

STUDENT PORTFOLIO



Name: ANAS AHMED ATHER
Register Number: RA2011031010006
Mail ID: aa0094@srmist.edu.in
Department: NWC CSE
Specialization: Information technology
Semester: 5th

Subject Title: 18CSC301J Formal Language & Automata

Handled By: Dr.P.Balaji Srikanth (102776)

Regex Module of Hackerrank

SUBDOMAINS



Introduction

Q.1.Matching Specific String

HackerRank Prepare > Regex > Introduction Matching Specific String Exit Full Screen View

Problem
Regular expression (or RegEx)
A regular expression is a sequence of characters that define a search pattern. It is mainly used for string pattern matching.

The diagram illustrates the matching process. A box labeled 'Regex Pattern' contains the text 'wikipedia'. A blue arrow points down to a box labeled 'Test String' which contains the URL 'https://en.wikipedia.org/'.

In the above image, a Regex Pattern is matched with the Test String

Regular expressions are extremely useful in extracting information from text such as: code, log files, spreadsheets, documents, etc.

We can match a specific string in a test string by making our regex pattern.

```
1 Regex_Pattern = r'hackerrank' # Do not delete 'r'.
2 import re
3
4 Test_String = input()
5
6 match = re.findall(Regex_Pattern, Test_String)
7
8 print("Number of matches :", len(match))
```

Upload Code as File Test against custom input Run Code Submit Code

HackerRank Prepare > Regex > Introduction Matching Specific String Exit Full Screen View

Problem
Regular expression (or RegEx)
A regular expression is a sequence of characters that define a search pattern. It is mainly used for string pattern matching.

The diagram illustrates the matching process. A box labeled 'Regex Pattern' contains the text 'wikipedia'. A blue arrow points down to a box labeled 'Test String' which contains the URL 'https://en.wikipedia.org/'.

In the above image, a Regex Pattern is matched with the Test String

Regular expressions are extremely useful in extracting information from text such as: code, log files, spreadsheets, documents, etc.

We can match a specific string in a test string by making our regex pattern.

Congratulations
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Compiler Message
Success

Input (stdin)
The hackerrank team is on a mission to flatten the world by restructuring the DNA of every company on the planet. We rank programmers based on their coding skills, helping companies source great programmers and reduce the time to hire. As a result, we are revolutionizing the way companies discover and evaluate talented engineers. The hackerrank platform is the destination for the best engineers to hone their skills and companies to find top engineers.

Expected Output
Number of matches : 2

Upload Code as File Test against custom input Run Code Submit Code

Q.2.Matching digit and non digit character

Problem

Submissions

Leaderboard

Discussions

`\d`

The expression `\d` matches any digit [0 - 9].

Regex Pattern

`\d\d\d`

Test String

Hack101

In the above image, a Regex Pattern is matched with the Test String

`\D`

The expression `\D` matches any character that is not a digit.

Change Theme

Language Python 3

Exit Full Screen View

```
1 Regex_Pattern = r"^\d{2}.\d{2}.\d{4}" # Do not delete 'r'.
2
3
4 import re
5
6 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 2 Col: 1

Problem

Submissions

Leaderboard

Discussions

`\d`

The expression `\d` matches any digit [0 - 9].

Regex Pattern

`\d\d\d`

Test String

Hack101

In the above image, a Regex Pattern is matched with the Test String

`\D`

The expression `\D` matches any character that is not a digit.

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

06-11-2015

Download

Test case 2

Expected Output

true

Download

Test case 3

Test case 4

Test case 5

Q.3. Matching anything but a newline

Problem

dot

The dot (.) matches anything (except for a newline).

Submissions

Leaderboard

Discussions

Regex Pattern

A.B.C.D.

Test String

A+B-C=DE

In the above image, a *Regex Pattern* is matched with the *Test String*

Note: If you want to match (.) in the test string, you need to escape the dot by using a slash \. .
In Java, use \\., instead of \. .

Change Theme

Language Python 3

```
1 regex_pattern = r"^(...\\.){3}..." # Do not delete 'r'.
2
3 import re
4 import sys
5
6 test_string = input()
7
8 match = re.match(regex_pattern, test_string) is not None
9
10 print(str(match).lower())
```

Line: 1 Col: 34

Problem

dot

The dot (.) matches anything (except for a newline).

Submissions

Leaderboard

Discussions

Regex Pattern

A.B.C.D.

Test String

A+B-C=DE

In the above image, a *Regex Pattern* is matched with the *Test String*

Note: If you want to match (.) in the test string, you need to escape the dot by using a slash \. .
In Java, use \\., instead of \. .

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

123.456.abc.def

Expected Output

true

Q.4. Matching Word and non word character

HackerRank

Prepare > Regex > Introduction > Matching Word & Non-Word Character

Exit Full Screen View

Problem

`\w`

The expression `\w` will match any word character.
Word characters include alphanumeric characters (`a-z`, `A-Z` and `0-9`) and underscores (`_`).

Submissions

Leaderboard

Discussions

Regex Pattern

`\w\w\w`

Test String

`Sone`

In the above image, Regex Pattern is matched with the Test String

```
1 Regex_Pattern = r"\w{3}\W\w{10}\W\w{3}" # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 39

HackerRank

Prepare > Regex > Introduction > Matching Word & Non-Word Character

Exit Full Screen View

Problem

`\w`

The expression `\w` will match any word character.
Word characters include alphanumeric characters (`a-z`, `A-Z` and `0-9`) and underscores (`_`).

Submissions

Leaderboard

Discussions

Regex Pattern

`\w\w\w`

Test String

`Sone`

In the above image, Regex Pattern is matched with the Test String

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin)

1 `www.hackerrank.com` [Download](#)

Expected Output

1 `true` [Download](#)

Regex

Points: 310 Rank: 17697

★

Matching Specific String

Easy, Max Score: 5, Success Rate: 96.23%

Solved ✓

★

Matching Anything But a Newline

Easy, Max Score: 5, Success Rate: 83.66%

Solved ✓

★

Matching Digits & Non-Digit Characters

Easy, Max Score: 5, Success Rate: 97.33%

Solved ✓

★

Matching Word & Non-Word Character

Easy, Max Score: 5, Success Rate: 98.74%

Solved ✓

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☒ Introduction

☐ Character Class

☐ Repetitions

☐ Grouping and Capturing

☐ Backreferences

☐ Assertions



Character Class

Q.5. Matching Specific character

Problem

`[]`

The character class `[]` matches only one out of several characters placed inside the square brackets.

Regex Pattern

`[aeiou] is a vowel`

Test String

`o is a vowel`
`e is a vowel`

In the above image, the Regex Pattern is matched with the Test String

Task

Change Theme

Language

Python 3

```
1 Regex_Pattern = r'^[1-3][0-2][xs0][30Aa][xsu][.,]$' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 51

Problem

`[]`

The character class `[]` matches only one out of several characters placed inside the square brackets.

Regex Pattern

`[aeiou] is a vowel`

Test String

`o is a vowel`
`e is a vowel`

In the above image, the Regex Pattern is matched with the Test String

Task

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

1 1203x.

Test case 2

Expected Output

Download

1 true

Test case 3

Test case 4

Test case 5

Test case 6

Q.6.Excluding specific character

Problem

[^]

The negated character class [^] matches any character that is not in the square brackets.

Submissions

Leaderboard

Discussions

Regex Pattern

[^aeiou] is not a vowel

↓

Test String

k is not a vowel
p is not a vowel

In the above image, the Regex Pattern is matched with the Test String

Task

Change Theme

Language Python 3

Exit Full Screen View

```
1 Regex_Pattern = r'^[D[^aeiou][^bcDF]\S[^AEIOU][^.,]$\'' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 53

Problem

[^]

The negated character class [^] matches any character that is not in the square brackets.

Submissions

Leaderboard

Discussions

Regex Pattern

[^aeiou] is not a vowel

↓

Test String

k is not a vowel
p is not a vowel

In the above image, the Regex Pattern is matched with the Test String

Task

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

1 think?

Test case 2

Expected Output

Download

1 true

Test case 3

Test case 4

Test case 5

Test case 6

Q.7. Matching character ranges

Problem

Submissions

Leaderboard

Discussions

In the context of a regular expression (RegEx), a character class is a set of characters enclosed within square brackets that allows you to match one character in the set.

A hyphen (-) inside a character class specifies a range of characters where the left and right operands are the respective lower and upper bounds of the range. For example:

- [a-z] is the same as [abcdefghijklmnopqrstuvwxyz].
- [A-Z] is the same as [ABCDEFGHIJKLMNOPQRSTUVWXYZ].
- [0-9] is the same as [0123456789].

In addition, if you use a caret (^) as the first character inside a character class, it will match anything that is not in that range. For example, [^0-9] matches any character that is not a digit in the inclusive range from 0 to 9. It's important to note that, when used outside of (immediately preceding) a character or character class, the caret matches the first character in the string against that character or set of characters.

Regex Pattern

[x-z][4-8][A-K]

Change Theme

Language Python 3

1 Regex_Pattern = r'^[a-z][1-9][^a-z][^A-Z][A-Z]'

2 # Do not delete 'r'.

3 import re

4

5 print(str(bool(re.search(Regex_Pattern, input()))).lower())

Line: 1 Col: 48

HackerRank

Prepare > Regex > Character Class > Matching Character Ranges

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

In the context of a regular expression (RegEx), a character class is a set of characters enclosed within square brackets that allows you to match one character in the set.

A hyphen (-) inside a character class specifies a range of characters where the left and right operands are the respective lower and upper bounds of the range. For example:

- [a-z] is the same as [abcdefghijklmnopqrstuvwxyz].
- [A-Z] is the same as [ABCDEFGHIJKLMNOPQRSTUVWXYZ].
- [0-9] is the same as [0123456789].

In addition, if you use a caret (^) as the first character inside a character class, it will match anything that is not in that range. For example, [^0-9] matches any character that is not a digit in the inclusive range from 0 to 9. It's important to note that, when used outside of (immediately preceding) a character or character class, the caret matches the first character in the string against that character or set of characters.

Regex Pattern

[x-z][4-8][A-K]

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

h4ckR

Expected Output

true

HackerRank

Prepare > Regex > Character Class > Matching Character Ranges

Exit Full Screen View

HackerRank

PREPARE

CERTIFY

COMPETE

Search

aa0094

Prepare > Regex

Points: 310 Rank: 17697

Matching Specific Characters

★

Solved

Easy, Max Score: 10, Success Rate: 96.34%

Excluding Specific Characters

★

Solved

Easy, Max Score: 10, Success Rate: 97.86%

Matching Character Ranges

★

Solved

Easy, Max Score: 10, Success Rate: 96.22%

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Introduction

☒ Character Class

☐ Repetitions

☐ Grouping and Capturing

☐ Backreferences

☐ Assertions

☐ Applications

Q.9. Matching {x,y} Repetitions

Problem

{x,y}

The **{x,y}** tool will match between **x** and **y** (both inclusive) repetitions of character/character class/group.

Submissions

Leaderboard

Discussions

Regex Pattern

`\w{1,4}\d{4,}`

↓

Test String

`Hk132156153186131`
`Hack1021`

In the above image, the Regex Pattern is matched with the Test String

For Example:

Change Theme

Language

Python 3

Exit Full Screen View

```
1 Regex_Pattern = r'^\d{1,2}[a-zA-Z]{3,}[.]{0,3}$' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 48

Problem

{x,y}

The **{x,y}** tool will match between **x** and **y** (both inclusive) repetitions of character/character class/group.

Submissions

Leaderboard

Discussions

Regex Pattern

`\w{1,4}\d{4,}`

↓

Test String

`Hk132156153186131`
`Hack1021`

In the above image, the Regex Pattern is matched with the Test String

For Example:

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Download

Input (stdin)

`3threeorealphabets.`

Download

Expected Output

`true`

Download

Q.10. Matching zero and more repetitions

Problem

Submissions

Leaderboard

Discussions

The * tool will match zero or more repetitions of character/character class/group.

Regex Pattern

Ab*s

Test String

As
Abbbbbs

In the above image, the Regex Pattern is matched with the Test String

For Example:

w*: It will match the character w 0 or more times.

Change Theme

Language Python 3

```
1 Regex_Pattern = r'^\d{2,}[a-z]*[A-Z]+' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 39

Problem

Submissions

Leaderboard

Discussions

The * tool will match zero or more repetitions of character/character class/group.

Regex Pattern

Ab*s

Test String

As
Abbbbbs

In the above image, the Regex Pattern is matched with the Test String

For Example:

w*: It will match the character w 0 or more times.

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

14

Download

Test case 2

Expected Output

true

Download

Test case 3

Download

Test case 4

Download

Test case 5

Download

Test case 6

Download

HackerRank

PREPARE

CERTIFY

COMPETE

Search

aa0094

Prepare > Regex

Points: 310 Rank: 17697

Matching {x} Repetitions

Easy, Max Score: 20, Success Rate: 95.54%

Solved

Matching {x, y} Repetitions

Easy, Max Score: 20, Success Rate: 97.79%

Solved

Matching Zero Or More Repetitions

Easy, Max Score: 20, Success Rate: 98.93%

Solved

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Introduction

☐ Character Class

☒ Repetitions



Grouping and Capturing

Q.11. Matching word boundaries

HackerRank Prepare > Regex > Grouping and Capturing > Matching Word Boundaries Exit Full Screen View

Problem
`\b` assert position at a word boundary.
Three different positions qualify for word boundaries :

- ▶ Before the first character in the string, if the first character is a word character.
- ▶ Between two characters in the string, where one is a word character and the other is not a word character.
- ▶ After the last character in the string, if the last character is a word character.

Submissions
Leaderboard
Discussions

Regex Pattern
`\bcat\b`

Test String
Acat
A cat

```
1 Regex_Pattern = r'\b[aeiouAEIOU][a-zA-Z]+\b' # Do not delete 'r'.
2
3 > import re...
```

Line: 1 Col: 44

HackerRank Prepare > Regex > Grouping and Capturing > Matching Word Boundaries Exit Full Screen View

Problem
`\b` assert position at a word boundary.
Three different positions qualify for word boundaries :

- ▶ Before the first character in the string, if the first character is a word character.
- ▶ Between two characters in the string, where one is a word character and the other is not a word character.
- ▶ After the last character in the string, if the last character is a word character.

Submissions
Leaderboard
Discussions

Regex Pattern
`\bcat\b`

Test String
Acat
A cat

Upload Code as File Test against custom input Run Code Submit Code

Congratulations
You solved this challenge. Would you like to challenge your friends? [Facebook](#) [Twitter](#) [LinkedIn](#) [Next Challenge](#)

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6

Compiler Message
Success

Input (stdin)
1 Found any match?

Expected Output
1 true

Download Download

Type here to search Desktop 12:14 PM 13-11-2022

Q.12.Capturing and non-capturing group

HackerRank

Prepare > Regex > Grouping and Capturing > Capturing & Non-Capturing Groups

Exit Full Screen View

Problem

()

Parenthesis () around a regular expression can group that part of regex together. This allows us to apply different quantifiers to that group.

These parenthesis also create a numbered capturing. It stores the part of string matched by the part of regex inside parentheses.

These numbered capturing can be used for backreferences. (We shall learn about it later)

Submissions

Leaderboard

Discussions

Regex Pattern

It is (not)? your fault

↓

Test String

It is not your fault
It is your fault

Change Theme

Language Python 3

⌵ ⌵ ⌵

```
1 Regex_Pattern = r'(ok){3,}' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 1 Col: 27

HackerRank

Prepare > Regex > Grouping and Capturing > Capturing & Non-Capturing Groups

Exit Full Screen View

Problem

()

Parenthesis () around a regular expression can group that part of regex together. This allows us to apply different quantifiers to that group.

These parenthesis also create a numbered capturing. It stores the part of string matched by the part of regex inside parentheses.

These numbered capturing can be used for backreferences. (We shall learn about it later)

Submissions

Leaderboard

Discussions

Regex Pattern

It is (not)? your fault

↓

Test String

It is not your fault
It is your fault

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin)

Download

okokok! cya

Expected Output

Download

true

Q.13. Alternative Matching

HackerRank Prepare > Regex > Grouping and Capturing > Alternative Matching

Exit Full Screen View

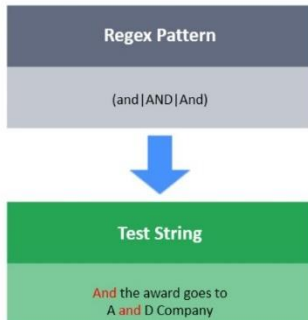
Problem

Alternations, denoted by the `|` character, match a single item out of several possible items separated by the vertical bar. When used inside a character class, it will match characters; when used inside a group, it will match entire expressions (i.e., everything to the left or everything to the right of the vertical bar). We must use parentheses to limit the use of alternations.

Submissions

Leaderboard

Discussions



In the image above, the `Regex` pattern is matched with the test string.

```
1 Regex_Pattern = r'^(Mr?s|[MDE]r)\.[a-zA-Z]+$' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Line: 3 Col: 10

HackerRank Prepare > Regex > Grouping and Capturing > Alternative Matching

Exit Full Screen View

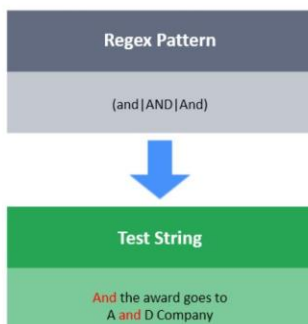
Problem

Alternations, denoted by the `|` character, match a single item out of several possible items separated by the vertical bar. When used inside a character class, it will match characters; when used inside a group, it will match entire expressions (i.e., everything to the left or everything to the right of the vertical bar). We must use parentheses to limit the use of alternations.

Submissions

Leaderboard

Discussions



In the image above, the `Regex` pattern is matched with the test string.

Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6

Compiler Message

Success

Input (stdin)

Download

Mr.DOSHI

Expected Output

Download

true

Type here to search

Desktop 12:17 PM 13-11-2022

HackerRank

PREPARE

CERTIFY

COMPETE

Search

aa0094

Prepare > Regex

Points: 310 Rank: 17697

Matching Word Boundaries

★

Solved

Easy, Max Score: 20, Success Rate: 96.90%

Capturing & Non-Capturing Groups

★

Solved

Easy, Max Score: 20, Success Rate: 98.97%

Alternative Matching

★

Solved

Easy, Max Score: 20, Success Rate: 94.04%

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Introduction

☐ Character Class

☐ Repetitions

☒ Grouping and Capturing

☒☒☒

Q.15.Branch reset group

Problem

Submissions

Leaderboard

Discussions

NOTE - Branch reset group is supported by Perl, PHP, Delphi and R.

(?)regex

A branch reset group consists of alternations and capturing groups. `(?(regex1)(regex2))`

Alternatives in branch reset group share same capturing group.

Regex Pattern

`(?|(Haa)|(Hee)|(bye)|(k))\1`

↓

Test String

HaaHaa
kk

In the above image, Regex Pattern is matched with the Test String.

Change Theme

Language

PHP

Exit Full Screen View

```
1 <?php
2
3 $Regex_Pattern = '/^\d{2}((-|:|---|.)?)\d{2}\1\d{2}\1\d{2}$/' ; //Do not delete '/'
4 ' , Replace _____ with your regex.
5 > $handle = fopen ("php://stdin","r");...
```

Line: 3 Col: 61

Problem

Submissions

Leaderboard

Discussions

NOTE - Branch reset group is supported by Perl, PHP, Delphi and R.

(?)regex

A branch reset group consists of alternations and capturing groups. `(?(regex1)(regex2))`

Alternatives in branch reset group share same capturing group.

Regex Pattern

`(?|(Haa)|(Hee)|(bye)|(k))\1`

↓

Test String

HaaHaa
kk

In the above image, Regex Pattern is matched with the Test String.

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

[f](#) [t](#) [in](#)

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

1 12-34-56-78

Expected Output

1 true

HackerRank Prepare > Regex > Backreferences > Forward References Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

HackerRank

Prepare · Regex · Backreferences · Forward References

NOTE - Forward reference is supported by 3dssoft, .NET, Java, Perl, PCRE, PHP, Delphi and Ruby regex flavors.

Forward reference creates a back reference to a regex that would appear later. Forward references are only useful if they're inside a repeated group.

Then there may arise a case in which the regex engine evaluates the backreference after the group has been matched already.

Regex Pattern

(\2amigo|go!)+

Test String

go!go!amigo

Type here to search

HackerRank

Prepare · Regex · Backreferences · Forward References

Change Theme

Language

PHP

```
1 <?php
2
3 $Regex_Pattern = '/^(\2tic|tac)+$/'; //Do not delete '/'. Replace _____
  with your regex.
4
5 > $handle = fopen ("php://stdin","r");...
```

Line: 3 Col: 37

Type here to search

HackerRank

Prepare · Regex · Backreferences · Forward References

Problem

Submissions

Leaderboard

Discussions

HackerRank

Prepare · Regex · Backreferences · Forward References

NOTE - Forward reference is supported by 3dssoft, .NET, Java, Perl, PCRE, PHP, Delphi and Ruby regex flavors.

Forward reference creates a back reference to a regex that would appear later. Forward references are only useful if they're inside a repeated group.

Then there may arise a case in which the regex engine evaluates the backreference after the group has been matched already.

Regex Pattern

(\2amigo|go!)+

Test String

go!go!amigo

Type here to search

HackerRank

Prepare · Regex · Backreferences · Forward References

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

1 tactactic

Test case 2

Expected Output

Download

1 true

Test case 3

Test case 4

Test case 5

Type here to search

HackerRank

Prepare · Regex · Backreferences · Forward References

HackerRank

PREPARE

CERTIFY

COMPETE

Search

aa0094

Prepare > Regex

Points: 310 Rank: 17697

Matching Same Text Again & Again

★

Solved

Easy, Max Score: 20, Success Rate: 97.98%

Branch Reset Groups

★

Solved

Easy, Max Score: 20, Success Rate: 95.31%

Forward References

★

Solved

Easy, Max Score: 20, Success Rate: 89.19%

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Introduction

☐ Character Class

☐ Repetitions

☐ Grouping and Capturing

☒ Backreferences



Assertions

Q.17. Positive Lookahead

Problem

regex_1(?=regex_2)

The positive lookahead (?=) asserts regex_1 to be immediately followed by regex_2. The lookahead is excluded from the match. It does not return matches of regex_2. The lookahead only asserts whether a match is possible or not.

Submissions

Leaderboard

Discussions

Regex Pattern

c(?=o)

Test String

chocolate

In the above image, the Regex Pattern is matched with the Test String

Change Theme

Language

Python 2

Exit Full Screen View

```
1 Regex_Pattern = r'o(?=oo)' # Do not delete 'r'.
2
3 > import re...
```

Line: 1 Col: 26

Type here to search

Desktop

ENG

12:23 PM

13-11-2022

HackerRank

Prepare

Regex

Assertions

Positive Lookahead

Exit Full Screen View

Problem

regex_1(?=regex_2)

The positive lookahead (?=) asserts regex_1 to be immediately followed by regex_2. The lookahead is excluded from the match. It does not return matches of regex_2. The lookahead only asserts whether a match is possible or not.

Submissions

Leaderboard

Discussions

Regex Pattern

c(?=o)

Test String

chocolate

In the above image, the Regex Pattern is matched with the Test String

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

Success

Input (stdin)

Download

Expected Output

Download

1

gooooo!

1

Number of matches : 3

Q.18. Positive Lookbehind

Problem

Submissions

Leaderboard

Discussions

(?<=regex_2)regex_1

The positive lookbehind (?<=) asserts regex_1 to be immediately preceded by regex_2. Lookbehind is excluded from the match (do not consume matches of regex_2), but only assert whether a match is possible or not.

Regex Pattern

(?<=[a-z])[aeiou]

Test String

he1o

In above image Regex Pattern is matched with the Test String.

Change Theme

Language Python 2

Exit Full Screen View

```
1 Regex_Pattern = r"(?<=[13579])\d" # Do not delete 'r'.
2
3 import re
4
5 Test_String = raw_input()
6
7 match = re.findall(Regex_Pattern, Test_String)
8
9 print "Number of matches :", len(match)
```

Line: 1 Col: 33

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

1 123Go!

Test case 2

Expected Output

Download

1 Number of matches : 1

Discussions

123Go!

Number of matches : 1

Desktop

12:24 PM

13-11-2022

Desktop

12:25 PM

13-11-2022

Q.19.Negative Lookahead

Problem

regex_1(?!regex_2)

The negative lookahead (?!) asserts regex_1 not to be immediately followed by regex_2. Lookahead is excluded from the match (do not consume matches of regex_2), but only assert whether a match is possible or not.

Submissions

Leaderboard

Discussions

Regex Pattern

c(?!o)

Test String

chocolate

In above image Regex Pattern is matched with the Test String.

Change Theme

Language Python 2

```
1 Regex_Pattern = r"(.)(?!1)" # Do not delete 'r'.
2
3 import re
4
5 Test_String = raw_input()
6
7 match = re.findall(Regex_Pattern, Test_String)
8
9 print "Number of matches :", len(match)
```

Line: 1 Col: 28

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

Download

1 gooooo

Test case 2

Expected Output

Download

1 Number of matches : 2

Type here to search

Desktop

12:26 PM

13-11-2022

Q.20. Negative Lookbehind

Problem

(?<!regex_2)regex_1

The negative lookbehind (?<!) asserts regex_1 not to be immediately preceded by regex_2. Lookbehind is excluded from the match (do not consume matches of regex_2), but only assert whether a match is possible or not.

Submissions

Leaderboard

Discussions

Regex Pattern

(?<![a-z])[aeiou]

↓

Test String

he1o

In above image Regex Pattern is matched with the Test String.

Change Theme

Language Python 2

1 Regex_Pattern = r"(?<![aeiouAEIOU])." # Do not delete 'r'.

2

3 import re

4

5 Test_String = raw_input()

6

7 match = re.findall(Regex_Pattern, Test_String)

8

9 print "Number of matches :", len(match)

Line: 1 Col: 37

Problem

(?<!regex_2)regex_1

The negative lookbehind (?<!) asserts regex_1 not to be immediately preceded by regex_2. Lookbehind is excluded from the match (do not consume matches of regex_2), but only assert whether a match is possible or not.

Submissions

Leaderboard

Discussions

Regex Pattern

(?<![a-z])[aeiou]

↓

Test String

he1o

In above image Regex Pattern is matched with the Test String.

Upload Code as File

Test against custom input

Run Code

Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

Success

Input (stdin)

1 1o1s

Download

Expected Output

1 Number of matches : 3

Download

Positive Lookahead	★	Solved
Easy, Max Score: 20, Success Rate: 99.50%		
Negative Lookahead	★	Solved
Easy, Max Score: 20, Success Rate: 97.12%		
Positive Lookbehind	★	Solved
Easy, Max Score: 20, Success Rate: 98.47%		
Negative Lookbehind	★	Solved
Easy, Max Score: 20, Success Rate: 97.83%		

STATUS

☒ Solved

☐ Unsolved

DIFFICULTY

☐ Easy

☐ Medium

☐ Hard

SUBDOMAINS

☐ Introduction

☐ Character Class

☐ Repetitions

☐ Grouping and Capturing

☐ Backreferences

☒ Assertions

Chapterwise Worksheet

Unit 1

UNIT-1.
Worksheet 1

RA2011031010006
Anas Ahmed Ather
01 CSE-IT.

M.C.Q.

(1) (b) $P(k) = m(k) + 5$

(2) (c) Trivial proof

3] ~~None of the above~~ (d) $m^3 + 3m$

4.] (a)

Descriptive Question:

1.] $f(n) = (2^n - 1) \times 3 = 0$

$f(1) \rightarrow (2^1 - 1) \times 3 = 3 \times 3 = 0$

$\therefore f(1)$ is true

Assume, $P(k)$ is true for some k , then

$P(k) = 2^k - 1 = 3a$

$\rightarrow P(k+1) \cdot 2^{2k+2} - 1 = 2^k \cdot 4 - 1 = 2^{2k} \cdot 3 + (2^{2k} - 1)$
 $= 2^{2k} \cdot 3 + 3a$

and $3(2^{2k} + a)$ is divisible by 3

\therefore by mathematical induction, statement is proved.

2.] let $a^2 = 3k$ — (1)

and $a = 3k+1$ — (2)

$a = 3k+2$ — (3)

on squaring (2) $\rightarrow a^2 = 9k^2 + 1 + 3k$ — (4)

on squaring (3) $\rightarrow a^2 = 9k^2 + 4 + 6k$ — (5)

Since (4) and (5) are not divisible by 3, it is proved by contradiction.

3.) a is odd = $2n+1$

b is even = $2n$

$a+b = 4n+1$ which is always odd.
(proved)

4.) $P = 2n^2 + 31 - 16n$

by counter example

$n = 5$

$$P \Rightarrow 2(5)^2 + 31 - 16(5)$$

$$\Rightarrow P = 5 \times 32 - 80 = 1 \text{ (positive)}$$

$n = 4$

$$P \Rightarrow 2(4)^2 + 31 - 16(4)$$

$$\Rightarrow P = 32 + 31 - 64 = -1 \text{ (negative)}$$

$\therefore P$ is not always true

(5) $n \geq 5, 2n < n^2$

$$n \geq 5 \rightarrow 2n < n^2$$

$$n = 5 \rightarrow 10 < 25$$

$$\text{for } n = k \rightarrow 2k < k^2$$

$$\text{for } n = k+1 \rightarrow 2k+2 < (k+1)^2$$

$$= 2k+2 < k^2 + 1 + 2k$$

$$= 2k < k^2 + 2k - 1$$

$$\text{and } k^2 + 2k - 1 > k^2, \text{ for } k \geq 5$$

\therefore from induction it's proved.

$$6.) \quad p(n) = 1^2 + 2^2 + 3^2$$

$$n^2 = \frac{n(n+1)(2n+1)}{6}$$

for $n=1$

$$p(n) = \frac{1 \times 2 \times 3}{6} = 1 \quad (\text{true})$$

$$\text{for } n=k \rightarrow p(n) = \frac{1 \times (k+1)(2k+1)}{6}$$

for $n=k+1 \rightarrow$

$$p(k) = 1^2 + 2^2 + 3^2 + \dots + k^2 + (k+1)^2$$

$$= \frac{k(k+1)(2k+1)}{6} + (k+1)^2$$

$$= \frac{(k+1)[k(2k+1) + (6k+6)]}{6}$$

$$= \frac{(k+1)(k+2)(2k+3)}{6}$$

\therefore It's true for $p(k+1)$

Unit 2

CT1-Worksheet-2

RA2011031010006
Ancy Ahmed Ather

Part-A.

1.) (i) Set of all strings starting and ending with 1's in binary '10'.

2.) (ii) $n = (0+1)^+ 1001(0+1)^+$

3.) (c) (i) & (iii)

4.) (i) Regular Languages.

5.) (c) The set of all strings containing at least two 0's.

Part-B.

1.) The language generated by RE $0^*(101)^+11$ is the string beginning with 0 & ending with 11 and containing 101...1 = {010111}

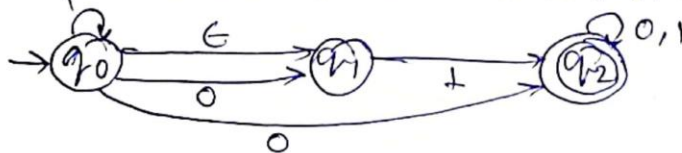
2.) RE is $(01)^+(0+1)^+ 01$

3.) Yes, it is possible to create such a scenario take any substring of the alphabet set and use $(\text{string})^+$ which signifies a repetition of for more than 1 times.

Eg- $(a+b)^+(ab)^+(a+b)$

4.) ϵ closure of state q is all the following transition out of q that are labelled

ϵ closure of $q_0 \rightarrow \{q_0, q_1, q_2\}$



ϵ closure of q_0

$\Rightarrow \epsilon^+ \cup \epsilon^*$

$q_0 \rightarrow q_1 \rightarrow \emptyset \rightarrow \emptyset$

$\epsilon^+ \cup \epsilon^*$

$q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$

$\therefore \epsilon^+ \text{ of } q_0 \Rightarrow \{q_0, q_1\}$

8.] 5 tuple structure of NFA & DFA

DFA $\rightarrow \{Q, \Sigma, q_0, F, \delta\}$

$Q \rightarrow$ set of all set

$\Sigma \rightarrow$ Input symbols

$q_0 \rightarrow$ Initial state

$F \rightarrow$ final state

$\delta \rightarrow$ Transition $F \rightarrow Q \times \Sigma = Q$

NFA $\rightarrow \{Q, \Sigma, q_0, F, \delta\}$

$Q \rightarrow$ Set of all states

$\Sigma \rightarrow$ Input symbols.

$q_0 \rightarrow$ Initial state

$F \rightarrow$ final state

$\delta \rightarrow$ Transition $F \rightarrow Q \times \Sigma = 2^Q$

Unit 3

1.) (i) Set of all strings starting and ending ^{OL: CSE-IT} of 1's in b/w '10'.

2.) (ii) $n = (0+1)^+ 1001(0+1)^+$

3.) (c) (i) & (iii)

4.) (i) Regular Languages.

5.) (c) The set of all strings containing at least 1000's.

Part-B.

1.) The language generated by RE $0^*(101)^+11$ is the string beginning with 0 & ending with 11 and containing 101...1 = {010111}

2.) RE is $(01)^+(0+1)^+ 01$

3.) Yes, it is possible to create such a scenario take any substring of the alphabet set and use $(\text{string})^+$ which signifies a repetition of for more than 1 times.

Eg- $(a+b)^+(ab)^+(a+b)$

4.) ϵ closure of state q is all the following transition out of q that are labelled

ϵ closure of $q_0 \rightarrow \{q_0, q_1, q_2\}$



Unit 4

Unit-4 (Worksheet)-1

Anas Ahmed
Atuer.

1.) True

2.) True

3.) (b) 7

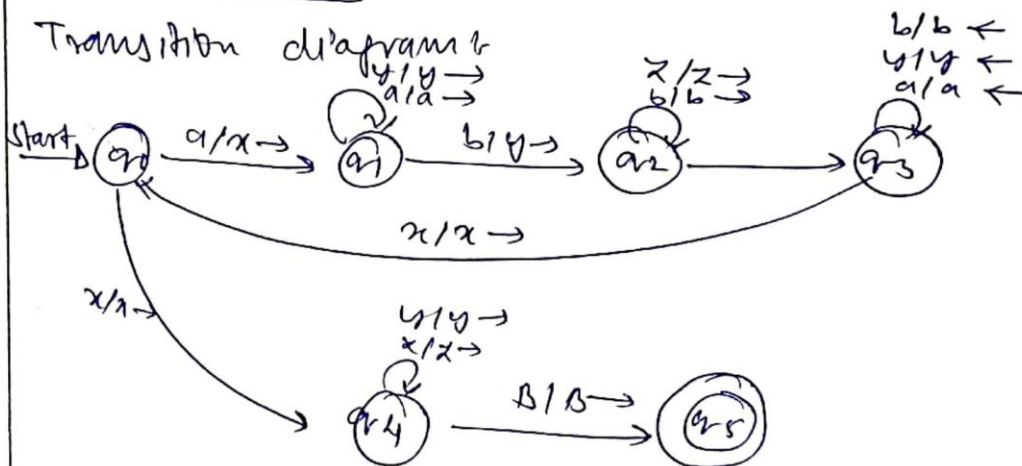
4.) A. Left

5.) A. Both S_1 and S_2 are false.

6.)

Scenario based

1.) Transition diagram



3.) TM M is

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, x, y, B\}, \delta, q_0, B, \{q_5\})$$

where δ is given.

State	0	1	x	y	B
q_0	(q_1, x, R)	-	-	(q_3, y, R)	-
q_1	$(q_1, 0, R)$	(q_2, y, L)	-	(q_1, y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, x, R)	(q_2, y, L)	-
q_3	-	-	-	(q_3, y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Unit 5

(UNIT-5 (kloster))

1.)

Assume $H = \text{Head}$
 $T = \text{Tail}$

Answer Armed
Answer

List $R = (T, H, HT, TT)$

List $S = (TH, TH, HT, T)$

Now, we have to find out a sequence
that strings formed by R and S are identical
such a sequence is 1, 2, 1, 3, 3, 4.

Hence from the R and S list

1	2	1	3	3	4	1	2	1	3	3	4
T	H	T	HT	HT	TT	TH	TH	TH	HT	HT	T

(or)

i	List R w_i	List S x_i
1	T	TH
2	H	TH
3	HT	HT
4	TT	T

Take $M = 5$

Take the combination 1 2 1 3 3 4

THHTHT HTHTT = THTHHTHTT

Instance of P/P = 1 2 1 3 3 4

4) In Endc, there - -
5) Country sort takes $O(n+k)$ time and $O(n+k)$ space, where n is the number of items we're sorting and k is the no. of possible values.

We iterate through the input items twice - once to populate counts and once to fill in the output array. Both iterate are $O(n)$ time. Additionally, we iterate through counts once to fill in next index, which is $O(k)$ time.

The algorithm allocates three additional arrays one for counts one for next index, and one for the output. The first two are $O(k)$ space and the final one is $O(n)$ space.