1055. Shortest Way to Form String Premium

Medium Topics Companies Hint

A **subsequence** of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without c
relative positions of the remaining characters. (i.e., `"ace"` is a subsequence of `"abcde"` while `"aec"` is not).

Given two strings `source` and `target`, return *the minimum number of* ***subsequences*** *of* `source` *such that their concatenation equals* `targ
impossible, return `-1`.

Example 1:

```
Input: source = "abc", target = "abcbc"
Output: 2
Explanation: The target "abcbc" can be formed by "abc" and "bc", which are subsequences of source "a
```

Example 2:

```
Input: source = "abc", target = "acdbc"
Output: -1
Explanation: The target string cannot be constructed from the subsequences of source string due to t
"d" in target string.
```

Example 3:

```
Input: source = "xyz", target = "xzyxz"
Output: 3
Explanation: The target string can be constructed as follows "xz" + "y" + "xz".
```

Constraints:

- `1 <= source.length, target.length <= 1000`
- `source` and `target` consist of lowercase English letters.

Seen this question in a real interview before? 1/4

Yes No

Accepted 86.1K Submissions 143.4K Acceptance Rate 60.0%

Topics

Companies

Hint 1

Accept is not enough to get a hire. Interviewee 4 follow up

happyleetcode
1078 35507 Jul 09, 2019

first opinion is that we can use two pointer , one iterate src, another iterate tar.
for each tar char, we move j until src[j] == tar[i], if j == src.length, res++, j = 0;
in this solution, we greedy match as many chars from src to tar as possible which can lead mininum use of src.
and we can build a set to save all the char in src, if there exists a char from tar which not exists in set, return -1.

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    boolean[] map = new boolean[26];
    for (int i = 0; i < cs.length; i++)
        map[cs[i] - 'a'] = true;
    int j = 0, res = 1;
    for (int i = 0; i < ts.length; i++,j++) {
        if (!map[ts[i] - 'a']) return -1;
        while (j < cs.length && cs[j] != ts[i]) {
            j++;
        }
        if (j == cs.length) {
            j = -1;
            res++;
            i--;
        }
    }
    return res;
}
```

follow up 1: yes, correct. could u implement it with O 1 space, which mean without set.

okay. without set, we need a way to make sure there is a char which not in src. we can iterate src completely. if the j not move, then we can return -1.

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int res = 0;
    for (int i = 0; i < ts.length; ) {
```

```
    return res;
}
```

follow up 1: yes, correct. could u implement it with O 1 space, which mean without set.

okay. without set, we need a way to make sure there is a char which not in src. we can iterate src completely. if the j not move, then we can return -1.

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int res = 0;
    for (int i = 0; i < ts.length; ) {
        int oriI = i;
        for (int j = 0; j < cs.length; j++) {
            if (i < ts.length && cs[j] == ts[i])
                i++;
        }
        if (i == oriI) return -1;
        res++;
    }
    return res;
}
```

follow up 2: fine. what's the time complexity for above solutions. O(MN). could u make it better?

the time complexity is better than O (MN), should be O(logM * N) or O (N)
to find a logM way, it is easy to think of binary search. for each char in tar, we need loop from j to end, to find a char same as tar[i].
we can build a map which key is from 'a' -> 'z', the value is idx for this char in src. because idx is add from small to big. when we iterate tar[i], we can easily to find the tar[i]'s idx list. to search is there a idx is larger or equal than j+1. it is logM. and we have N char in tar, so the time complexity is N * logM
the time is to build the map is O(M);

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int res = 1;
    List<Integer>[] idx = new List[26];
    for (int i = 0; i < 26; i++) idx[i] = new ArrayList<>();
    for (int i = 0; i < cs.length; i++) idx[cs[i] - 'a'].add(i);
```

follow up 2: fine. what's the time complexity for above solutions. O(MN). could u make it better?

the time complexity is better than O (MN), should be O(logM * N) or O (N)
to find a logM way, it is easy to think of binary search. for each char in tar, we need loop from j to end, to find a char same as tar[i].
we can build a map which key is from 'a' -> 'z', the value is idx for this char in src. because idx is add from small to big. when we iterate tar[i], we can easily to find the tar[i]'s idx list. to search is there a idx is larger or equal than j+1. it is logM. and we have N char in tar, so the time complexity is N * logM
the time is to build the map is O(M);

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int res = 1;
    List<Integer>[] idx = new List[26];
    for (int i = 0; i < 26; i++) idx[i] = new ArrayList<>();
    for (int i = 0; i < cs.length; i++) idx[cs[i] - 'a'].add(i);
    int j = 0;
    for (int i = 0; i < ts.length; ) {
        List<Integer> tar = idx[ts[i] - 'a'];
        if (tar.isEmpty()) return -1;
        int k = Collections.binarySearch(tar,j);
        if (k < 0) k = -k - 1;
        if (k == tar.size()) {
            res++;
            j = 0;
        } else {
            j = tar.get(k) + 1;
            i++;
        }

    }
    return res;
}
```

follow up 3: great. could u improve it more?

so we have to think a solution which is O(N), how should we use O(1) to know the next J pos? maybe we can use more to save time.
in binary search solution we will have a map like a ->{1,3,7,16} (total src length is 20), so we need

follow up 3: great. could u improve it more?

so we have to think a solution which is O(N), how should we use O(1) to know the next J pos? maybe we can use more to save time.
in binary search solution we will have a map like a ->{1,3,7,16} (total src length is 20), so we need binary search.
if we can flatten them, i mean for each pos in 20 length, we just save the next idx, we can use O 1 to find the next J.
a -> {1,1,3,3,7,7,7,7,16,16,16,16,16,16,16,16,16,0,0,0}
for example if now j is 4, we can just check map[4] = 7; we know 7 pos have an 'a', so next j will be 7 + 1.
if now j is 17, we get map[17] = 0, we know there is no more j after. so j = 0, res++;
the time complexity is O (N) , and build the map cost 26 * M

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int[][] idx = new int[26][cs.length];
    for (int i = 0; i < cs.length; i++) idx[cs[i] - 'a'][i] = i + 1;
    for (int i = 0; i < 26; i++) {
        for (int j = cs.length - 1, pre = 0; j >= 0; j--) {
            if (idx[i][j] == 0) idx[i][j] = pre;
            else pre = idx[i][j];
        }
    }
    int res = 1, j = 0;
    for (int i = 0; i < ts.length; i++) {
        if (j == cs.length) {
            j = 0;
            res++;
        }
        if (idx[ts[i] - 'a'][0] == 0) return -1;
        j = idx[ts[i] - 'a'][j];
        if (j == 0 ) {
            res++;
            i--;
        }
    }
    return  res;
}
```

follow up 4: cool, if we assume which can copy a array to another array with 26 length in constant time. could u implement it with O(M + N)

follow up 4: cool, if we assume which can copy a array to another array with 26 length in constant time. could u implement it with O(M + N)

it sounds like we need switch the map from [26][src.length] to [src.length][26].
and we also need to use O 1 time to know what's next j position.
now we are in the 2rd idx (j = 1), so tar[i] = 'a' we should save the map[1]['a'] the next position of j.
if we are in the last idx, i think the map should be all 0, except the last tar char. for example the char is 'a'
so the map for it is [last+1,0,0,...,0]
how about last -1 idx, if the tar[last - 1] is same as tar[last], we just update it , [last - 1 + 1, 0....0]
if is not same. we can update a 0 with last - 1 + 1

```
public int shortestWay(String source, String target) {
    char[] cs = source.toCharArray(), ts = target.toCharArray();
    int[][] idx = new int[cs.length][26];
    idx[cs.length - 1][cs[cs.length - 1] - 'a'] = cs.length;
    for (int i = cs.length - 2; i >= 0; i--) {
        idx[i] = Arrays.copyOf(idx[i + 1],26);
        idx[i][cs[i] - 'a'] = i + 1;
    }
    int j = 0, res = 1;
    for (int i = 0; i < ts.length; i++) {
        if (j == cs.length) {
            j = 0;
            res++;
        }
        j = idx[j][ts[i] - 'a'];
        if (idx[0][ts[i] - 'a'] == 0) return -1;
        if (j == 0) {
            res++;
            i--;
        }
    }
    return res;
}
```

Next
Python O(M + N*logM) using inverted index + binary search (Similar to LC 792)

1060. Missing Element in Sorted Array Premium

Medium Topics Companies Hint

Given an integer array `nums` which is sorted in **ascending order** and all of its elements are **unique** and given also an integer `k`, return the starting from the leftmost number of the array.

Example 1:

```
Input: nums = [4,7,9,10], k = 1
Output: 5
Explanation: The first missing number is 5.
```

Example 2:

```
Input: nums = [4,7,9,10], k = 3
Output: 8
Explanation: The missing numbers are [5,6,8,...], hence the third missing number is 8.
```

Example 3:

```
Input: nums = [1,2,4], k = 3
Output: 6
Explanation: The missing numbers are [3,5,6,7,...], hence the third missing number is 6.
```

Constraints:

- `1 <= nums.length <= 5 * 10^4`
- `1 <= nums[i] <= 10^7`
- `nums` is sorted in **ascending order,** and all the elements are **unique**.
- `1 <= k <= 10^8`

Follow up: Can you find a logarithmic time complexity (i.e., `O(log(n))`) solution?

Facebook Run Submit

Description Solutions Submissions Editorial

Code

Java Auto

```java
class Solution {

// Let missingNum be amount of missing number in the array. Two cases that need to be handled:

// missingNum < k, then return nums[n - 1] + k - missingNum
// missingNum >= k, then use binary search(during the search k will be updated) to find the index in the array, where

    public int missingElement(int[] nums, int k) {
       int n = nums.length;
        int l = 0;
        int h = n - 1;
        int missingNum = nums[n - 1] - nums[0] + 1 - n;

        if (missingNum < k) {
            return nums[n - 1] + k - missingNum;
        }

        while (l < h - 1) {
            int m = l + (h - l) / 2;
            int missing = nums[m] - nums[l] - (m - l);

            if (missing >= k) {
          // when the number is larger than k, then the index won't be located in (m, h]
                h = m;
            } else {
          // when the number is smaller than k, then the index won't be located in [l, m), update k -= missing
                k -= missing;
                l = m;
            }
        }

        return nums[l] + k;
    }
}
```

Saved to cloud

Testcase Test Result

Case 1 Case 2 Case 3

nums =

[4,7,9,10]

k =

1120. Maximum Average Subtree Premium

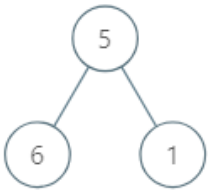
Medium Topics Companies Hint

Given the `root` of a binary tree, return *the maximum* ***average*** *value of a* ***subtree*** *of that tree*. Answers within `$10^{-5}$` of the actual answer will

A **subtree** of a tree is any node of that tree plus all its descendants.

The **average** value of a tree is the sum of its values, divided by the number of nodes.

Example 1:




```
Input: root = [5,6,1]
Output: 6.00000
Explanation:
For the node with value = 5 we have an average of (5 + 6 + 1) / 3 = 4.
For the node with value = 6 we have an average of 6 / 1 = 6.
For the node with value = 1 we have an average of 1 / 1 = 1.
So the answer is 6 which is the maximum.
```

Example 2:

```
Input: root = [0,null,1]
Output: 1.00000
```

Constraints:

- The number of nodes in the tree is in the range `[1, $10^4$]`.
- `0 <= Node.val <= $10^5$`

Seen this question in a real interview before? 1/4

Yes No

Accepted 67.8K Submissions 102K Acceptance Rate 66.5%

Topics

[Java] 8 line clean recursive code w/ brief comment and analysis.

rock
24077 8697 Jul 13, 2019

```
lic double maximumAverageSubtree(TreeNode root) {
 return helper(root)[2];

vate double[] helper(TreeNode n) {
 if (n == null)  // base case.
     return new double[]{0, 0, 0}; // sum, count  & average of nodes
 double[] l = helper(n.left), r = helper(n.right); // recurse to children.
 double child = Math.max(l[2], r[2]); // larger of the children.
 double sum = n.val + l[0] + r[0], cnt = 1 + l[1] + r[1]; // sum & count of subtree rooted at
 double maxOfThree = Math.max(child, sum  / cnt); // largest out of n and its children.
 return new double[]{sum, cnt, maxOfThree};
```

Analysis:

Time & space: O(n), n is the # of nodes in the tree.

Next
[Python] Recursion

Comments (8)

Sort by: Best

Type comment here... (Markdown supported)

Preview Comment

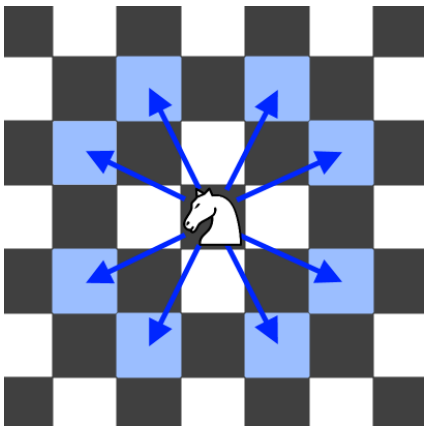
lioncruise Feb 21, 2020

1197. Minimum Knight Moves Premium

Medium Topics Companies Hint

In an **infinite** chess board with coordinates from `-infinity` to `+infinity`, you have a **knight** at square `[0, 0]`.

A knight has 8 possible moves it can make, as illustrated below. Each move is two squares in a cardinal direction, then one square in an orth



Return *the minimum number of steps needed to move the knight to the square* `[x, y]`. It is guaranteed the answer exists.

Example 1:

```
Input: x = 2, y = 1
Output: 1
Explanation: [0, 0] → [2, 1]
```

Example 2:

```
Input: x = 5, y = 5
Output: 4
Explanation: [0, 0] → [2, 1] → [4, 2] → [3, 4] → [5, 5]
```

Constraints:

- `-300 <= x, y <= 300`
- `0 <= |x| + |y| <= 300`

Facebook Run Submit

Description Solutions Submissions Editorial

Code

Java Auto

```java
class Solution {
    public int minKnightMoves(int x, int y) {
      x=Math.abs(x);
      y=Math.abs(y);
      int[][] offsets = {{1, 2}, {2, 1}, {2, -1}, {1, -2},
                {-1, -2}, {-2, -1}, {-2, 1}, {-1, 2}};

        Set<String> visited = new HashSet<>();
        visited.add("0,0");

        Deque<int[]> queue = new LinkedList<>();
        queue.addLast(new int[]{0, 0});
        int steps = 0;

        while (queue.size() > 0) {
            int currLevelSize = queue.size();
            // iterate through the current level
            for (int i = 0; i < currLevelSize; i++) {
                int[] curr = queue.removeFirst();
                if (curr[0] == x && curr[1] == y) {
                    return steps;
                }

// The key thing to note here is
// x = Math.abs(x);
// y = Math.abs(y);

// Here we are forcing the original co-ordinates to be in 1st Quadrant only. ( since we can use symmetry )

// you cannot reach from 0,0 to 1,1 using only 1st quadrant. hence we allow x >=-1 y>=-1
//instead of x>=0, y>=0 limit

                for (int[] offset : offsets) {
                    int[] next = new int[]{curr[0] + offset[0], curr[1] + offset[1]};
                    // align the coordinate to the bitmap
                    if (!visited.contains(next[0]+","+next[1]) && next[0]>=-1 && next[1]>=-1) {
                      queue.add(new int[]{next[0],next[1]});
                      visited.add(next[0]+","+next[1]);
                    }
                }

            }
            steps++;
        }
        return steps;
    }


}
```

Saved to cloud

Testcase Test Result

1213. Intersection of Three Sorted Arrays Premium

Easy Topics Companies Hint

Given three integer arrays `arr1`, `arr2` and `arr3` **sorted** in **strictly increasing** order, return a sorted array of **only** the integers that appear

Example 1:

```
Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]
Output: [1,5]
Explanation: Only 1 and 5 appeared in the three arrays.
```

Example 2:

```
Input: arr1 = [197,418,523,876,1356], arr2 = [501,880,1593,1710,1870], arr3 = [521,682,1337,1395,176
Output: []
```

Constraints:

- `1 <= arr1.length, arr2.length, arr3.length <= 1000`
- `1 <= arr1[i], arr2[i], arr3[i] <= 2000`

Seen this question in a real interview before? 1/4

Yes No

Accepted 84.3K Submissions 105.7K Acceptance Rate 79.8%

Topics

Companies

Hint 1

Hint 2

Similar Questions

Discussion (2)

Simple Java solution - beats 100%

harsh_vardan
725 8785 Oct 05, 2019

Java

```
public List<Integer> arraysIntersection(int[] arr1, int[] arr2, int[] arr3) {

    List<Integer> result = new ArrayList<>();

    int i = 0;
    int j = 0;
    int k = 0;

    while (i < arr1.length && j < arr2.length && k < arr3.length) {
        if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
            result.add(arr1[i]);
            i++;
            j++;
            k++;
        } else if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr2[j] < arr3[k]) {
            j++;
        } else k++;
    }

    return result;

}
```

Next
Java O(n) solution using 3 pointers

1216. Valid Palindrome III Premium

Hard Topics Companies Hint

Given a string `s` and an integer `k`, return `true` if `s` is a `k`**-palindrome**.

A string is `k`**-palindrome** if it can be transformed into a palindrome by removing at most `k` characters from it.

Example 1:

```
Input: s = "abcdeca", k = 2
Output: true
Explanation: Remove 'b' and 'e' characters.
```

Example 2:

```
Input: s = "abbababa", k = 1
Output: true
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consists of only lowercase English letters.
- `1 <= k <= s.length`

Facebook Run Submit

Description Solutions Submissions Editorial

Code

Java Auto

```java
class Solution {

    public boolean isValidPalindrome(String s, int k) {
        Integer[][] cache = new Integer[s.length()][s.length()];
        return aux(s, 0, s.length()-1, cache) <= k;
    }

    private int aux(String s, int left, int right, Integer[][] cache) {
        if (right - left < 1) return 0;
        if (cache[left][right] != null) return cache[left][right];

        int step = 0;
        if (s.charAt(left) == s.charAt(right)) {
            step = aux(s, left+1, right-1, cache);
        } else {
            step = 1 + Math.min(aux(s, left+1, right, cache), aux(s, left, right-1, cache));
        }
        cache[left][right] = step;
        return step;
    }

}
```

Saved to cloud

Testcase Test Result

Case 1 Case 2

s =

"abcdeca"

k =

1236. Web Crawler Premium

Medium Topics Companies Hint

Given a url `startUrl` and an interface `HtmlParser`, implement a web crawler to crawl all links that are under the **same hostname** as `star`

Return all urls obtained by your web crawler in **any** order.

Your crawler should:

- Start from the page: `startUrl`
- Call `HtmlParser.getUrls(url)` to get all urls from a webpage of given url.
- Do not crawl the same link twice.
- Explore only the links that are under the **same hostname** as `startUrl`.

As shown in the example url above, the hostname is `example.org`. For simplicity sake, you may assume all urls use **http protocol** without a
For example, the urls `http://leetcode.com/problems` and `http://leetcode.com/contest` are under the same hostname, while urls `http://example.org/test` and `http://example.com/abc` are not under the same hostname.

The `HtmlParser` interface is defined as such:

```
interface HtmlParser {
  // Return a list of all urls from a webpage of given url.
  public List<String> getUrls(String url);
}
```

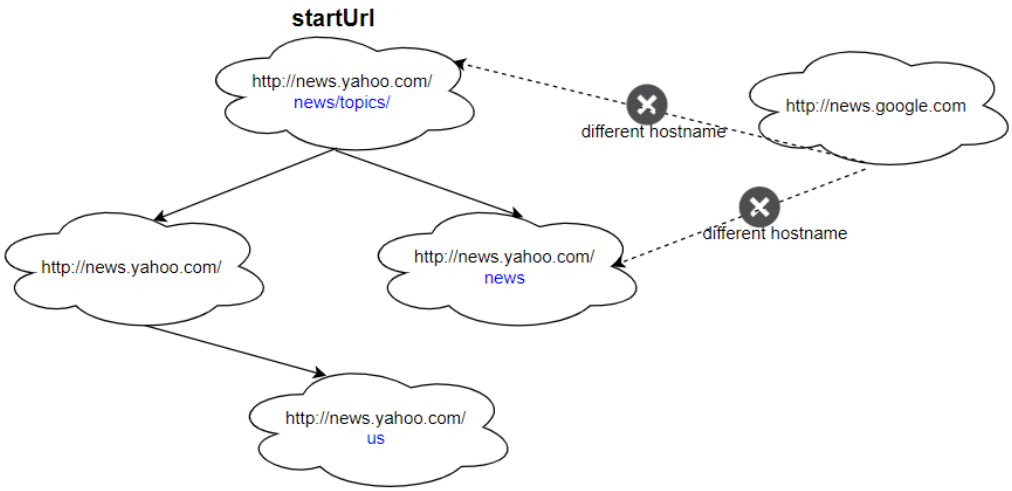
Below are two examples explaining the functionality of the problem, for custom testing purposes you'll have three variables `urls`, `edges` a
Notice that you will only have access to `startUrl` in your code, while `urls` and `edges` are not directly accessible to you in code.

Note: Consider the same URL with the trailing slash "/" as a different URL. For example, "http://news.yahoo.com", and "http://news.yahoo.co
urls.

Example 1:

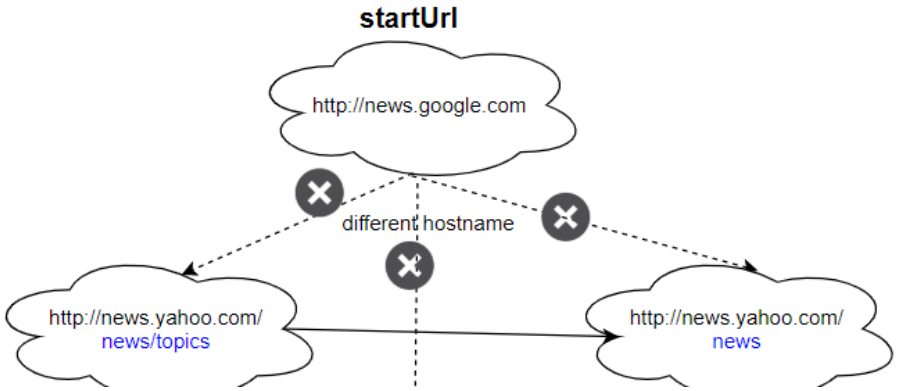
Note: Consider the same URL with the trailing slash "/" as a different URL. For example, "http://news.yahoo.com", and "http://news.yahoo.co
urls.

Example 1:




```
Input:
urls = [
  "http://news.yahoo.com",
  "http://news.yahoo.com/news",
  "http://news.yahoo.com/news/topics/",
  "http://news.google.com",
  "http://news.yahoo.com/us"
]
edges = [[2,0],[2,1],[3,2],[3,1],[0,4]]
startUrl = "http://news.yahoo.com/news/topics/"
Output: [
  "http://news.yahoo.com",
  "http://news.yahoo.com/news",
  "http://news.yahoo.com/news/topics/",
  "http://news.yahoo.com/us"
]
```

Example 2:



```
 * // This is the HtmlParser's API interface.
 * // You should not implement it, or speculate about its implementation
 * interface HtmlParser {
 *     public List<String> getUrls(String url) {}
 * }
 */
class Solution {
    public List<String> crawl(String startUrl, HtmlParser htmlParser) {
        Set<String> set = new HashSet<>();
        Queue<String> queue = new LinkedList<>();
        String hostname = getHostname(startUrl);

        queue.offer(startUrl);
        set.add(startUrl);

        while (!queue.isEmpty()) {
            String currentUrl = queue.poll();
            for (String url : htmlParser.getUrls(currentUrl)) {
                if (url.contains(hostname) && !set.contains(url)) {
                    queue.offer(url);
                    set.add(url);
                }
            }
        }

        return new ArrayList<String>(set);
    }

    private String getHostname(String Url) {
        String[] ss = Url.split("/");
        return ss[2];
    }
}
```

Next
Python BFS straightforward

1242. Web Crawler Multithreaded Premium

Medium Topics Companies

Given a URL `startUrl` and an interface `HtmlParser`, implement **a Multi-threaded web crawler** to crawl all links that are under the **same** `startUrl`.

Return all URLs obtained by your web crawler in **any** order.

Your crawler should:

- Start from the page: `startUrl`
- Call `HtmlParser.getUrls(url)` to get all URLs from a webpage of a given URL.
- Do not crawl the same link twice.
- Explore only the links that are under the **same hostname** as `startUrl`.

As shown in the example URL above, the hostname is `example.org`. For simplicity's sake, you may assume all URLs use **HTTP protocol** witl specified. For example, the URLs `http://leetcode.com/problems` and `http://leetcode.com/contest` are under the same hostname, whi `http://example.org/test` and `http://example.com/abc` are not under the same hostname.

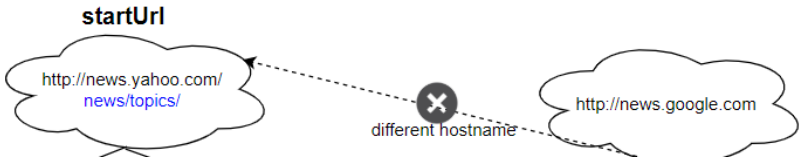
The `HtmlParser` interface is defined as such:

```
interface HtmlParser {
  // Return a list of all urls from a webpage of given url.
  // This is a blocking call, that means it will do HTTP request and return when this request is fir
  public List<String> getUrls(String url);
}
```

Note that `getUrls(String url)` simulates performing an HTTP request. You can treat it as a blocking function call that waits for an HTTP r guaranteed that `getUrls(String url)` will return the URLs within **15ms.** Single-threaded solutions will exceed the time limit so, can your r crawler do better?

Below are two examples explaining the functionality of the problem. For custom testing purposes, you'll have three variables `urls`, `edges` Notice that you will only have access to `startUrl` in your code, while `urls` and `edges` are not directly accessible to you in code.

Example 1:

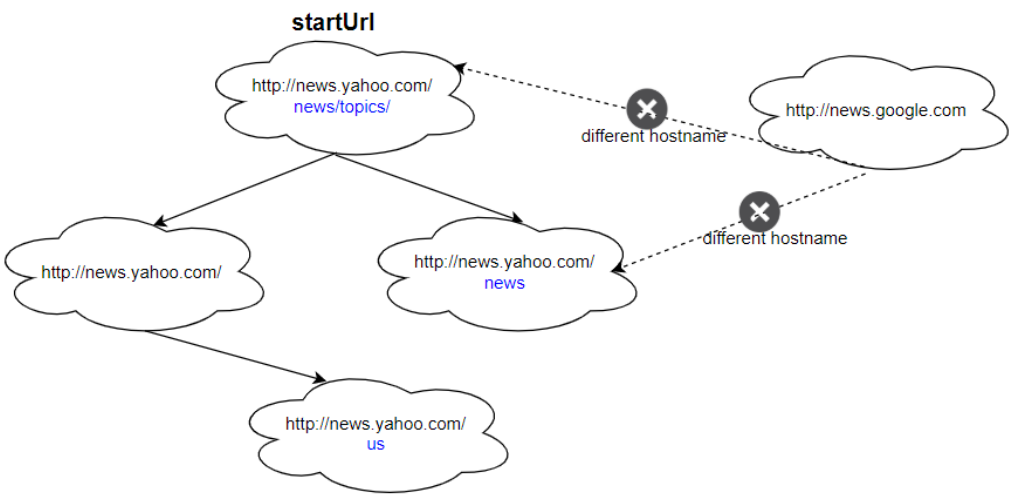


```
  public List<String> getUrls(String url);
}
```

Note that `getUrls(String url)` simulates performing an HTTP request. You can treat it as a blocking function call that waits for an HTTP r
guaranteed that `getUrls(String url)` will return the URLs within **15ms.** Single-threaded solutions will exceed the time limit so, can your r
crawler do better?

Below are two examples explaining the functionality of the problem. For custom testing purposes, you'll have three variables `urls`, `edges`
Notice that you will only have access to `startUrl` in your code, while `urls` and `edges` are not directly accessible to you in code.

Example 1:




```
Input:
urls = [
  "http://news.yahoo.com",
  "http://news.yahoo.com/news",
  "http://news.yahoo.com/news/topics/",
  "http://news.google.com",
  "http://news.yahoo.com/us"
]
edges = [[2,0],[2,1],[3,2],[3,1],[0,4]]
startUrl = "http://news.yahoo.com/news/topics/"
Output: [
  "http://news.yahoo.com",
  "http://news.yahoo.com/news",
  "http://news.yahoo.com/news/topics/",
  "http://news.yahoo.com/us"
]
```

Example 2:

Concise and Beautiful Python

auwdish
657 13269 Jul 16, 2020

Summary
We implement a classic BFS but the entries in our queue are future objects instead of primitve values. A pool of at most max_workers threads is used to execute getUrl calls asynchronously. Calling result() on our futures blocks until the task is completed or rejected.

```python
from concurrent import futures

class Solution:
    def crawl(self, startUrl: str, htmlParser: 'HtmlParser') -> List[str]:
        hostname = lambda url: url.split('/')[2]
        seen = {startUrl}

        with futures.ThreadPoolExecutor(max_workers=16) as executor:
            tasks = deque([executor.submit(htmlParser.getUrls, startUrl)])
            while tasks:
                for url in tasks.popleft().result():
                    if url not in seen and hostname(startUrl) == hostname(url):
                        seen.add(url)
                        tasks.append(executor.submit(htmlParser.getUrls, url))

        return list(seen)
```

Next
Efficient C++ with mutex and condition variable 97%

Pssssyduck Oct 09, 2020

The great thing about this solution is:

1245. Tree Diameter Premium

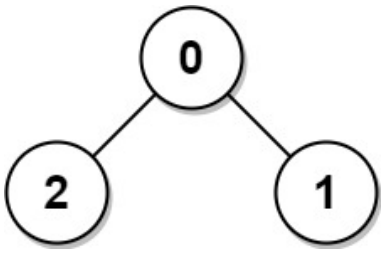
Medium Topics Companies Hint

The **diameter** of a tree is **the number of edges** in the longest path in that tree.

There is an undirected tree of `n` nodes labeled from `0` to `n - 1`. You are given a 2D array `edges` where `edges.length == n - 1` and `ed` indicates that there is an undirected edge between nodes a_i and b_i in the tree.

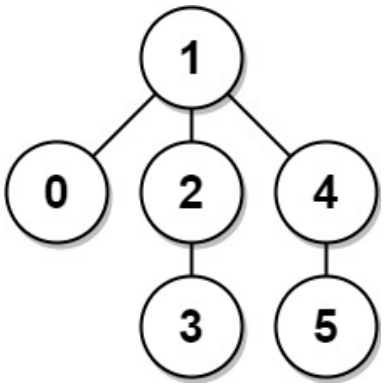
Return *the **diameter** of the tree.*

Example 1:




```
Input: edges = [[0,1],[0,2]]
Output: 2
Explanation: The longest path of the tree is the path 1 - 0 - 2.
```

Example 2:




```
Input: edges = [[0,1],[1,2],[2,3],[1,4],[4,5]]
Output: 4
Explanation: The longest path of the tree is the path 3 - 2 - 1 - 4 - 5.
```

Constraints:

- `n == edges.length + 1`
- `1 <= n <= 10^4`

Facebook Run Submit

Description Solutions Submissions Editorial

Code

Java Auto

```java
class Solution {
    int diameter = 0;
    public int treeDiameter(int[][] edges) {
        int n = edges.length + 1;
        List<Integer>[] graph = new List[n];
        for (int i = 0; i < n; ++i) graph[i] = new LinkedList<>();
        for (int[] e : edges) {
            graph[e[0]].add(e[1]);
            graph[e[1]].add(e[0]);
        }
        diameter = 0;
        depth(0, -1, graph);
        return diameter;
    }
    // Depth of the tree is the number of nodes along the longest path from the root node down to the farthest leaf nc
    int depth(int root, int parent, List<Integer>[] graph) {
        int maxDepth1st = 0, maxDepth2nd = 0;
        for (int child : graph[root]) {
            if (child == parent) continue; // Only one way from root node to child node, don't allow child node go to
            int childDepth = depth(child, root, graph);
            if (childDepth > maxDepth1st) {
                maxDepth2nd = maxDepth1st;
                maxDepth1st = childDepth;
            } else if (childDepth > maxDepth2nd) {
                maxDepth2nd = childDepth;
            }
        }
        int longestPathThroughRoot = maxDepth1st + maxDepth2nd ;
        diameter = Math.max(diameter, longestPathThroughRoot );
        return maxDepth1st + 1;
    }

}
```

Saved to cloud

Testcase Test Result

Case 1 Case 2

edges =

```
[[0,1],[0,2]]
```

1265. Print Immutable Linked List in Reverse Premium

Medium Topics Companies

You are given an immutable linked list, print out all values of each node in reverse with the help of the following interface:

- `ImmutableListNode`: An interface of immutable linked list, you are given the head of the list.

You need to use the following functions to access the linked list (you **can't** access the `ImmutableListNode` directly):

- `ImmutableListNode.printValue()`: Print value of the current node.
- `ImmutableListNode.getNext()`: Return the next node.

The input is only given to initialize the linked list internally. You must solve this problem without modifying the linked list. In other words, yc linked list using only the mentioned APIs.

Example 1:

```
Input: head = [1,2,3,4]
Output: [4,3,2,1]
```

Example 2:

```
Input: head = [0,-4,-1,3,-5]
Output: [-5,3,-1,-4,0]
```

Example 3:

```
Input: head = [-2,0,6,4,4,-6]
Output: [-6,4,4,6,0,-2]
```

Constraints:

- The length of the linked list is between `[1, 1000]`.
- The value of each node in the linked list is between `[-1000, 1000]`.

Follow up:

Could you solve this problem in:

- Constant space complexity?
- Linear time complexity and less than linear space complexity?

Iterative - Recursive

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        if (head == null) {
            return;
        }
        printLinkedListInReverse(head.getNext());
        head.printValue();
    }
}
```

Iterative - With Stack

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        Stack<ImmutableListNode> stack = new Stack<>();
        while (head != null) {
            stack.push(head);
            head = head.getNext();
        }
        while (!stack.isEmpty()) {
            stack.pop().printValue();
        }
    }
}
```

Follow Up - Constant space complexity (Time: O(n^2))

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        int numNodesCount = getNumNodesCount(head);
        for (int i = numNodesCount; i >= 1; i--) {
            printNthNode(head, i);
        }
    }

    private void printNthNode(ImmutableListNode head, int index) {
        ImmutableListNode node = head;
        for (int i = 0; i < index - 1; i++) {
            node = node.getNext();
```

```
    }
}
```

Follow Up - Constant space complexity (Time: O(n^2))

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        int numNodesCount = getNumNodesCount(head);
        for (int i = numNodesCount; i >= 1; i--) {
            printNthNode(head, i);
        }
    }

    private void printNthNode(ImmutableListNode head, int index) {
        ImmutableListNode node = head;
        for (int i = 0; i < index - 1; i++) {
            node = node.getNext();
        }
        node.printValue();
    }

    private int getNumNodesCount(ImmutableListNode head) {
        int count = 0;
        ImmutableListNode node = head;
        while (node != null) {
            count++;
            node = node.getNext();
        }
        return count;
    }
}
```

Follow Up - Linear time complexity and less than linear space complexity
Idea - Split the list to sqrt(n) equal-size small list. Print each part in reverse order.

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        // Time: O(n)
        int numNodesCount = getNumNodesCount(head);

        // Time: O(n) Space: O(sqrt(n))
```

Follow Up - Linear time complexity and less than linear space complexity
Idea - Split the list to sqrt(n) equal-size small list. Print each part in reverse order.

```
class Solution {
    public void printLinkedListInReverse(ImmutableListNode head) {
        // Time: O(n)
        int numNodesCount = getNumNodesCount(head);

        // Time: O(n) Space: O(sqrt(n))
        int step = (int)Math.sqrt(numNodesCount) + 1;
        Stack<ImmutableListNode> headNodes = new Stack<>();
        addNodeWithStep(head, step, headNodes);

        // Time: O(n) Space: O(sqrt(n))
        printEachHeadNodesInReverseOrder(headNodes);
    }

    private int getNumNodesCount(ImmutableListNode head) {
        int count = 0;
        ImmutableListNode node = head;
        while (node != null) {
            count++;
            node = node.getNext();
        }
        return count;
    }

    private void addNodeWithStep(ImmutableListNode head, int step, Stack<ImmutableListNode> he
        ImmutableListNode node = head;
        int i = 0;
        while (node != null) {
            if (i % step == 0) {
                headNodes.push(node);
            }
            node = node.getNext();
            i++;
        }
    }

    private void printEachHeadNodesInReverseOrder(Stack<ImmutableListNode> headNodes) {
        ImmutableListNode startNode = null;
        ImmutableListNode endNode = null;
```

```
        int i = 0;
        while (node != null) {
            if (i % step == 0) {
                headNodes.push(node);
            }
            node = node.getNext();
            i++;
        }
    }

    private void printEachHeadNodesInReverseOrder(Stack<ImmutableListNode> headNodes) {
        ImmutableListNode startNode = null;
        ImmutableListNode endNode = null;
        ImmutableListNode tempNode = null;

        while (!headNodes.isEmpty()) {
            endNode = startNode;
            startNode = headNodes.pop();
            tempNode = startNode;

            Stack<ImmutableListNode> stack = new Stack<>();
            while (tempNode != endNode) {
                stack.push(tempNode);
                tempNode = tempNode.getNext();
            }

            while (!stack.isEmpty()) {
                stack.pop().printValue();
            }
        }
    }
}
```

Next
[Py] 5 time space combinations, space O(n) O(√n) O(lg n) O(1)

Comments (12)

Sort by: Best

Type comment here... (Markdown supported)

1428. Leftmost Column with at Least a One Premium

Medium Topics Companies Hint

A **row-sorted binary matrix** means that all elements are `0` or `1` and each row of the matrix is sorted in non-decreasing order.

Given a **row-sorted binary matrix** `binaryMatrix`, return *the index (0-indexed) of the* ***leftmost column*** *with a 1 in it*. If such an index does

You can't access the Binary Matrix directly. You may only access the matrix using a `BinaryMatrix` interface:

- `BinaryMatrix.get(row, col)` returns the element of the matrix at index `(row, col)` (0-indexed).
- `BinaryMatrix.dimensions()` returns the dimensions of the matrix as a list of 2 elements `[rows, cols]`, which means the matrix is `row`

Submissions making more than `1000` calls to `BinaryMatrix.get` will be judged *Wrong Answer*. Also, any solutions that attempt to circumv result in disqualification.

For custom testing purposes, the input will be the entire binary matrix `mat`. You will not have access to the binary matrix directly.

Example 1:

0	0
1	1

```
Input: mat = [[0,0],[1,1]]
Output: 0
```

Example 2:

0	0
0	1

```
Input: mat = [[0,0],[0,1]]
Output: 1
```

Example 3:

0	0
0	0

```
Input: mat = [[0,0],[0,0]]
Output: -1
```

Constraints:

Facebook

Run Submit

Description Solutions Submissions Editorial

Code

Java Auto

```java
/**
 * // This is the BinaryMatrix's API interface.
 * // You should not implement it, or speculate about its implementation
 * interface BinaryMatrix {
 *     public int get(int row, int col) {}
 *     public List<Integer> dimensions {}
 * };
 */

class Solution {
    public int leftMostColumnWithOne(BinaryMatrix binaryMatrix) {
        List<Integer> dimen = binaryMatrix.dimensions();
        int m = dimen.get(0), n = dimen.get(1);
        int left = 0, right = n - 1, ans = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (existOneInColumn(binaryMatrix, m, mid)) {
                ans = mid;          // record as current ans
                right = mid - 1;    // try to find in the left side
            } else {
                left = mid + 1;     // try to find in the right side
            }
        }
        return ans;
    }
    boolean existOneInColumn(BinaryMatrix binaryMatrix, int m, int c) {
        for (int r = 0; r < m; r++) if (binaryMatrix.get(r, c) == 1) return true;
        return false;
    }

}
```

Saved to cloud

Testcase Test Result

Case 1 Case 2 Case 3

```
[[0,0],[1,1]]
```