ind all duplicate files in current directory and sub directory
---
// Question: given a file path, find duplicate files in the file system and list their paths in List<List<String>>
// Duplicate files: files with the same content but different file names
// Example:
// root/a 1.txt("abcd") 2.txt("efgh")
// root/c 3.txt("abcd")
// root/c/d 4.txt("efgh")
// Output: [[root/a/1.txt, root/c/3.txt], [root/a/2.txt, root/c/d/4.txt]]

// Idea: Your solution needs to be tackle a couple of problems: obtaining a list of all the files in the file system (e.g. via DFS), binning the lists into . 1point 3acres 璁哄潧
// possible matches, repeat via swappable heuristics until your certainty is 100%. (eg size 1st, md5 2nd, byte stream 3rd)
// 1. parse the whole file system
// File class method: boolean isDirectory(), boolean isFile(), String[] list(), long length(), read(buffer, offset, size)
// 2. Binning the lists into possible matches
// 1) hashing: MD5, SHA1, SHA256
// 2) metadata: file size; the first 1kb of data, the second 1kb of data....

```java



    class Solution {
    class FileKey {
        Long fileSize;
        String shaB0;
        FileKey(Long fileSize, String shaB0) {
            this.fileSize = fileSize;
            this.shaB0 = shaB0;
        }

        @Override
        public int hashCode() {
            return fileSize.hashCode() + 33 * shaB0.hashCode();
        }

        @Override
        public boolean equals(Object fk) {
            if (fk == null || !(fk instanceOf FileKey)) return false;
            FileKey _fk = (FileKey) fk;
            if (this.fileSize != _fk.fileSize || !this.shaB0.equals(_fk.shaB0)) return false;
            return true;
        }
    }

    class FileSuperKey extends FileKey{
        String shaAll;
        FileKey(Long fileSize, String shaB0, String shaAll) {
            super(fileSize, shaB0);
            this.shaAll = shaAll;
        }

        @Override
        public int hashCode() {
            int res = super.hashCode();
            res += shaAll.hashCode();
            return res;
        }

        @Override
        public boolean equals(Object fsk) {
            if (fsk == null || !(fsk instanceOf FileSuperKey)) return false;
            FileSuperKey _fsk = (FileSuperKey) fsk;
            if (this.fileSize != _fsk.fileSize || !this.shaB0.equals(_fsk.shaB0)) return false;
            if (!this.shaAll.equals(_fsk.shaAll)) return false;
            return true;
        }
    }

    private final static int BLOCK_SIZE = 64 * 1024;
    Map<Long， List<String>> fileSizeMap = new HashMap<>();
    Map<FileKey， List<String>> firstBlockMap = new HashMap<>();
    Map<FileSuperKey， List<String>> allBlockMap = new HashMap<>();
    List<List<String>> res = new ArrayList<>();


    static void compareFileSize(String path) {
        Stack<String> stack = new Stack<>();
        stack.push(path);
        while (stack.size() != 0) {
            String cur = stack.pop();
            File file = new File(cur);
            if (!file.exists()) continue;
            if(file.isDir()) {
                List<String> children = file.listDir();
                for(String child : children) {
                    compareFileSize(joinPath(cur, child));
                }
            } else if (file.isFile()){
                Long fileSize = file.length();;
                fileSizeMap.put(fileSize, fileSizeMap.getOrDefault(fileSize, new ArrayList<>()).add(cur));
            } else if (file.isSymbolicLink()) {
                continue;
            } else {

            }
        }

    }

    static void compareFileSize(String path) {
        Queue<String> queue = new LinkedList<>();
        queue.offer(path);
        while (queue.size() != 0) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                String cur = queue.poll();
                File file = new File(cur);
                if (!file.exists()) continue;
                if(file.isDir()) {
                    List<String> children = file.listDir();
                    for(String child : children) {
                        queue.offer(joinPath(cur, child));
                    }
                } else if (file.isFile()){
                    Long fileSize = file.length();;
                    fileSizeMap.put(fileSize, fileSizeMap.getOrDefault(fileSize, new ArrayList<>()).add(cur));
                } else if (file.isSymbolicLink()) {
                    continue;
                } else {
//devices
                }
            }
        }
    }


    static void compareFirstBlock() {
        for (Map.Entry<Long， List<String>> entry : fileSizeMap) {
            List<String> paths = entry.getValue();
            if (paths.size() <= 1) continue;
            for (String path : paths) {
                String shaB0 = checkSumB0(path);
                FileKey fk = new FileKey(entry.getKey(), shaB0);
                firstBlockMap.put(fk, firstBlockMap.getOrDefault(fk, new ArrayList<>()).add(path));
            }
        }
    }


    static void compareAllBlock() {
        for (Map.Entry<FileKey， List<String>> entry : firstBlockMap) {
            List<String> paths = entry.getValue();
            Set<String> matched = new HashSet<>();
            if (paths.size() <= 1) continue;
            for (String path : paths) {
                String shaAll = checkSumAll(path);
                FileSuperKey fsk = new FileSuperKey(entry.getKey(), shaB0, shaAll);
                allBlockMap.put(fsk, allBlockMap.getOrDefault(fsk, new ArrayList<>()).add(path));
            }
        }
    }

    static void compareBytes() {
        for (Map.Entry<FileSuperKey， List<String>> entry : allBlockMap) {
            List<String> paths = entry.getValue();
            Set<String> matched = new HashSet<>();
            if (paths.size() <= 1) continue;
            for (int i = 0; i < paths.size(); i++) {
                String path0 = paths.get(i);
                if (matched.contains(path0)) continue;
                List<String> local = new ArrayList<>();
                local.add(path0);
                for (int j = i + 1;  j < paths.size(); j++) {
                    String path1 = paths.get(j);
                    if (matched.contains(path1)) continue;
                    if (checkBytes(path0, path1) {
                        matched.add(path1);
                        local.add(path1);
                    }
                }
                res.add(local);
            }
        }
    }

    static String checkSymbLinkLoop(File file) {

    }

    static String checkSumB0(String path) {
        MessageDigest md = MessageDigest.getInstance("SHA-512");
        FileInputStream fileInput = new FileInputStream(new File(path));
        byte[] fileData = new byte[BLOCK_SIZE];
        int read = 0;
        if ((read = fileInput.read(fileData)) > 0) {
            fileInput.read(fileData);
            messageDigest = MessageDigest.getInstance("SHA-512");
            String fileHash = new BigInteger(1, messageDigest.digest(fileData)).toString(16);
        }
        fileInput.close();
        return fileHash;
    }

    static String checkSumAll(String path) {
        MessageDigest md = MessageDigest.getInstance("SHA-512");
        FileInputStream fileInput = new FileInputStream(new File(path));
        byte[] fileData = new byte[BLOCK_SIZE];
        int read = 0;
        while((read = fileInput.read(fileData)) > 0) {
            fileInput.read(fileData);
            messageDigest = MessageDigest.getInstance("SHA-512");
            String fileHash = new BigInteger(1, messageDigest.digest(fileData)).toString(16);
        }
        fileInput.close();
        return fileHash;
    }

    static boolean checkBytes(String path0, String path1) {
        return FileUtils.contentEquals(new File(path0), new File(path1));
    }


    public static boolean checkBytes(String path1, String path2) {
        File f1=new File(path1);
        File f2=new File(path2);
        FileInputStream fis1 = new FileInputStream (f1);
        FileInputStream fis2 = new FileInputStream (f2);
        int n=0;
        byte[] b1 = new byte[BLOCK_SIZE];
        byte[] b2 = new byte[BLOCK_SIZE];
        while (fis1.available() > 0) {
            fis1.read(b1);
            fis2.read(b2);
            if (Arrays.equals(b1,b2)==false) return false;
        }
        return true;
    }

    static List<List<String>> findDuplicates(String path) {
        if (path == null || !Files.exists(path)) return res;
        compareFileSize(path);
        compareFirstBlock();
        compareAllBlock();
        compareBytes();
        return res;
    }
}
```

Follow ups:
1. Different files types, like symblokc link (dead loop)
如何确定这是个regular文件，是一个符号链接，是一个devices，还是目录？这是在遍历时候的一个必要步骤
symbolic 直接跳过   is_regular_file(), is_block_file()
如何确定文件类型？遍历目录文件时候，如何确定这是个regular文件，是一个符号链接，是一个特殊的设备文件，还是目录？这是在遍历时候的一个必要步骤。不同的语言和操作系统下可能有不同的方法。如果能够谈到linux下文件系统下有多种文件类型，Linux提供了多种API，比如is_regular_file(), is_block_file等保证确定文件类型，应该是加分项。

2. 如果死循环了为什么 怎么办 soft link 说删除link 不行 用hashset visited把树的问题转换成图的问题 需要实现

3. DFS vs BFS
DFS:
Case: it is very flat and lots of files in each levels, DFS may be better
lots of data centers have very flat layout, GFS

BFS:
If it is very deep and not much files or almost same files in every level, BFS is better and BFS can reduce file system access 一般文件系统倾向于把同一目录下的文件放在一起，这样BFS就可以显示出优势，可以减少硬盘读次数 read ahead

3. 如果目录很深怎么办？这个需要考虑，用来存储文件路径名的字符数组是否有溢出的可能性

4. 文件规模巨大的情况下，就是多机分布式查询。这个涉及到system design方面了 Map Reduce，Map: path -> <fileKey, path>

6. 查询过程中司机, 加入checking point

7. hash function在什么情况下efficient，什么情况下很垃圾。。hash collisions
Md5, shsa1 are fast but unsecure, sha256 and sha512 are better

15. hash function在什么情况下efficient，什么情况下很垃圾。。

# game of life

```java
class Solution {
    public void gameOfLife(int[][] board) {
        if (board == null || board.length == 0) return;
        int m = board.length, n = board[0].length;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                int lives = liveNeighbors(board, m, n, i, j);
                // In the beginning, every 2nd bit is 0;
                // So we only need to care about when will the 2nd bit become 1.
                // Any live cell with two or three live neighbors lives on to the next generation.
                if (board[i][j] == 1 && lives >= 2 && lives <= 3) {
                    board[i][j] = 3; // Make the 2nd bit 1: 01 ---> 11
                }
                //Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.
                if (board[i][j] == 0 && lives == 3) {
                    board[i][j] = 2; // Make the 2nd bit 1: 00 ---> 10
                }
            }
        }
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                board[i][j] >>= 1;  // Get the 2nd state.
            }
        }
    }

    int[] dxs = new int[]{-1, -1, -1, 0, 0, 1, 1, 1};
    int[] dys = new int[]{-1, 0, 1, -1, 1, -1, 0, 1};
    public int liveNeighbors(int[][] board, int m, int n, int x, int y) {
        int lives = 0;
        for (int i = 0; i < 8; i++) {
            int nx = x + dxs[i], ny = y + dys[i];
            if (nx < 0 || ny < 0 || nx == m || ny == n) continue;
            lives += board[nx][ny] & 1;
        }
        return lives;
    }
}


class Solution {
    public void gameOfLife(char[][] board) {
        if (board == null || board.length == 0) return;
        int m = board.length, n = board[0].length;
        char[][] temp = new char[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                int lives = liveNeighbors(board, m, n, i, j);
                if (board[i][j] == '+' && lives >= 2 && lives <= 3) {
                    temp[i][j] = '+';
                } else if (board[i][j] == '-' && lives == 3) {
                    temp[i][j] = '+';
                } else temp[i][j] = '-';
            }
        }
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                board[i][j] = temp[i][j];
            }
        }
    }

    int[] dxs = new int[]{-1, -1, -1, 0, 0, 1, 1, 1};
    int[] dys = new int[]{-1, 0, 1, -1, 1, -1, 0, 1};
    public int liveNeighbors(char[][] board, int m, int n, int x, int y) {
        int lives = 0;
        for (int i = 0; i < 8; i++) {
            int nx = x + dxs[i], ny = y + dys[i];
            if (nx < 0 || ny < 0 || nx == m || ny == n) continue;
            lives += board[nx][ny] == '+' ? 1 : 0;
        }
        return lives;
    }
}


class Solution {


// return null if read to the end of file
  // otherwise return the next row of game status
  // as an interger array with 0 and 1
  int[] readRow();

  // write an integer array to the file
  // return true if success
  // otherwise return false
  boolean writeRow(int[]);
  int[] pre = null, cur = null, next = null;
  int[] row = null;
  void gameOfLife() {
    while((row = readRow())!=null) {
      if(cur == null) {
        cur = row;
        continue;
      }
      if(next == null) {
        next = row;
      }

      if(pre == null) {
        int[][] nextStateBoard = gameOfLife(new int[]{cur, next});
        writeRow(nextStateBoard[0]);
      } else {
        int[][] nextStateBoard = gameOfLife(new int[]{pre, cur, next});
        writerow(nextStateBoard[1]);
      }
      pre = cur;
      cur = next;
      next = null;
    }
    int[][] nextStateBoard = gameOfLife(new int[]{pre, cur});
    writeRow(nextStateBoard[1]);
  }
}
```

Follow - up:
1.follow up问如果board太大怎么办，我说那就每次读三行，然后他给我一个API让我写改进之后的code。
follow up2问如果board太大，一个机器放不下怎么办。我说那就存distributed file system，然后他说那我问你一个extra credit吧，如果用MapReduce的话这题该怎么做。
follow up 1: 读一行，写回去一行
follow up 2: 每个机器负责一行的改写，一行的改写需要上下两行的context。最后每个机器合并写入到最后的一个表中
<I, array[]>

# highest minimum sharpness

```java
class Solution {
  public int minSharp(int[][] matrix) {
    if (matrix == null || matrix.length == 0 || matrix[0].length == 0) return 0;
    int m = matrix.length, n = matrix[0].length;
    int[][] dp = new int[m][n];
    for (int i = 0; i < n; i++) {
      dp[i][0] = matrix[i][0];
```

~~~
    }
    for (int j = 1; j < n; j++) {
      for (int i = 0; i < m; i++) {
        int up = i - 1 >= 0 ? dp[i - 1][j - 1] : Integer.MIN_VALUE, down = i + 1 < m ? dp[i + 1][j - 1] :
Integer.MIN_VALUE;
        dp[i][j] = matrix[i][j] + Math.max(dp[i][j - 1], Math.max(up, down));
      }
    }
    int res = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
      res = Math.max(res, dp[i][n - 1]);
    }
    return res;
  }
}
```
~~~

followup和之前的面经一样，问的是如果是100million * 100 million怎么办。因为看过面经，我先回答的是答案是把这个matrix翻转90度，然后一行行处理，但翻转的时候，读行输出列会有硬盘写文件耗时，读列输出行会有硬盘读文件耗时。

# hit counter

~~~
```java



long getTimeStamp() {
    return System.currentTimeMillis() / 1000;
}
写多
class HitCounter {

        Queue<Integer> q = null;
        /** Initialize your data structure here. */
        public HitCounter() {
            q = new LinkedList<Integer>();
        }



        /** Record a hit.
            @param timestamp - The current timestamp (in seconds granularity). */
        public void hit(int timestamp) {
            q.offer(timestamp);
        }

        /** Return the number of hits in the past 5 minutes.
            @param timestamp - The current timestamp (in seconds granularity). */
        public int getHits(int timestamp) {
            while(!q.isEmpty() && timestamp - q.peek() >= 300) {
~~~

```
                q.poll();
            }
            return q.size();
        }
}

class HitCounter {
    Map<Integer, Integer> map = new HashMap<>();
    /** Initialize your data structure here. */
    public HitCounter() {
    }


    public void hit(int timestamp) {
        // int timestamp = System.currentTimeMillis();
        map.put(timestamp, map.getOrDefault(timestamp, 0) + 1);
    }

    /** Return the number of hits in the past 5 minutes.
    @param timestamp - The current timestamp (in seconds granularity). */
    public int getHits(int timestamp) {
        // int timestamp = System.currentTimeMillis();
        int res = 0;
//synchronized(map){
        Iterator<Integer> it = map.keySet().iterator();
        while (it.hasNext()) {
            Integer key = it.next();
            if (timestamp - key >= 300) it.remove();
            else res += map.get(key);
        }
//}
        return res;
    }

}


//读多的话 把求和放在getHits
//
public class HitCounter {
    public static final int TIME_WINDOW = TIME_WINDOW;
    private int[] times;
    private int[] hits;
    /** Initialize your data structure here. */
    public HitCounter() {
        times = new int[TIME_WINDOW];
        hits = new int[TIME_WINDOW];
    }

    /** Record a hit.
        @param timestamp - The current timestamp (in seconds granularity). */
    public void hit(int timestamp) {
        int index = timestamp % TIME_WINDOW;
//synchronized(this){
        if (times[index] != timestamp) {
            times[index] = timestamp;
            hits[index] = 1;
        } else {
            hits[index]++;
        }
//}
    }

    /** Return the number of hits in the past 5 minutes.
        @param timestamp - The current timestamp (in seconds granularity). */
    public int getHits(int timestamp) {
        int total = 0;
//synchronized(this){
        for (int i = 0; i < TIME_WINDOW; i++) {
            if (timestamp - times[i] < TIME_WINDOW) {
                total += hits[i];
            }
        }
//}

        return total;
    }
}

```

特别多ollow up：一直hit怎么办，edge cases有哪些，怎么提高时间和空间复杂度 balabala....

1. Arraylist to store all data
2. Use queue
3. Use circular array

1. 优化存储空间： queue -> hashmap -> two array -
2. 多线程：
    1. Add Synchronized
    2. ConcurrentHashMap
    3. Thread Local and add lock at getHits

1. last hit和cur hit发生在同一秒
2. last hit和cur hit发生在不同秒, 然后把last hit到cur hit之间的element reset 0 检查结果是否正确的
3. 同2, 且last hit跟cur hit发生的间隔很大(ex. 30000s), 检查run time, 看是否在reset完300个element后就early return

精确度考量： queue
内存考量：用array

# phone combination
---
phone number + dictionary：leetcode 上的电话本问题的扩展。区别在于多了一个 dictionary，提供了一个函数 isWord（String word）返回 word 是否在 dictionary 中

```java
public class Solution {
    public List<String> letterCombinations(String digits) {
        if (digits == null || digits.length() == 0) return new ArrayList<>();
        String [] map=new String[]{"0","1","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"};
        Queue<String> res =new LinkedList<>();
        res.add("");
        for (int i = 0; i < digits.length(); i++) {
            int digit = digits.charAt(i) - '0';
            String cs = map[digit];
            int size = res.size();
            for (int j = 0; j < size; j++) {
                String curr = res.poll();
                for (int k = 0; k < cs.length(); k++) {
                    String next = curr + cs.charAt(k);
                    res.offer(next);
                }
            }
        }
        return new ArrayList<>(res);
    }
}

public List<String> wordBreak(String s, List<String> wordDict) {
        Trie trie = new Trie();
        for (String word : wordDict) {
            trie.insert(word);
        }
        int m = s.length();
        boolean[][] valid = new boolean[m + 1][m + 1];
        boolean[] f = new boolean[m + 1];
        f[0] = true;
        Set<String> set = new HashSet<>(wordDict);
        for (int i = 1 ; i <= m; i++) {
            for (int j = i - 1; j >= 0; --j) {
                if (f[j] && set.contains(s.substring(j, i))) {
                    valid[j][i] = true;
                    f[i] = true;
                }
            }
        }
        List<String> res = new ArrayList<>();
        dfs(s, valid, m, new ArrayList<>(), res);
        return res;
    }

    private static void dfs(String s, boolean[][] prev, int cur, List<String> path, List<String> result) {
        if (cur == 0) {
            StringBuilder sb = new StringBuilder();
            for (int i = path.size() - 1; i >= 0; --i) sb.append(path.get(i)).append(' ');
            sb.deleteCharAt(sb.length()-1);
            result.add(sb.toString());
        }
        for (int i = 0; i < cur; ++i) {
            if (prev[i][cur]) {
                path.add(s.substring(i, cur));
                dfs(s, prev, i, path, result);
                path.remove(path.size()-1);
            }
        }
    }
```

follow up 1: 字典里的词组成的也可以，同时知道字典里的词最小长度是3。那么只有两种情况，3+4或者4+3.

follow up 2: 如果可以分成任意的词该怎么办。

follow up 3: 如果字典里的词是给定的，可以怎么优化。想法是用trie来prune一些

问了复杂度，问了优化方式，需要用 trie

# Insert Intervals

# 判断byte[] 是否在文件中

```java
public class Solution {
    /**
     * @param source a source string
     * @param target a target string
     * @return an integer as index
     */
    int BASE = 1000000;
    int CHUNK_SIZE = 1024 * 1024;

    public int getHashCode(byte[] target) {
        int targetCode = 0;
        for (int i = 0; i < target.length; i++) {
            targetCode = (targetCode * 31 + (int)target[i]) % BASE;
        }
    }
    public boolean containBytes(byte[] target, String path) {
        File file = new File(path);
        if (!file.exists()) return false;
        if (target == null || target.length() == 0) return true;

        int targetCode = getHashCode(target);

        int power = 1;
        for (int i = 0; i < m; i++) {
            power = (power * 31) % BASE;
        }


        int m = target.length;
        int hashCode = 0;
        byte[] pre = new byte[CHUNK_SIZE], fileData = new byte[CHUNK_SIZE];
        int read = 0, base = 0, i = 0;
        while((read = fileInput.read(fileData)) > 0) {
            for (int i = 0; i < read; i++) {
                hashCode = (hashCode * 31 + (int)fileData[i]) % BASE;
                if (i + base < m - 1) continue;
                if (i >= m)
                    hashCode = (hashCode -  (int)fileData[i - m] * power)) % BASE;
                else
                    hashCode = (hashCode -  (int)pre[pre.length - m + i] * power)) % BASE;

                if (hashCode < 0) {
                    hashCode += BASE;
                }
                if (hashCode == targetCode) {
                    if (i >= m - 1 && new String(fileData, i + 1 - m, m).equals(new String(target))) {
                        return i + 1 - m + base;
                    } else if (i < m - 1 && new String(fileData, 0, i + 1) + new String(pre, pre.length + i - m + 1, m - i -
1).equals(new String(target))){
                        return i - m + base;
                    }

                }
            }
            pre = fileData;
            base += read;
        }
        return -1;
    }
}
```

```
class Solution {
    int BASE = 1000000;
    public int strStr2(String source, String target) {
        if (target == null || source == null) return -1;
        int  m = source.length(), n = target.length();
        if (m < n) return -1;
        if (n == 0) return 0;


        int hashTar = 0;
        for (int i = 0; i < n; i++) {
            hashTar = (hashTar * 31 % BASE + target.charAt(i)) % BASE;
        }

        int power = 1;
        for (int i = 0; i < n; i++) power = (power * 31) % BASE;

        int hashSou = 0;
        for (int i = 0; i < m; i++) {
            hashSou = (hashSou * 31 % BASE + source.charAt(i)) % BASE;
            if (i < n - 1) {
                continue;
            }

            if (i >= n)hashSou = (hashSou - source.charAt(i - n) * power) % BASE;
            if (hashSou < 0) hashSou += BASE;

            if (hashSou != hashTar) continue;
            if (target.equals(source.substring(i - n + 1, i + 1))) return i - n + 1;
        }
        return -1;
    }
}
```

读file的时候具体发生了什么？读一个大的array要的时间和一个小的array，哪个需要的时间多?
rolling hash

# token bucket

# Word Pattern

```java
class Solution {
    public boolean wordPatternMatch(String pattern, String str) {
        return helper(new HashMap<>(), pattern, 0, str, 0);
    }

    public boolean helper (Map<Character, String> map,
     String pattern, int startP, String str, int startS){
        if (startP == pattern.length() && startS == str.length())return true;
        if (startS == str.length())return false;
        if (startP == pattern.length())return false;
```

~~~
        char c = pattern.charAt(startP);
        if (map.containsKey(c)){
            String target = map.get(c);
            if (str.length() - startS < target.length()) return false;
            if (!str.substring(startS, startS + target.length()).equals(target)) return false;
            return helper(map, pattern, startP + 1, str,startS + target.length());
        }
        for (int i = startS + 1; i <= str.length();i++){
            String temp=str.substring(startS, i);
            if (map.containsValue(temp)) continue;
            map.put(c, temp);
            if(helper(map,pattern,startP + 1, str, i)) return true;
            map.remove(c);
        }
        return false;
    }
}
```
~~~

# Bit Torrent

~~~
```java
import java.io.*;
import java.util.*;

/*
* To execute Java, please define "static void main" on a class
* named Solution.
*
* If you need more classes, simply define them inline.
*/

class Range {
  int lower;
  int higher;

  /* Implementation omitted */
}

// 0, 1,2,3,. ..7
/* isFileDone([[3, 7), [0, 1), [2, 5), [6, 8)], 8) -> false */
/* isFileDone([[3, 7), [0, 2), [2, 5), [6, 8)], 8) -> true */
//blocks: [[0, 2), [2, 5), [3, 7),[6, 8)]

class Solution {
    public boolean isFileDone(List<Range> blocks, int size) {
        BisSet bs = new BisSet(size);
        for (Range block : blocks) {
            bs.set(block.lower, block.higher, true);
        }
        return bs.cardinality()
~~~

```
    }
}


class Solution {
    public boolean isFileDone(List<Range> blocks, int size) {
        PriorityQueue<Range> pq = new PriorityQueue<>((a, b) -> a.lower - b.lower);
        for (Range block : blocks) pq.add(block);
            if (pq.size() == 0) return size == 0;
        Range pre = pq.poll();
        while (pq.size() != 0) {
            Range cur = pq.poll();
            if (pre.higher < cur.lower) break;
            pre.higer = cur.higher;
        }
        return pre.lower == 0 && pre.higher == size;
    }
}

class Downloader {

    PriorityQueue<Range> pq;
    int size;

    public Downloader(int size) {
        this.pq = PriorityQueue<Range> pq = new PriorityQueue<>((a, b) -> a.lower - b.lower);
        this.size = size;
    }
    public void addBlock(Range r) {
        pq.add(r);
    }

    public boolean isDone() {
        if(pq.size == 0) return false;
        Range pre = pq.poll();
        while (pq.size() != 0) {
            Range cur = pq.poll();
            if (pre.higher < cur.lower) break;
            pre.higer = cur.higher;
        }
        pq.poll(pre);
        return pre.lower == 0 && pre.higher == size;
    }
}




class Downloader {

    List<Range> list;
    int size;
    public Downloader(int size) {
        this.list = new ArrayList<>();
        this.size = size;
    }
    public void addBlock(Range r) {
        Range rangeLo = binarySearch(r.lower);
        Range rangeHi = binarySearch(r.higher);
    }

    public Range binarySearch(int byteIndex) {
        int lo = 0, hi = list.size() - 1;
        while (lo + 1 < hi) {
            int mid = lo + (hi - lo) / 2;
            Range midRange = list.get(mid);
            if (midRange.lower > byteIndex) {
                hi = mid - 1;
            } else if (midRange.higher < byteIndex) {
                lo = mid + 1;
            } else {
                return midRange;
            }
        }
        if (rangeFit(lo, byteIndex)) return list.get(lo);
        if (rangeFit(hi, byteIndex)) return list.get(hi);
        return null;
    }

    public boolean rangeFit(int rangeIndex, int byteIndex) {
        Range range = list.get(rangeIndex);
        return byteIndex >= range.lower && byteIndex <= range.higher;
    }

    public boolean isDone() {
        if (list.size() != 1) return false;
        if (list.lower != 0 || list.higher != size) return false;
        return true;
    }
}


use BST
```

# Coins Change

```java
class Solution {
    public List<List<Integer>> buySoda(int[] coins, int target) {
        List<List<Integer>> res = new ArrayList<>();
        boolean[] dp = new boolean[target+1];
        for (int i = 0; i < m; i++) {
            for (int j = s[i]; j <= n; j++) {
                dp[j] = dp[j] || dp[j - coins[i]];
            }
        }
        dfs(res, new ArrayList<>(), coins, 0, target, dp);
        return res;
    }

    public void dfs(List<List<Integer>> res, List<Integer> cur int[] coins, int start, int target, boolean[] dp) {
        if (target == 0) {
            res.add(new ArrayList<>(cur));
            return;
        }

        int m = coins.length;
        for (int i = start; i < m; i++) {
            cur.add(coins[i]);
            int pre = target - coins[i];
            if (pre >= 0 && dp[pre]) {
                dfs(res, cur, coins, i, target - coins[i], dp);
            }
            cur.remove(cur.length() - 1);
        }
    }
}
```

# Web crawl

# Allocate / Deallocate ID

Queue -> O(1). O(n)
Bitmap -> O(n) O(1)

```java

public static int searchBit(boolean[][] matrix) {
    int curLevel = 0, m = matrix.length;
    int index = 0;
    if (matrix[0][0]) return -1;
    while (curLevel < m - 1) {
        if (!matrix[curLevel][index * 2]) {
            index = index * 2;
        } else {
            index = index * 2 + 1;
        }
        curLevel++;
    }
    return index;
}

public static void clearBit(boolean[][] matrix,  int index) {
    int curLevel = matrix.length - 1;
    while (curLevel >= 0) {
        matrix[curLevel][index] = false;
        index /= 2;
        curLevel--;
    }
}


public static void setBit(boolean[][] matrix, int index) {
    int curLevel = matrix.length - 1;
    matrix[curLevel][index] = true;
    while (curLevel > 0) {
        if (index % 2 == 1) {
            if (!matrix[curLevel][index - 1]) return;
            matrix[curLevel + 1][index / 2] = true;
        } else {
            if (!matrix[curLevel][index + 1]) return;
            matrix[curLevel + 1][index / 2] = true;
        }
        curLevel--;
    }
}
```

# Design Phone Dictionary

照片, 维持一个k大小的小顶堆，思路可以参考 李特抠 二九五，具体参看 https://instant.1point3acres.com/thread/199521.鏈□杓鍘熷垱鑷�

max photo count。 实现一个类，用户可以view一张相片（给id），然后也可以求被目前被view最多k个photos（多次调用）。 印象最深压力最大最自豪也是感觉面得最不好的一轮。主要因为是新题目，之前没彻底分析过。

当场给出两个可能，平衡树或者堆，表示堆容易实现view但是不好实现高性能get_most_k，平衡树容易实现容易实现get_most_k难实现view。面试官不断追问平衡树怎么实现update_key（view）。当时我是希望写出至少log n的update，又不想delete再insert（平衡树没练习很难写），然而一时半会又想不出更好的解法，回避这个问题大概两分钟。期间被追问了两次，感觉要扣分。都怪我没认真复习平衡树。

后来准备实用heap实现并尽量提升性能。讨论的过程中得到之用简单的priority queue加上swap的实现方法并且做到O(1) view, O(k) get_most_k，总体 O(n) space。不知道面试官有没有想到，反正之前我是没想到会有这样的解法，我跟自己解释了很久。开始写，因为这个算法比看上去难，而且一开始没设计好，中途多次修改代码，导致一些引用错误被面试官指出（我喜欢用python list 代替c struct 存储信息。因为没有关键字只有数字经常会引用错误），扣分。做出来后面试官要求我算多个操作的均摊时间，我说了一些简单的思想并表示我很久没算过已经不记得怎么算了，扣很多很多很多很多分。（他好像不知道什么是均摊时间，但我觉得他要我算的就是这个东西）感觉如果我挂了就是因为这个地方。

面试官的评价是：“...the probrom is in your general coding style, your code is hard to read”，后面也没跟我握手，我很伤心。这个算法已经是the best,现场想出来我觉得已经很牛逼了（我是渣不是大神）。后面写代码很紧张没能把代码完美，没想到载道这种地方。晚吃饭的时候发现代码可以写的更加精简，如果面试官发现了也要继续扣分。