key += c;
            String ceiling = map.ceilingKey(key);
            String floor = map.floorKey(key + "~");
            if (ceiling == null || floor == null) break;
            res.add(productsList.subList(map.get(ceiling), Math.min(map.get(ceiling) +
3, map.get(floor) + 1)));
        }

        while (res.size() < searchWord.length()) res.add(new ArrayList<>());
        return res;
    }
```

Copy List with Random Pointer

https://leetcode.com/problems/copy-list-with-random-pointer/

```
    public Node copyRandomList(Node head) {
        Node cur = head;
        while(cur != null){
            Node copy = new Node(cur.val);
            copy.next = cur.next;
            cur.next = copy;
            cur = copy.next;
        }

        cur = head;
        while(cur != null){
            if(cur.random != null) cur.next.random = cur.random.next;
            cur = cur.next.next;
        }

        Node result = new Node(0), copy = result;
        cur = head;
        while(cur != null){
            Node tnext = cur.next.next;
            copy.next = cur.next;
            copy = copy.next;
            cur.next = tnext;
            cur = cur.next;

        }

        return result.next;
    }
```

Merge Two Sorted Lists

https://leetcode.com/problems/merge-two-sorted-lists/

```
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        if(l1 == null) return l2;
        if(l2 == null) return l1;
        ListNode result = new ListNode(0), cur = result;
        result.next = l1;
        while(l1!=null && l2!=null){
            if(l1.val < l2.val){
                l1 = l1.next;
            }
            else{
                ListNode tmp = cur.next;
                cur.next = l2;
                ListNode other = l2.next;
                l2.next = tmp;
                l2 = other;
            }
            cur = cur.next;
        }
        if(l1 == null) cur.next = l2;
        return result.next;
    }
```

Subtree of Another Tree

https://leetcode.com/problems/subtree-of-another-tree/

```
   public boolean isSubtree(TreeNode s, TreeNode t) {
      if(s == null) return false;
      if(isSame(s, t)) return true;
      return isSubtree(s.left, t) || isSubtree(s.right, t);
   }

   private boolean isSame(TreeNode s, TreeNode t){
      if(s == null && t == null) return true;
      if(s == null || t == null) return false;
      if(s.val != t.val) return false;
      return isSame(s.left, t.left) && isSame(s.right, t.right);
   }
```

Search a 2D Matrix II

https://leetcode.com/problems/search-a-2d-matrix-ii/

```
public boolean searchMatrix(int[][] matrix, int target) {
      if(matrix == null || matrix.length <= 0 || matrix[0].length <= 0)  return false;
      for(int i = 0, j = matrix[0].length-1; i < matrix.length && j >= 0;){
         if(matrix[i][j] == target)  return true;
         else if(matrix[i][j] > target) j--;
         else if(matrix[i][j] < target) i++;
      }
      return false;
   }
```

Critical Connections

https://leetcode.com/discuss/interview-question/372581

```
class Solution{
   List<PairInt> list;
```

```
    Map<Integer, Boolean> visited;
    List<PairInt> criticalConnections(int numOfServers, int numOfConnections,
List<PairInt> connections)
    {
        Map<Integer, HashSet<Integer>> adj = new HashMap<>();
        for(PairInt connection : connections){
            int u = connection.first;
            int v = connection.second;
            if(adj.get(u) == null){
                adj.put(u,new HashSet<Integer>());
            }
            adj.get(u).add(v);
            if(adj.get(v) == null){
                adj.put(v,new HashSet<Integer>());
            }
            adj.get(v).add(u);
        }

        list = new ArrayList<>();
        for(int i =0;i<numOfConnections;i++){
            visited = new HashMap<>();
            PairInt p = connections.get(i);
            int x = p.first;
            int y = p.second;
            adj.get(x).remove(y);
            adj.get(y).remove(x);
            DFS(adj,1);
            if(visited.size()!=numOfServers){
                    if(p.first > p.second)
                        list.add(new PairInt(p.second,p.first));
                    else
                        list.add(p);
            }
            adj.get(x).add(y);
            adj.get(y).add(x);
        }
        return list;
    }

    public void DFS(Map<Integer, HashSet<Integer>> adj, int u){
        visited.put(u, true);
        if(adj.get(u).size()!=0){
            for(int v : adj.get(u)){
                if(visited.getOrDefault(v, false)== false){
                    DFS(adj,v);
                }
            }
        }
    }
}
```

Favorite Genres

https://leetcode.com/discuss/interview-question/373006

```
class Solution {
    Map<String, List<String>> favoriteGenre(Map<String, List<String>> userSongs,
Map<String, List<String>> songGenres) {
        Map<String, String> songToGenre = new HashMap<>();
        songGenres.forEach((genre, songs) -> songs.forEach(song -> songToGenre.put(song,
genre)));
        Map<String, List<String>> favoriteGenre = new HashMap<>();
        userSongs.forEach((user, songs) -> favoriteGenre.put(user,
calculateFavoriteGenre(songs, songToGenre)));
        return favoriteGenre;
    }

    private List<String> calculateFavoriteGenre(List<String> songs, Map<String, String>
songToGenre) {
        Map<String, Integer> genreFrequency = new HashMap<>();
        List<String> favGenre = new ArrayList<>();
        int maxFrequency = 0;
        for (String song : songs) {
            String genre = songToGenre.get(song);
            Integer frequency = genreFrequency.merge(genre, 1, (ov, nv) -> ov + 1);
            maxFrequency = Math.max(frequency, maxFrequency);
        }
        for (Map.Entry<String, Integer> entry : genreFrequency.entrySet()) {
            if (entry.getValue() == maxFrequency) {
                favGenre.add(entry.getKey());
            }
        }
        return favGenre;
    }
}
```

Two Sum - Unique Pairs

https://leetcode.com/discuss/interview-question/372434

```
    public static int uniquePairs(int[] nums, int target){
        Set<Integer> set = new HashSet<Integer>();
        Set<Integer> seen = new HashSet<Integer>();
        int count = 0;
        for(int num : nums){
            if(set.contains(target-num) && !seen.contains(num)){
                count++;
                seen.add(target-num);
                seen.add(num);
            }
            else if(!set.contains(num)){
                set.add(num);
            }

        }

        return count;
    }
```

Spiral Matrix II

https://leetcode.com/problems/spiral-matrix-ii/

```
class Solution {
    public int[][] generateMatrix(int n) {
        int[][] res = new int[n][n];
        int[][] dirt = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
        int count = 1, row = 0, col = 0, d = 0;
        while(count <= n*n){
            res[row][col] = count++;
            int r = Math.floorMod(row + dirt[d][0], n);
            int c = Math.floorMod(col + dirt[d][1], n);

            if(res[r][c] != 0) d = (d+1) % 4;
            row += dirt[d][0];
            col += dirt[d][1];
        }
        return res;
    }
}
```

Count LRU Cache Misses

https://aonecode.com/amazon-online-assessment-lru

```
public static int lruCacheMisses(int num, List<Integer>pages, int maxCacheSize)  {
    LRUcache MemoA = new LRUcache(maxCacheSize);

    for(int i = 0; i < num; i++) MemoA.set(pages.get(i), i);
    return MemoA.missNum;
}

class LRUcache {
    HashMap<Integer, Integer> map;
    ArrayList<Integer> list;
    int capacity;
    int missNum;

    public LRUcache(int capacity_val)
    {
        capacity = capacity_val;
        map = new HashMap<Integer,Integer>(capacity);
        list = new ArrayList<Integer>(capacity);
        missNum = 0;
    }

    public int set(int key, int i2) {
        if(map.size() < capacity || map.containsKey(key)){
            if(map.containsKey(key)){
                map.put(key, i2);
                list.remove(new Integer(key));
                list.add(key);
            }
            else{
                map.put(key, i2);
                list.add(key);
                missNum++;
            }
        }
        else{
            int lastkey = list.get(0);
            list.remove(0);
            map.remove(lastkey);

            list.add(key);
            map.put(key, i2);
            missNum++;
        }
        return key;
    }

    public int get(int key) {
        if(map.containsKey(key)) {
            list.remove(new Integer(key));
            list.add(key);
            return map.get(key);
        }
        else return -1;
    }
}
```

Turnstile

https://leetcode.com/discuss/interview-question/699973/Goldman-Sachs-or-OA-or-Turnstile

```
class Solution {
    public int[] getTimes(int numCustomers, int[] arrTime, int[] direction) {
        PriorityQueue<Integer> en = new PriorityQueue<Integer>((a,
b)->arrTime[a]-arrTime[b]);
        PriorityQueue<Integer> ex = new PriorityQueue<Integer>((a,
b)->arrTime[a]-arrTime[b]);
        int[] res = new int[numCustomers];

        for(int i = 0; i < direction.length; i++){
            if(direction[i] == 1) ex.add(i);
            else en.add(i);
        }

        int curt = 0, lastd = -1;
        while(!ex.isEmpty() || !en.isEmpty()){
            if(!ex.isEmpty() && arrTime[ex.peek()] <= curt && ( lastd == 1 || lastd == -1
|| en.isEmpty() || arrTime[en.peek()] > curt && lastd == 0)) {
                res[ex.peek()] = curt;
                lastd = 1;
                ex.poll();
            }
            else if(!en.isEmpty() && arrTime[en.peek()] <= curt)
            {
                res[en.peek()] = curt;
                lastd = 0;
                en.poll();
            }
            else lastd = -1;

            curt++;
        }
        return res;
    }
}
```

Amazon Debt Records

https://aonecode.com/amazon-online-assessment-amazon-debt-records

```
class Solution {
    List<String> minimumDebtMembers(List<debtRecord> records){
        HashMap<String, Integer> debt = new HashMap<String, Integer>();
        for(debtRecord record: records){
            debt.put(record.borrower, debt.getOrDefault(record.borrower, 0) - 
record.amount);
            debt.put(record.lender, debt.getOrDefault(record.lender, 0) + 
record.amount);
        }

        int min = Collections.min(debt.values());
        List<String> res = new ArrayList<>();
        for (Map.Entry<String, Integer> entry : debt.entrySet()) {
            if (min == entry.getValue()) {
                res.add(entry.getKey());
            }
        }
        Collections.sort(res);
        return res;
    }
}
```

Baseball Scorekeeping

https://aonecode.com/amazon-online-assessment-baseball-scorekeeping

```
class Solution {
    int baseballScorekeeping(String[] blocks){
        int res = 0;
        Stack<Integer> b = new Stack<Integer>();

        for(String block: blocks){
            if(block == "X" && !b.isEmpty()){
                int cur = b.peek()*2;
                res += cur;
                b.push(cur);
            }
            else if(block == "+" && !b.isEmpty()){
                int temp = b.pop();
                int cur = temp;
                if(!b.isEmpty()) cur += temp;
                b.push(temp);
                res += cur;
                b.push(cur);
            }
            else if(block == "Z" && !b.isEmpty()){
                int cur = b.pop();
                res -= cur;
            }
            else{
                int cur = Integer.valueOf(block);
                res += cur;
                b.push(cur);
            }
        }
        return res;
    }
}
```

Find The Highest Profit

https://leetcode.com/discuss/interview-question/823177/amazon-oa-2020-find-the-highest-profit

```
class Solution {
    int highestProfit(int numSuppliers, int[] inventory, int order){
        Map<Integer, Integer> p = new HashMap<>();
        for(int price: inventory) p.put(price, p.getOrDefault(price, 0)+1);
        int currMax = Collections.max(p.keySet());

        int res = 0;
        while(order > 0){
            int maxi = Math.min(order, p.get(currMax));
            res += currMax*maxi;
            order -= maxi;
            p.put(currMax, p.get(currMax) - maxi);
            p.put(currMax-1, p.getOrDefault(currMax-1, 0) + maxi);
            if(p.get(currMax) == 0){
                p.remove(currMax);
                currMax -= 1;
            }
        }
        return res;
    }
}
```

Squared Shortest Distance

https://leetcode.com/discuss/interview-question/821708/amazon-oa-shortest-mean-squared-distance-between-robots-help

```
public static int shortestDistace(int n, int[] xPos, int[] yPos) {
        int minDistance = Integer.MAX_VALUE;
        for(int i = 0; i < n - 1; i++) {
            for(int j = i + 1; j < n; j++) {
                int distance = ((xPos[j] - xPos[i]) * (xPos[j] - xPos[i])) + ((yPos[j] -
yPos[i]) * (yPos[j] - yPos[i]));
                if(distance > 0) { // Not considering robos at same point
                    minDistance = Math.min(minDistance, distance);
                }
            }
        }
        return minDistance;
    }
```

```
class Solution{
    long ans;

    public long closestPair(int numRobots, int[] positionX, int[] positionY){
        if(numRobots < 2)  return 0;
        int N = 0;

        ans = Long.MAX_VALUE;
        HashSet<String> myset = new HashSet<>();
        for(int i = 0; i < numRobots; i++){
            String thep = Integer.toString(positionX[i])+"
"+Integer.toString(positionY[i]);
            if(myset.contains(thep))  continue;
            positionX[N]=positionX[i];
            positionY[N]=positionY[i];
            N++;
            myset.add(thep);
        }
        if(N<2)  return 0;

        int e;
        for(e = 29;e >= 0;e--)
            if(!search(positionX, positionY, N, e, false)) break;

        search(positionX, positionY, N, e+1, true);
        if(e+2<30)  search(positionX, positionY, N, e+2, true);
        return ans;
    }

    boolean search(int[] positionX, int[] positionY, int n, int e,boolean checkpairs){
        HashMap<Long, ArrayList<Integer>> S = new HashMap<>();

        boolean found=false;
        for(int i = 0; i < n; i++) {
            long hx = positionX[i] >> e;
            long hy = positionY[i] >> e;

            for (long gx = Math.max(0L, hx - 1L); gx <= hx + 1L; gx++){
                for (long gy = Math.max(0L, hy - 1L); gy <= hy + 1L; gy++){
                    long val = (gx << 30) + gy;
                    if (S.containsKey(val)) {
                        found = true;
                        if (!checkpairs) return true;

                        for (int j : S.get(val)) {
                            int dx = positionX[i] - positionX[j];
                            int dy = positionY[i] - positionY[j];
                            long square_dist = (long) dx * dx + (long) dy * dy;
                            ans = Math.min(ans, square_dist);
                        }
                    }
                }
            }
            long key = (hx<<30)+hy;
            ArrayList<Integer> ka = S.getOrDefault(key, new ArrayList<>());
            ka.add(i);
            S.put(key, ka);
        }
        if(checkpairs)  assert(found);//safe check
```

```
      return found;
   }
};
```

Split String Into Unique Primes

https://leetcode.com/discuss/interview-question/833164/

```
   private static int countPrimeStrings(int n) {
      int mod = (int)1e9 + 7;
      boolean[] arr = new boolean[(int)1e6 + 1];
      Arrays.fill(arr, true);
      for(int i = 2; i*i <= (int)1e6; i++) {
         if(arr[i]) {
            for(int j = i; j*i <= (int)1e6; j++) {
               arr[i*j] = false;
            }
         }
      }
      arr[1] = false;
      arr[0] = false;
      String s = String.valueOf(n);
      int[] dp = new int[s.length() + 1];
      dp[0] = 1;
      for(int i = 1;i <= s.length(); i++) {
         for(int j = Math.max(0, i-6); j < i; j++) {
            if(arr[Integer.parseInt(s.substring(j, i))]) {
               dp[i] = (dp[i] + dp[j]) % mod;
            }
         }
      }
      return dp[s.length()];
   }
```

Disk Space Analysis

https://leetcode.com/discuss/interview-question/808348/amazon-oa-2020-disk-space-analysis

```
class Solution{
    int diskSpaceAnalysis(int computers, List<Integer> hardDisks, int length) {
        int ans = -1;
        Deque<Integer> q = new ArrayDeque<>();

        for (int right = 0; right < hardDisks.size(); right++) {
            int current = hardDisks.get(right);
            while (q.size()>0 && current<q.getLast()) {
                q.removeLast();
            }
            q.addLast(current);

            if (right >= length-1) {
                ans = Math.max(ans, q.getFirst());
            }

            if (q.size() >= length) {
                q.removeFirst();
            }
        }

        return ans;
    }
};
```

Secret Fruit List

https://leetcode.com/discuss/interview-question/762546/amazon-oa-2020-amazon-fresh-promotion

```
public class FindFruitCombs {
    public static int winPrize(String[][] codeList, String[] shoppingCart) {
        if(codeList == null || codeList.length == 0) return 1;
        if(shoppingCart == null || shoppingCart.length == 0) return 0;
        int i = 0, j = 0;
        while (i < codeList.length && j + codeList[i].length <= shoppingCart.length) {
            boolean match = true;
            for (int k = 0; k < codeList[i].length; k++) {
                if (!codeList[i][k].equals("anything")
&& !shoppingCart[j+k].equals(codeList[i][k])) {
                    match = false;
                    break;
                }
            }
            if (match) {
                j += codeList[i].length;
                i++;
            } else {
                j++;
            }
        }
        return (i == codeList.length) ? 1 : 0;
    }
}
```

Find Related Products

https://aonecode.com/amazon-online-assessment-find-related-books

```
class Solution {
    int DFSUtil(String v, Map<String, Boolean> visited, Map<String, Set<String>> map,
List<String> comp) {
        visited.put(v, true);
        comp.add(v);
        for (String x : map.get(v)) {
            if (visited.get(x) != true) DFSUtil(x, visited, map, comp);
        }
        return comp.size();
    }

    public List<String> largestItemAssociation(List<PairString> items){
        Map<String, Set<String>> map = new HashMap<>();
        Map<String, Boolean> visited = new HashMap<>();
        for(PairString item: items){
            if(!map.containsKey(item.first)) map.put(item.first, new HashSet<>());
            if(!map.containsKey(item.second)) map.put(item.second, new HashSet<>());
            if(!visited.containsKey(item.first)) visited.put(item.first, false);
            if(!visited.containsKey(item.second)) visited.put(item.second, false);
            map.get(item.first).add(item.second);
            map.get(item.second).add(item.first);
        }

        List<String> res = new ArrayList<String>();
        int max = 0;

        for (String v: visited.keySet()) {
            if (!visited.get(v)) {
                List<String> cur = new ArrayList<String>();
                int size = DFSUtil(v, visited, map, cur);
                if(size > max){
                    max = size;
                    res = cur;
                }
            }
        }
        return res;
    }
}
```

Count Cluster

https://roooooobin.github.io/2020/06/09/Number-of-Islands-Solution/

```
public static void dfs(char[][] grid, int i, int j){
    if(i < 0 || i >= grid.length || j < 0 || j >= grid[i].length || grid[i][j] == '0')
return;
    grid[i][j] = '0';
    dfs(grid, i+1, j);
    dfs(grid, i-1, j);
    dfs(grid, i, j+1);
    dfs(grid, i, j-1);
}
```

```
public static int numIslands(char[][] grid){
    int cnt = 0;
    for(int i=0; i<grid.length; ++i){
        for(int j=0; j<grid[i].length; ++j){
            if(grid[i][j] == '1'){
                dfs(grid, i, j);
                cnt++;
            }
        }
    }
    return cnt;
}
```

Minimum Difficulty of a Job Schedule

https://leetcode.com/problems/minimum-difficulty-of-a-job-schedule/

```
class Solution {
    public int minDifficulty(int[] A, int D) {
        int n = A.length, maxd;
        if(n < D) return -1;
        int[] dp = new int[n + 1];

        for(int i = n - 1; i >= 0; i--) dp[i] = Math.max(dp[i+1], A[i]);
        for(int d = 2; d <= D; d++){
            for(int i = 0; i <= n - d; i++){
                maxd = 0;
                dp[i] = Integer.MAX_VALUE;
                for(int j = i; j <= n - d; j++){
                    maxd = Math.max(maxd, A[j]);
                    dp[i] = Math.min(dp[i], maxd + dp[j + 1]);
                }
            }
        }
        return dp[0];
    }
}
```

Break a Palindrome

https://leetcode.com/problems/break-a-palindrome/

int str1.compareTo(String str2) return + when str1 > str2

```
class Solution {
    public String breakPalindrome(String palindrome) {
        char[] s = palindrome.toCharArray();
        int n = s.length;

        for(int i = 0; i < n/2; i++){
            if(s[i] != 'a') {
                s[i] = 'a';
                return String.valueOf(s);
            }
        }
        s[n-1] = 'b';
        return n<2 ? "" : String.valueOf(s);
    }
}
```

Max Of Min Altitudes

https://leetcode.com/discuss/interview-question/383669/

```
// DP (One Row or Column)
  // Time: O(rc) Space: O(r or c)
  // DP (One Row or Column)
  private static int maxScore1D(int[][] grid) {
    int r = grid.length, c = grid[0].length;
    int[] dp = new int[c];

    dp[0] = Integer.MAX_VALUE; // first entry is not considered
    for (int j = 1; j < c; ++j) dp[j] = Math.min(dp[j - 1], grid[0][j]);

    for (int i = 1; i < r; ++i) {
      dp[0] = Math.min(dp[0], grid[i][0]);
      for (int j = 1; j < c; ++j) {
```

```
      if (i == r - 1 && j == c - 1) {
        dp[j] = Math.max(dp[j - 1], dp[j]); // last entry is not considered
      } else {
        int score1 = Math.min(dp[j - 1], grid[i][j]); // left  dp[i][j-1]
        int score2 = Math.min(dp[j], grid[i][j]);    // up    dp[i-1][j]
        dp[j] = Math.max(score1, score2);
      }
    }
  }
  return dp[c - 1];
}
```

Distinct Product IDs after removing k

https://leetcode.com/problems/least-number-of-unique-integers-after-k-removals/

```
class Solution {
    public int findLeastNumOfUniqueInts(int[] arr, int k) {
        Map<Integer, Integer> count = new HashMap<>();
        for(int a: arr) count.put(a, count.getOrDefault(a, 0) + 1);

        int remain = count.size(), occur = 1;
        int[] occurCount = new int[arr.length + 1];
        for(int v: count.values()) occurCount[v]++;
        while(k > 0){
            if(k - occur*occurCount[occur] >= 0){
                k -= occur*occurCount[occur];
                remain -= occurCount[occur++];
            }
            else return remain - k / occur;
        }
        return remain;
    }
}
```

Most frequent word

https://leetcode.com/problems/most-common-word/

```
class Solution {
    public String mostCommonWord(String paragraph, String[] banned) {
        Set<String> ban = new HashSet<>(Arrays.asList(banned));
        Map<String, Integer> count = new HashMap<>();
        String[] word = paragraph.replaceAll("\\W+", " ").toLowerCase().split("\\s+");
        for(String w: word) if(!ban.contains(w)) count.put(w, count.getOrDefault(w, 0)
+ 1);
        return Collections.max(count.entrySet(),
Map.Entry.comparingByValue()).getKey();
    }
}
```

Amazon 原则

Work simulation(原则有先后顺序)
目前两大做题中最重要原则：
1.requirement 排在第一，deadline 第二。
2.有 manager 出现的选项无脑选 manager，manager 就是一个组的地头蛇。

Amazon9 条主要原则
原则 1：客户是上帝，requirement 优先，任何影响上帝的事情都不能干，
　　如某个 requirement 影响了上帝的体验，
　　你就是死键盘上也不能砍了，宁愿 miss deadline
原则 2：为长远考虑，即客户几年之后可能会出现的需求也要考虑到，
　　不会为了交付短期的 deadline，
　　而牺牲长期的价值。（比如 global api 和 local api）
原则 3：最高标准，“最高”对应上面的“长远”。
原则 4：一般情况，能请示 manager 就请示 manager，manager 一般不会出错
原则 5：速度很重要，决策和行动都可以改变，因此不需要进行过于广泛的推敲
　　，但提倡在深思熟虑下进行冒险。
原则 6：不需要一定要坚持“非我发明”，需求帮助也是可以的，四处寻找创意

，并且接受长期被误导的可能
原则 7：敢于承担责任，任劳任怨，比如领导说谁会 java，你会你就跳出来说我会
原则 8：对问题刨根问底，探究细节
原则 9：服从大局（团队比个人重要）

客户的痴迷
领导者从客户开始，然后倒退。他们努力工作以赢得并保持客户的信任。尽管领导者关注竞争者，但他们仍然痴迷于客户。

所有权
负责人是所有者。他们长期考虑，不为短期结果牺牲长期价值。他们不仅代表自己的团队，还代表整个公司行事。他们从不说“那不是我的工作”。

发明和简化
领导者期望并要求其团队进行创新和发明，并始终寻求简化的方法。他们具有外部意识，可以从任何地方寻找新的想法，并且不受“此处未发明”的限制。在我们做新事物时，我们接受我们可能会长期误解。

是正确的，很多
领导者是正确的。他们有很强的判断力和良好的直觉。他们寻求不同的观点，并努力证明自己的信念。

学会学习和保持好奇心
领导者永远都不会学习，而总是寻求自我完善。他们对新的可能性感到好奇，并采取行动探索它们。

雇用和培养最佳
领导者每次聘用和晋升时，绩效标准都会提高。他们认识到卓越的人才，并乐意将他们转移到整个组织中。领导者要培养领导者，并认真对待自己的教练角色。我们代表员工开展工作，发明诸如“职业选择”之类的发展机制。

坚持最高标准
领导者坚持不懈地制定高标准，许多人可能认为这些标准过高。领导者不断提高标准，推动他们的团队提供高质量的产品，服务和流程。领导者确保缺陷不会下传，并且问题已得到解决，因此问题得以解决。

思考大处
小处思考是一种自我实现的预言。领导者创造并传达大胆的方向以激发成果。他们有不同的想法，四处寻找服务客户的方式。

行动
速度偏差在企业中至关重要。许多决定和行动是可逆的，不需要大量研究。我们重视计算的风险承担。

节俭

事半功倍。约束会滋生足智多谋，自给自足和发明创造。增加员工人数，预算规模或固定费用没有额外的要点。

赢得信任
领导者要专心倾听，坦率说话并尊重他人。他们口头批评自己，即使这样做尴尬或尴尬。领导者不相信他们或他们团队的体味有香水味。他们将自己和他们的团队与最佳水平进行比较。

Dive Deep
Leaders 在各个级别运作，与细节保持联系，经常审核，并且对度量和轶事有所不同时持怀疑态度。他们下面没有任务。

有骨干，异议和承诺
领导人有义务在不同意时对决策提出挑战，即使他们感到不舒服或疲惫不堪。领导者有信念并且坚韧。他们不会为了社会凝聚力而妥协。一旦决定，他们将全权负责。

交付成果
领导者专注于其业务的关键输入，并以适当的质量和及时的方式交付它们。尽管遭受了挫折，但他们还是挺身而出，从不停歇。