

STUDENT PORTFOLIO



Name: ANAS AHMED AATHER
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 Srikaanth (102776)

Regex Module of Hackerrank

SUBDOMAINS



Introduction

Q.1. Matching Specific String

HackerRank [Prepare](#) > [Regex](#) > [Introduction](#) > [Matching Specific String](#) [Exit Full Screen View](#)

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.

Problem Submissions Leaderboard Discussions

```
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))
```

Change Theme Language Python 3

Run Code Submit Code

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.

HackerRank [Prepare](#) > [Regex](#) > [Introduction](#) > [Matching Specific String](#) [Exit Full Screen View](#)

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.

Problem Submissions Leaderboard Discussions

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)
	1 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.
Test case 2	Expected Output
	1 Number of matches : 2

Run Code Submit Code

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.

Q.2. Matching digit and non digit character

HackerRank Prepare > Regex > Introduction > Matching Digits & Non-Digit Characters

Exit Full Screen View

Problem

\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.

Type here to search

HackerRank Prepare > Regex > Introduction > Matching Digits & Non-Digit Characters

Exit Full Screen View

Change Theme Language Python 3

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

Line: 2 Col: 1

Run Code Submit Code

Upload Code as File Test against custom input

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 Input (stdin) Download

1 06-11-2015

Test case 3

Test case 4 Expected Output Download

1 true

Test case 5

Desktop 12:01 PM 13-11-2022

Problem

\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.

Type here to search

HackerRank Prepare > Regex > Introduction > Matching Digits & Non-Digit Characters

Exit Full Screen View

Upload Code as File Test against custom input

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 Input (stdin) Download

1 06-11-2015

Test case 3

Test case 4 Expected Output Download

1 true

Test case 5

Desktop 12:02 PM 13-11-2022

Q.3. Matching anything but a newline

HackerRank Prepare > Regex > Introduction > Matching Anything But a Newline

Exit Full Screen View

Problem

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

Regex Pattern
A.B.C.D.

Test String
A+B-C=DE

Code Editor

```
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())
```

Change Theme Language Python 3

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 \\..

HackerRank Prepare > Regex > Introduction > Matching Anything But a Newline

Exit Full Screen View

Problem

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

Regex Pattern
A.B.C.D.

Test String
A+B-C=DE

Code Editor

Upload Code as File Test against custom input

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

Run Code Submit Code

Test Cases

Test case 0	Compiler Message
Success	Success
Test case 1	
Test case 2	
Test case 3	
Test case 4	
Test case 5	
Test case 6	

Input (stdin) Download
1 123.456.abc.def

Expected Output Download
1 true

Next Challenge

Type here to search

Desktop 12:03 PM ENG 13-11-2022

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 (**_**).

Regex Pattern

\w\w\w

↓

Test String

\$one

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

Change Theme Language Python 3

```
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 (**_**).

Regex Pattern

\w\w\w

↓

Test String

\$one

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	Compiler Message
Success	
Test case 1	Input (stdin)
Success	1 www.hackerrank.com
Test case 2	Download
Test case 3	Input (stdin)
Success	1 true
Test case 4	Expected Output
Success	Download
Test case 5	Input (stdin)

Desktop 12:05 PM 13-11-2022

Prepare > Regex

Regex

Points: 310 Rank: 17697

Matching Specific String

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



Solved

STATUS Solved Unsolved

Matching Anything But a Newline

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



Solved

DIFFICULTY Easy Medium Hard

Matching Digits & Non-Digit Characters

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



Solved

SUBDOMAINS Introduction Character Class Repetitions Grouping and Capturing Backreferences Assertions

Matching Word & Non-Word Character

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



Solved



Character Class

Q.5. Matching Specific character

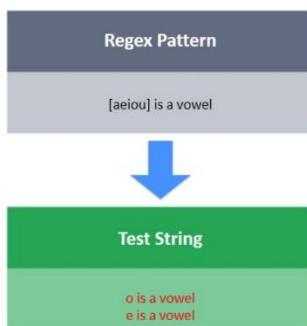
HackerRank [Prepare](#) > [Regex](#) > [Character Class](#) > [Matching Specific Characters](#)

Exit Full Screen View

Problem

[]

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



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

Task

Type here to search

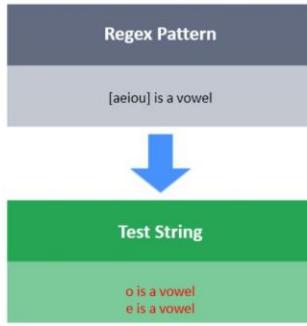
HackerRank [Prepare](#) > [Regex](#) > [Character Class](#) > [Matching Specific Characters](#)

Exit Full Screen View

Problem

[]

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



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

Task

Type here to search

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	Success
Test case 1	Input (stdin)
1 123x.	Download
Test case 2	Expected Output
1 true	Download
Test case 3	
Test case 4	
Test case 5	
Test case 6	

Desktop 12:06 PM ENG 13-11-2022

Q.6.Excluding specific character

HackerRank Prepare > Regex > Character Class > Excluding Specific Characters

Exit Full Screen View

Problem

[^]

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

Regex Pattern

[^aeiou] is not a vowel

Test String

k is not a vowel
p is not a vowel

Change Theme Language Python 3

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())

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

Task

Type here to search

HackerRank Prepare > Regex > Character Class > Excluding Specific Characters

Exit Full Screen View

Problem

[^]

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

Regex Pattern

[^aeiou] is not a vowel

Test String

k is not a vowel
p is not a vowel

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 Compiler Message Success

Test case 1 Input (stdin) Download

Test case 2 Input (stdin)
1 think?

Test case 3 Input (stdin)

Test case 4 Expected Output Download

Test case 5 Input (stdin)

Test case 6 Input (stdin)

Input (stdin)
1 true

Download

Desktop 12:08 PM 13-11-2022

Type here to search

HackerRank Prepare > Regex > Character Class > Excluding Specific Characters

Exit Full Screen View

Q.7. Matching character ranges

HackerRank [Prepare](#) > [Regex](#) > [Character Class](#) > [Matching Character Ranges](#) Exit Full Screen View

Problem

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]
```

Code Editor

```
1 Regex_Pattern = r'^[a-z][1-9][^a-z][^A-Z][A-Z]' # Do not delete 'r'.
2 
3 import re
4 
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Change Theme Language Python 3

Line: 1 Col: 48

Submissions

Type here to search

HackerRank [Prepare](#) > [Regex](#) > [Character Class](#) > [Matching Character Ranges](#) Exit Full Screen View

Problem

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]
```

Code Editor

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 cases

Test case 0	Compiler Message
Success	Success

Input (stdin) Download

```
h4CkR
```

Expected Output Download

```
true
```

Compiler Message Success

Input (stdin) Download

Expected Output Download

Desktop 12:09 PM ENG 13-11-2022

Prepare > Regex

Regex

Points: 310 Rank: 17697

Matching Specific Characters

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



Solved

Excluding Specific Characters

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



Solved

Matching Character Ranges

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



Solved

STATUS

- Solved
- Unsolved

DIFFICULTY

- Easy
- Medium
- Hard

SUBDOMAINS

- Introduction
- Character Class
- Repetitions
- Grouping and Capturing
- Backreferences
- Assertions
- Applications



Repetitions

Q.8. Matching {x} repetition

HackerRank [Prepare](#) > [Regex](#) > [Repetitions](#) Matching {x} Repetitions

Exit Full Screen View

Problem

{x}

The tool {x} will match exactly x repetitions of character/character class/groups.

Regex Pattern

$\backslash w\{4\}$

Test String

H_ck

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

For Example:

w{3}: It will match the character w exactly 3 times.

Code Editor

```
1 Regex_Pattern = r'^[a-zA-Z]{40}[13579\s]{5}$' # Do not delete 'r'.
2
3 import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Change Theme Language Python 3

Submissions

Leaderboard

Discussions

HackerRank [Prepare](#) > [Regex](#) > [Repetitions](#) Matching {x} Repetitions

Exit Full Screen View

Problem

{x}

The tool {x} will match exactly x repetitions of character/character class/groups.

Regex Pattern

$\backslash w\{4\}$

Test String

H_ck

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

For Example:

w{3}: It will match the character w exactly 3 times.

Code Editor

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 Compiler Message Success

Test case 1

Test case 2 Input (stdin) Download

222222222aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa13 57

Test case 3

Test case 4 Expected Output Download

true

Test case 5

Test case 6

Desktop 12:10 PM 13-11-2022

Q.9. Matching {x,y} Repitions

HackerRank Prepare > Regex > Repetitions > Matching {x,y} Repetitions

Exit Full Screen View

Problem

{x,y}

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

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:

Line: 1 Col: 48

HackerRank Prepare > Regex > Repetitions > Matching {x,y} Repetitions

Exit Full Screen View

Problem

{x,y}

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

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:

Run Code

Submit Code

Congratulations

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

Next Challenge

Test case 0

Compiler Message

Success

Test case 1

Input (stdin)

3three more alphabets.

Download

Test case 2

Expected Output

true

Download

Test case 3

Test case 4

Test case 5

Test case 6

Desktop 12:11 PM 13-11-2022

Q.10. Matching zero and more repetitions

HackerRank Prepare > Regex > Repetitions > Matching Zero Or More Repetitions

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

```
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())
```

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.

HackerRank Prepare > Regex > Repetitions > Matching Zero Or More Repetitions

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

Congratulations

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

Test case	Compiler Message
Test case 0	Success
Test case 1	Input (stdin)
Test case 2	14
Test case 3	Expected Output
Test case 4	true
Test case 5	Download
Test case 6	Download

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.

HackerRank NEW PREPARE CERTIFY COMPETE

Search | aa0094 ▾

Prepare > Regex

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

Submissions

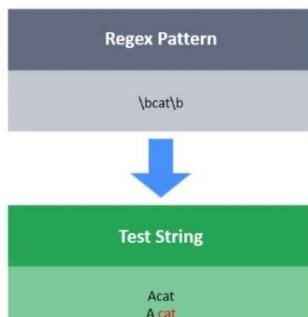
Leaderboard

Discussions

\b
assert position at a word boundary.

Three different positions qualify for word boundaries :

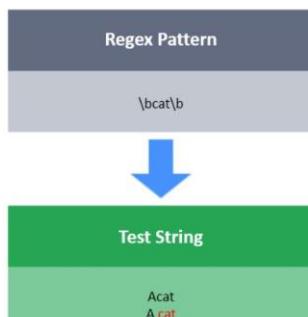
- ▶ Before the first character in the string, if the first character is a word character.
- ▶ Between two characters in the string, where one is a word character and the other is not a word character.
- ▶ After the last character in the string, if the last character is a word character.



\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.



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

Change Theme Language Python 3

Exit Full Screen View

Line: 1 Col: 44

Desktop > 12:13 PM ENG 13-11-2022

Submissions

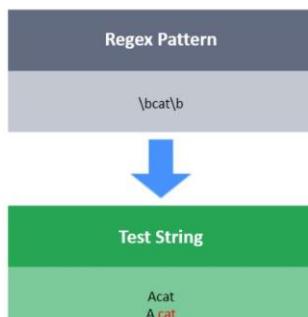
Leaderboard

Discussions

\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.



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 Found any match?

Test case 2

Expected Output

Download

1 true

Test case 3

Compiler Message

Success

Test case 4

Input (stdin)

Download

1 Found any match?

Test case 5

Expected Output

Download

1 true

Test case 6

Desktop > 12:14 PM ENG 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)

Regex Pattern

It is (not)? your fault

Test String

It is not your fault
It is your fault

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)

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

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	Compiler Message
Success	
Test case 1	Input (stdin)
	1 okokok! cya
Test case 2	Download
Test case 3	Expected Output
1 true	Download
Test case 4	
Test case 5	
Test case 6	

Desktop 12:14 PM ENG 13-11-2022

Desktop 12:15 PM ENG 13-11-2022

Q.13. Alternative Matching

HackerRank [Prepare](#) > [Regex](#) > [Grouping and Capturing](#) Alternative Matching

Exit Full Screen View

Problem Submissions Leaderboard Discussions

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.

Regex Pattern
(and|AND|And)

Test String
And the award goes to
A and D Company



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

Code Editor:

```
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 Submissions Leaderboard Discussions

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.

Regex Pattern
(and|AND|And)

Test String
And the award goes to
A and D Company



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

Code Editor:

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	Compiler Message
Success	
Test case 1	Download
Test case 2	Download
Test case 3	Download
Test case 4	Download
Test case 5	Download
Test case 6	Download

Input (stdin) Download
1 Mr.DOSHI

Expected Output Download
1 true

Desktop 12:16 PM ENG 13-11-2022

Prepare > Regex

Points: 310 Rank: 17697

Regex

Matching Word Boundaries

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



Solved

Capturing & Non-Capturing Groups

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



Solved

Alternative Matching

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



Solved

STATUS

-
- Solved
-
-
- Unsolved

DIFFICULTY

-
- Easy
-
-
- Medium
-
-
- Hard

SUBDOMAINS

-
- Introduction
-
-
- Character Class
-
-
- Repetitions
-
-
- Grouping and Capturing



Backreferences

Q.14. Matching same text again and again

HackerRank Prepare > Regex > Backreferences > Matching Same Text Again & Again

Problem

\group_number

This tool (`\1` references the first capturing group) matches the same text as previously matched by the capturing group.

Regex Pattern

Test String

malayalam

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

For Example:

HackerRank Prepare > Regex > Backreferences > Matching Same Text Again & Again

```
1 Regex_Pattern = r'^([a-z]\w\s\W\d[A-Z][a-zA-Z][aieouAEIOU]\S)\1$' # Do not
   delete 'r'.
2
3 v import re
4
5 print(str(bool(re.search(Regex_Pattern, input()))).lower())
```

Change Theme Language Python 3

Exit Full Screen View

HackerRank Prepare > Regex > Backreferences > Matching Same Text Again & Again

Problem

\group_number

This tool (`\1` references the first capturing group) matches the same text as previously matched by the capturing group.

Regex Pattern

Test String

malayalam

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

For Example:

HackerRank Prepare > Regex > Backreferences > Matching Same Text Again & Again

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 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin)

```
1 ab #1?AZa$ab #1?AZa$
```

Download

Expected Output

```
1 true
```

Download

Compiler Message Success

Input (stdin)

```
1 ab #1?AZa$ab #1?AZa$
```

Download

Expected Output

```
1 true
```

Download

Desktop 12:18 PM ENG 13-11-2022

Q.15.Branch reset group

HackerRank Prepare > Regex > Backreferences > Branch Reset Groups

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.

Problem Submissions Leaderboard Discussions

Regex Pattern

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

Test String

```
Haahaa  
kk
```

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

Type here to search

12:20 PM ENG 13-11-2022

Change Theme Language PHP

Exit Full Screen View

HackerRank Prepare > Regex > Backreferences > Branch Reset Groups

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.

Problem Submissions Leaderboard Discussions

Regex Pattern

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

Test String

```
Haahaa  
kk
```

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

Type here to search

12:20 PM ENG 13-11-2022

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 Compiler Message Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin) Download

```
1 12-34-56-78
```

Expected Output Download

```
1 true
```

12:20 PM ENG 13-11-2022

Change Theme Language PHP

Exit Full Screen View

Q.16. Forward References

HackerRank [Prepare](#) > [Regex](#) > [Backreferences](#) > [Forward References](#)

NOTE - Forward reference is supported by JGsoft, .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

```
golgolamigo
```

Change Theme Language PHP

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

Line: 3 Col: 37

Submissions Leaderboard Discussions

Type here to search

HackerRank [Prepare](#) > [Regex](#) > [Backreferences](#) > [Forward References](#)

NOTE - Forward reference is supported by JGsoft, .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

```
golgolamigo
```

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	Compiler Message
Success	
Test case 1	Success
Test case 2	Input (stdin)
tactic	Download
Test case 3	Expected Output
true	Download
Test case 4	
Test case 5	

Submissions Leaderboard Discussions

Type here to search

HackerRank NEW PREPARE CERTIFY COMPETE

Search | aa0094

Prepare > Regex

Regex

Points: 310 Rank: 17697

Matching Same Text Again & Again
Easy, Max Score: 20, Success Rate: 97.98% Solved

Branch Reset Groups
Easy, Max Score: 20, Success Rate: 95.31% Solved

Forward References
Easy, Max Score: 20, Success Rate: 89.19% Solved

STATUS

Solved Unsolved

DIFFICULTY

Easy Medium Hard

SUBDOMAINS

Introduction Character Class Repetitions Grouping and Capturing Backreferences



Assertions

Q.17. Positive Lookahead

HackerRank

Prepare

> Regex > Assertions

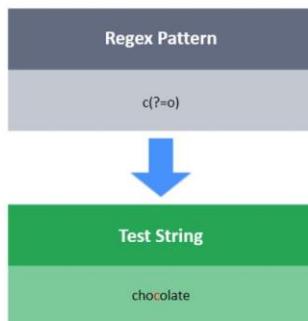
Positive Lookahead

Exit Full Screen View

Problem

Submissions
Leaderboard
Discussions

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.



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

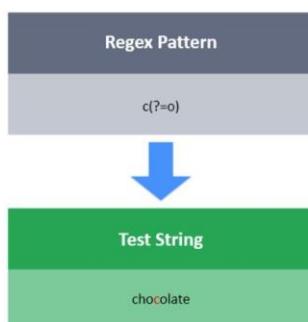
```
Change Theme Language Python 2
1 Regex_Pattern = r'c(?=oo)' # Do not delete 'r'.
2
3 > import re...
```

Line: 1 Col: 26

Problem

Submissions
Leaderboard
Discussions

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.



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

Type here to search

Desktop ENG 12:23 PM 13-11-2022 Exit Full Screen View

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	Compiler Message
Success	
Test case 1	Success
Test case 2	Input (stdin)
gooooo!	Download
Test case 3	Expected Output
Number of matches : 3	Download

Desktop ENG 12:23 PM 13-11-2022

Q.18. Positive Lockbehind

HackerRank Prepare > Regex > Assertions 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.

```
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)
```

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

HackerRank Prepare > Regex > Assertions 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.

Congratulations

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

Test case 0	Compiler Message
Success	

Test case 1	Input (stdin)	Download
Success	1 12366!	

Test case 2	Input (stdin)	Download
Success	1 12366!	

Test case 3	Expected Output	Download
Success	1 Number of matches : 1	

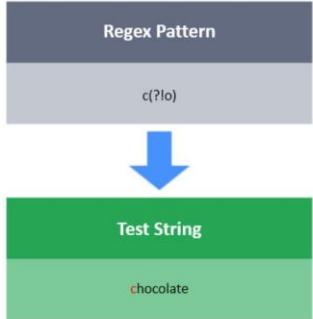
In above image Regex Pattern is matched with the Test String.

Q.19. Negative Lookahead

HackerRank Prepare > Regex > Assertions Negative Lookahead

Problem Submissions Leaderboard Discussions

`regex_1(?!regex_2)`
The negative lookahead (`?! regex_2`) 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.



```
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)
```

Change Theme Language Python 2

Exit Full Screen View

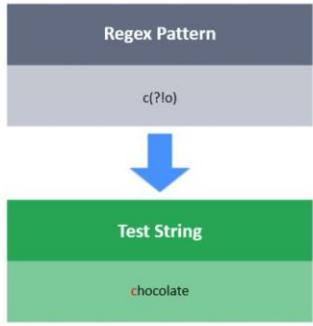
Line: 1 Col: 28

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

HackerRank Prepare > Regex > Assertions Negative Lookahead

Problem Submissions Leaderboard Discussions

`regex_1(?!regex_2)`
The negative lookahead (`?! regex_2`) 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.



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	Compiler Message
Success	Success
Test case 1	Input (stdin)
Success	gooooo
Test case 2	Input (stdin)
Success	Number of matches : 2
Test case 3	Expected Output
Success	Download

Desktop 12:25 PM 13-11-2022

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

Q.20. Negative Lookbehind

HackerRank [Prepare](#) > [Regex](#) > [Assertions](#) Negative Lookbehind

Exit Full Screen View

Problem Submissions Leaderboard Discussions

(?<!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.

```
1 Regex_Pattern = r"(?<![a-z])[aeiou]." # 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)
```

Change Theme Language Python 2

Line: 1 Col: 37

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

HackerRank [Prepare](#) > [Regex](#) > [Assertions](#) Negative Lookbehind

Exit Full Screen View

Problem Submissions Leaderboard Discussions

(?<!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.

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	Compiler Message
Success	
Test case 1	Input (stdin)
1 lois	Download
Test case 2	Expected Output
	Download
Test case 3	Number of matches : 3

Desktop 12:27 PM 13-11-2022

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

Prepare > Regex

Regex

Points: 310 Rank: 17697

Positive Lookahead

Easy, Max Score: 20, Success Rate: 99.50%



Solved

Negative Lookahead

Easy, Max Score: 20, Success Rate: 97.12%



Solved

Positive Lookbehind

Easy, Max Score: 20, Success Rate: 98.47%



Solved

Negative Lookbehind

Easy, Max Score: 20, Success Rate: 97.83%



Solved

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 Athar
01 CS-E-IT.

M.C.Q.

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

(2) (c) Trivial proof

3] Assume ~~for all k~~ (d) $m^3 + 3m^2k^2 + 3m^2k + 1 \equiv 9 \pmod{10}$

4.] (a)

Descriptive Question

1.) $f(m) = (2^m - 1) \cdot 3 = 0$

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

∴ $f(1)$ is true

Assume, $P(m)$ 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

∴ by mathematical induction statement

is proved.

2.) Let $a^2 = 3k - ①$

$$\text{and } a = 3k+1 - ②$$

$$\text{then } a = 3k+2 - ③$$

$$\text{on summing } ② \rightarrow a^2 = 9k^2 + 1 + 3k - ④$$

$$\text{on summing } ③ \rightarrow a^2 = 9k^2 + 4 + 6k - ⑤$$

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



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

$$a+b = 2n+1 + 2m \rightarrow 2(n+m) + 1 \text{ which is always odd}$$

(Proved)

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

by counter example

$$\begin{aligned} n &= 5 \\ P &\Rightarrow 2(5)^2 + 31 - 16(5) \\ \Rightarrow P &= 50 + 31 - 80 = -1 \end{aligned}$$

(positive)

$$\begin{aligned} n &= 4 \\ P &\Rightarrow 2(4)^2 + 31 - 16(4) \\ \Rightarrow P &= 32 + 31 - 64 = -1 \end{aligned}$$

(negative)

$\therefore P$ is not always true

(5) $\forall n > 5, 2n > n^2$

$$n > 5 \rightarrow 2n > n^2$$

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

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

$$\begin{aligned} \text{for } n = k+1 &\rightarrow 2(k+1) < (k+1)^2 \\ &\leq 2k+2 < k^2 + 1 + 2k \\ &= 2k < k^2 + 2k - 1 \end{aligned}$$

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

\therefore from induction it's proved.

6.)

$$P(n) = 1^2 + 2^2 + 3^2$$

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

for $n=1$

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

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

for $n=k+1 \rightarrow$

$$\begin{aligned} P(k) &= 1^2 + 2^2 + 3^2 + \dots + k^2 + (k+1)^2 \\ &= \frac{k(k+1)(2k+1) + (k+1)^2}{6} \\ &= \frac{(k+1)[k(2k+1) + (6k+1)]}{6} \\ &= \frac{(k+1)(k+2)(2k+3)}{6} \end{aligned}$$

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

Unit 2

CTI-1 Worksheet -2

RA2011031010006
Anay Ahmed Ather

- 1.) (i) Set of all strings starting and ending
of 1's in 6100 '10'.
Part-A.
O1.
CSE-IT
- 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 three 0's.

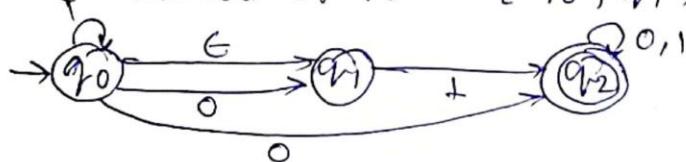
Part-B:-

- 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 $(String)^*$ where * signifies a repetition
of for more than 1 times.

$$\text{Ex} \quad (a+b)^* (ab)^* (a+b)$$

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

E closure of $q_0 \rightarrow \{q_0, q_1, q_2\}$.



\leftarrow closure of q_0

$\rightarrow \epsilon^*$ or ϵ^*

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

$\epsilon^* \downarrow \epsilon^*$

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

∴ ϵ^* 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 function $f: Q \times \Sigma \rightarrow Q$

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 states

$\delta \rightarrow$ transition function $f: Q \times \Sigma \rightarrow 2^Q$

Unit 3

- 1) (i) Set of all strings starting and ending
of 1's in 610110! Part-A
CSE-IT
Q1.
- 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 three 0's.

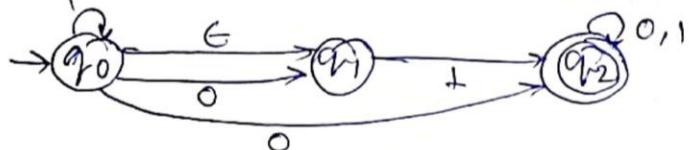
Part-B-

- 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 subset of the alphabet set and
use $(\text{String})^*$ where * signifies a repetition
of for more than 1 times.

$$\text{Ex} \quad (a+b)^* (ab)^* (a+b)$$

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

$$\epsilon \text{ closure of } q_0 \rightarrow \{q_0, q_1, q_2\}$$



Unit 4

Unit - 4 (Worksheet) - 1

Anas Ahmed
Atwal.

- 1) True
- 2) True
- 3) (b) 7
- 4) A. Left
- 5) A. Both S_1 and S_2 are three.
- 6)

Scenario based

- 1) Transition diagram
-
- ```

graph LR
 Start((Start)) -- "a/x" --> q0((q0))
 q0 -- "b/y" --> q1((q1))
 q1 -- "c/z" --> q2((q2))
 q2 -- "x/a" --> q3((q3))
 q3 -- "y/b" --> q4((q4))
 q4 -- "y1/y" --> q1
 q2 -- "z1/z" --> q2
 q3 -- "b1/b" --> q3
 q4 -- "x1/x" --> q4

```
- 2) TM M 1)
  $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$ 
 where  $\delta$  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 (klonesnel.)

1) Assume H = Head  
T = Tail

Anees Ahmed  
Anees

List R = (T, H, HTH, 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 HTH HTH TT TH TH HT HT T

(or )

| i | List R | List S |
|---|--------|--------|
| 0 | 0      | x1     |
| 1 | T      | TH     |
| 2 | H      | TH     |
| 3 | HTH    | HT     |
| 4 | TT     | T      |

Take M = 5

Take the combination 1 2 1 3 3 4

THHTHHTH HTHTTT = TH THHTHHTHHT

Instance of P(P) = 1 2 1 3 3 4

4) In Ercole, there -

5) Counting 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 - one to populate counts and one to fill in the output array. Both iterations 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 count, one for next index, and one for the output. The first two are  $O(k)$  space and the final one is  $O(n)$  space.