Problem 1236: Web Crawler

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given a url

startUrl

and an interface

HtmlParser

, implement a web crawler to crawl all links that are under the

same hostname

as

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 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 any

port

specified. 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

and

startUrl

. Notice that you will only have access to

startUrl

in your code, while

urls

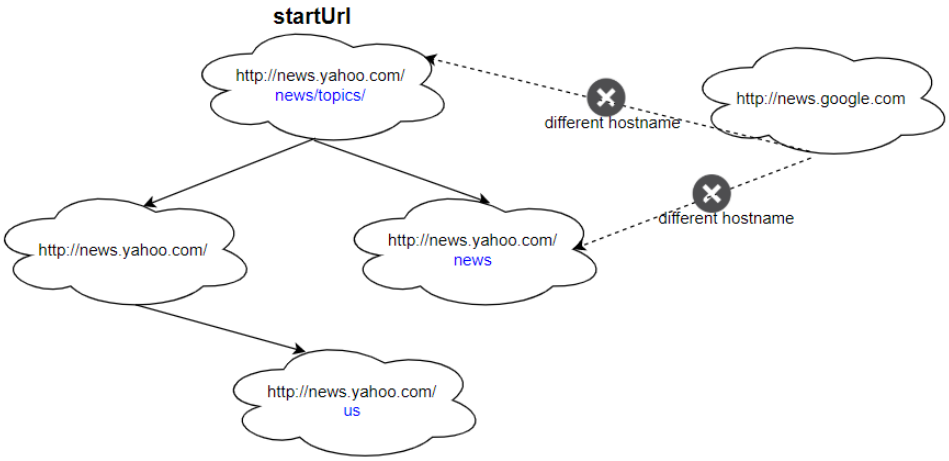
and

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.com/" are different urls.

Example 1:



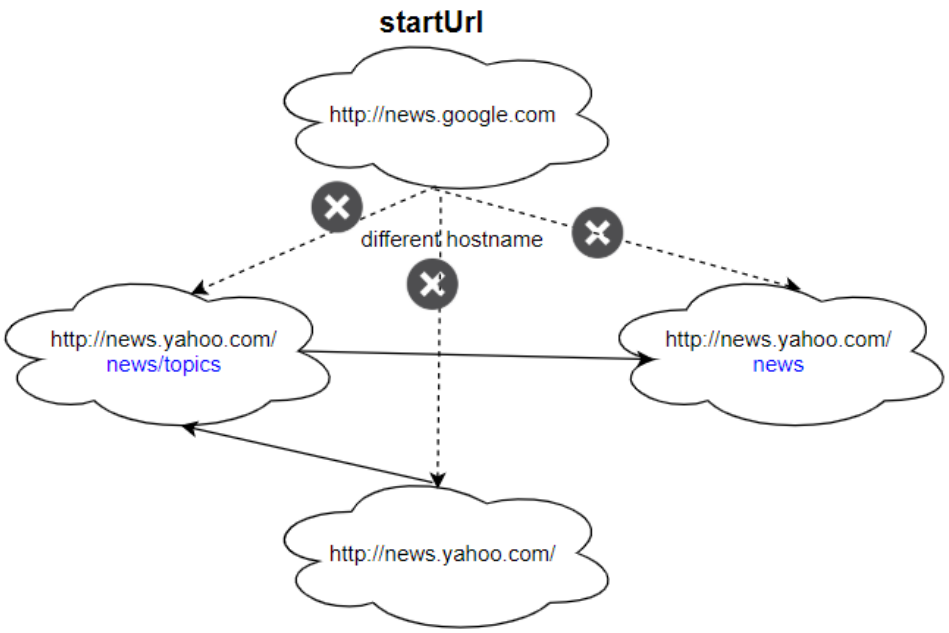
Input:

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:



Input:

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

Output:

["http://news.google.com"]

Explanation:

The startUrl links to all other pages that do not share the same hostname.

Constraints:

1 <= urls.length <= 1000

1 <= urls[i].length <= 300

startUrl

is one of the

urls

.

Hostname label must be from 1 to 63 characters long, including the dots, may contain only the ASCII letters from 'a' to 'z', digits from '0' to '9' and the hyphen-minus character ('-').

The hostname may not start or end with the hyphen-minus character ('-').

See:

https://en.wikipedia.org/wiki/Hostname#Restrictions_on_valid_hostnames

You may assume there're no duplicates in url library.

Code Snippets

C++:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* public:
* vector<string> getUrls(string url);
* };
*/
```

```
class Solution {
public:
vector<string> crawl(string startUrl, HtmlParser htmlParser) {

}
};
```

Java:

```
/**
* // 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) {

}
}
```

Python3:

```
# """
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# """
#class HtmlParser(object):
# def getUrls(self, url):
# """
# :type url: str
# :rtype List[str]
# """

class Solution:
def crawl(self, startUrl: str, htmlParser: 'HtmlParser') -> List[str]:
```

Python:

```
# """
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# """
#class HtmlParser(object):
# def getUrls(self, url):
# """
# :type url: str
# :rtype List[str]
# """

class Solution(object):
def crawl(self, startUrl, htmlParser):
"""
:type startUrl: str
:type htmlParser: HtmlParser
:rtype: List[str]
"""
```

JavaScript:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* function HtmlParser() {
*
* @param {string} url
* @return {string[]}
* this.getUrls = function(url) {
* ...
* };
* };
*/

/**
* @param {string} startUrl
* @param {HtmlParser} htmlParser
* @return {string[]}
*/
var crawl = function(startUrl, htmlParser) {

};
```

TypeScript:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* getUrls(url: string): string[] {}
* }
*/

function crawl(startUrl: string, htmlParser: HtmlParser): string[] {

};
```

C#:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* public List<String> GetUrls(String url) {}
* }
*/

class Solution {
public IList<string> Crawl(string startUrl, HtmlParser htmlParser) {

}
}
```

Go:

```
/**
* // This is HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* type HtmlParser struct {
* func GetUrls(url string) []string {}
* }
*/

func crawl(startUrl string, htmlParser HtmlParser) []string {

}
```

Kotlin:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* fun getUrls(url:String):List<String> {}
* }
*/

class Solution {
fun crawl(startUrl:String, htmlParser:HtmlParser):List<String> {

}
}
```

Swift:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* public class HtmlParser {
* public func getUrls(_ url: String) -> [String] {}
* }
*/

class Solution {
func crawl(_ startUrl: String, _ htmlParser: HtmlParser) -> [String] {

}
}
```

Ruby:

```
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# class HtmlParser
# def getUrls(url)
# @return {List[String]}
# end
# end

# @param {String} startUrl
```

```
# @param {HtmlParser} htmlParser
# @return {String}
def crawl(startUrl, htmlParser)

end
```

PHP:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class {
* public function getUrls($url) {}
* }
*/

class Solution {
/**
* @param String $startUrl
* @param HtmlParser $htmlParser
* @return String[]
*/
function crawl($startUrl, $htmlParser) {

}
}
```

Scala:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* def getUrls(url: String): List[String] = {}
* }
*/

object Solution {
def crawl(startUrl: String, htmlParser: HtmlParser): Array[String] = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Web Crawler
* Difficulty: Medium
* Tags: string, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* public:
* vector<string> getUrls(string url);
* };
*/

class Solution {
public:
vector<string> crawl(string startUrl, HtmlParser htmlParser) {

}
};
```

Java Solution:

```
/**
* Problem: Web Crawler
* Difficulty: Medium
* Tags: string, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* // 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) {

}
}
```

Python3 Solution:

```
"""
Problem: Web Crawler
Difficulty: Medium
Tags: string, search

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

# """
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# """
#class HtmlParser(object):
# def getUrls(self, url):
# """
# :type url: str
# :rtype List[str]
# """

class Solution:
def crawl(self, startUrl: str, htmlParser: 'HtmlParser') -> List[str]:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
# """
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# """
#class HtmlParser(object):
# def getUrls(self, url):
# """
# :type url: str
# :rtype List[str]
# """

class Solution(object):
def crawl(self, startUrl, htmlParser):
"""
:type startUrl: str
:type htmlParser: HtmlParser
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Web Crawler
* Difficulty: Medium
* Tags: string, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* function HtmlParser() {
*
* @param {string} url
```

```
* @return {string[]}
* this.getUrls = function(url) {
* ...
* };
* };
*/

/**
* @param {string} startUrl
* @param {HtmlParser} htmlParser
* @return {string[]}
*/
var crawl = function(startUrl, htmlParser) {

};
```

TypeScript Solution:

```
/**
* Problem: Web Crawler
* Difficulty: Medium
* Tags: string, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* getUrls(url: string): string[] {}
* }
*/

function crawl(startUrl: string, htmlParser: HtmlParser): string[] {

};
```

C# Solution:

```
/*
* Problem: Web Crawler
* Difficulty: Medium
* Tags: string, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* public List<String> GetUrls(String url) {}
* }
*/

class Solution {
public IList<string> Crawl(string startUrl, HtmlParser htmlParser) {

}
}
```

Go Solution:

```
// Problem: Web Crawler
// Difficulty: Medium
// Tags: string, search
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

/**
* // This is HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* type HtmlParser struct {
* func GetUrls(url string) []string {}
* }
*/
```

```
func crawl(startUrl string, htmlParser HtmlParser) []string {

}
```

Kotlin Solution:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* fun getUrls(url:String):List<String> {}
* }
*/

class Solution {
fun crawl(startUrl:String, htmlParser:HtmlParser):List<String> {

}
}
```

Swift Solution:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* public class HtmlParser {
* public func getUrls(_ url: String) -> [String] {}
* }
*/

class Solution {
func crawl(_ startUrl: String, _ htmlParser: HtmlParser) -> [String] {

}
}
```

Ruby Solution:

```
# This is HtmlParser's API interface.
# You should not implement it, or speculate about its implementation
# class HtmlParser
```

```
# def getUrls(url)
# @return {List[String]}
# end
# end

# @param {String} startUrl
# @param {HtmlParser} htmlParser
# @return {String}
def crawl(startUrl, htmlParser)

end
```

PHP Solution:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class {
* public function getUrls($url) {}
* }
*/

class Solution {
/**
* @param String $startUrl
* @param HtmlParser $htmlParser
* @return String[]
*/
function crawl($startUrl, $htmlParser) {

}
}
```

Scala Solution:

```
/**
* // This is the HtmlParser's API interface.
* // You should not implement it, or speculate about its implementation
* class HtmlParser {
* def getUrls(url: String): List[String] = {}
* }
```

```
*/

object Solution {
def crawl(startUrl: String, htmlParser: HtmlParser): Array[String] = {

}
}
```