# Содержит ли строка подстроку?

```
foreach ( @ARRAY ) {
    /required substring/ or next;
}
```

# Проголосуем

```
sub t_RegExp {
    my $cnt = 0;
    foreach ( @ARRAY ) {
        /required substring/ or next;
        ++$cnt;
    }
    return $cnt;
}
```

```
use Benchmark qw(cmpthese);

my $subs = {
    RegExp ⇒ sub {
        my $cnt = t_RegExp();
        $realCnt == $cnt
            or die "$cnt ≠ $realCnt ".__LINE__;
    },
};

cmpthese( -20, $subs );
```

```
./1.pl
          Rate RegExp
RegExp 104/s     --

./1.pl
          Rate RegExp
RegExp 106/s     --

./1.pl
           Rate RegExp
RegExp 99.7/s     --
```

Хотелось бы, чтобы результат был максимально стабилен

```
sudo nice -n -20 ./1.pl
          Rate RegExp
RegExp 103/s     --

sudo nice -n -20 ./1.pl
          Rate RegExp
RegExp 107/s     --

sudo nice -n -20 ./1.pl
          Rate RegExp
RegExp 106/s     --
```

```
RegExp ⇒ sub {
    for ( 1 .. 100 ) {
        my $cnt = t_RegExp();
        $realCnt == $cnt
            or die "$cnt ≠ $realCnt ".__LINE__;
    }
}
```

```
./2.pl && ./2.pl && ./2.pl && ./2.pl
        s/iter RegExp
RegExp    1.05      --
        s/iter RegExp
RegExp    1.04      --
        s/iter RegExp
RegExp    1.06      --
        s/iter RegExp
RegExp    1.04      --
```

```
/required substring/ or next;

index( $_, 'required substring' ) ⩾ 0 or next;
```

```
          s/iter RegExp  index
RegExp      1.06     --   -11%
index      0.947    12%     --
```

# Начинается ли строка с подстроки?

```
/^required substring/ or next;
```

```
/^required substring/ or next;

index( $_, 'required substring' ) == 0 or next;
```

# Проголосуем

```
          Rate RegExp  index
RegExp 1.43/s     --     -4%
index  1.49/s     4%      --
```

Это результат
для коротких строк,
а для длинных?

```
          Rate  index RegExp
index   1.09/s     --   -13%
RegExp  1.26/s    15%     --
```

```
my $substr = 'required substring';
my $length = length $substr;

substr( $_, 0, $length ) eq $substr or next;
```

# Проголосуем

```
           Rate  index RegExp substr
index  2.23/s     --   -14%   -16%
RegExp 2.61/s    17%     --    -2%
substr 2.65/s    19%     2%     --
```

Но на коротких...

```
          Rate substr RegExp  index
substr 2.80/s     --    -7%    -8%
RegExp 3.01/s     8%     --    -1%
index  3.05/s     9%     1%     --
```

```
rindex( $_, 'required substring', 0 ) == 0
    or next;
```

# Проголосуем

# Короткие

```
          Rate substr RegExp  index rindex
substr 2.73/s     --    -5%    -9%   -16%
RegExp 2.87/s     5%     --    -4%   -12%
index  2.99/s    10%     4%     --    -8%
rindex 3.25/s    19%    13%     8%     --
```

# Длинные

```
          Rate  index RegExp substr rindex
index  2.19/s     --   -10%   -12%   -27%
RegExp 2.45/s    12%     --    -2%   -18%
substr 2.51/s    14%     2%     --   -16%
rindex 2.99/s    36%    22%    19%     --
```

# Обработка JSON

```
zcat data.gz | head
{"i":0,"rand":0.397854047668265}
{"i":1,"rand":0.944829748102592}
{"i":2,"rand":0.330165124536631}
{"i":3,"rand":0.935485459255638}
{"i":4,"rand":0.881527066359087}
{"rand":0.479268929208033,"i":5}
{"rand":0.384015547714025,"i":6}
{"i":7,"rand":0.243269832845805}
{"i":8,"rand":0.111611173712308}
{"rand":0.219453575846956,"i":9}
```

```
my $sum = 0;

while ( my $str = <STDIN> ) {
    my $rec = decode_json( $str );
    $rec->{ rand } ⩾ 0.5 or next;
    $rec->{ i } % 10 == 0 or next;

    $sum += $rec->{ i };
}

say $sum;
```

# Проголосуем

```
my $json = '[';
my $bytes_read =
  read( STDIN, $json, $once_read, length $json );
defined $bytes_read
    or die "Can't read from STDIN";
$json .= <STDIN>;
chomp( $json );
$json .= ']';
$json =~ tr/\n/,/;

my $answer = decode_json( $json );
if ( @$answer == 0 )
{
    $answer = undef;
}
return $answer;
```

```
while ( my $data = JsonInStrBulk::get_data() )
{
    foreach my $rec ( @$data )
    {
        $rec->{ rand } ⩾ 0.5 or next;
        $rec->{ i } % 10 == 0 or next;

        $sum += $rec->{ i };
    }
}
```

# Проголосуем

```
time bash -c 'zcat data.gz | ./1.pl'
249994300181880
bash -c 'zcat data.gz | ./1.pl'  97,17s user 2,11s system 138%
cpu 1:11,89 total


time bash -c 'zcat data.gz | ./2.pl'
249994300181880
bash -c 'zcat data.gz | ./2.pl'  97,79s user 1,69s system 110%
cpu 1:30,43 total
```

1)
97,17s user 2,11s system 138% cpu 1:11,89 total

2)
97,79s user 1,69s system 110% cpu 1:30,43 total

72 сек VS 90 сек

Второй вариант медленнее на 25%

Попытаемся понять почему с помощью NYTProf

## Top 15 Subroutines

| Calls | P | F | Exclusive Time | Inclusive Time | Subroutine |
|---|---|---|---|---|---|
| 15217 | 1 | 1 | 27.9s | 27.9s | JsonInStrBulk::CORE:read (opcode) |
| 15217 | 1 | 1 | 25.9s | 25.9s | JSON::XS::decode_json (xsub) |
| 15217 | 1 | 1 | 17.8s | 71.7s | JsonInStrBulk::get_data |
| 15217 | 1 | 1 | 15.9ms | 15.9ms | JsonInStrBulk::CORE:rcatline (opcode) |

```
my $json = '[';
my $bytes_read =
  read( STDIN, $json, $once_read, length $json );
defined $bytes_read
    or die "Can't read from STDIN";
$json .= <STDIN>;
chomp( $json );
$json .= ']';
$json =~ tr/\n/,/;

my $answer = decode_json( $json );
if ( @$answer == 0 )
{
    $answer = undef;
}
return $answer;
```

| | | | |
|---|---|---|---|
| 6.69ms | | | `my $json = '[';` |
| 28.0s | 15217 | 27.9s | `my $bytes_read = read( STDIN, $json, $once_read, length $json );`<br>`# spent  27.9s making 15217 calls to JsonInStrBulk::CORE:read, avg 1.84ms/call` |
| 3.79ms | | | `defined $bytes_read or die "Can't read from STDIN";` |
| 47.7ms | 15217 | 15.9ms | `$json .= <STDIN>;`<br>`# spent  15.9ms making 15217 calls to JsonInStrBulk::CORE:rcatline, avg 1µs/call` |
| 6.29ms | | | `chomp( $json );` |
| 5.75ms | | | `$json .= ']';` |
| 17.6s | | | `$json =~ tr/\n/,/;` |
| | | | |
| 26.0s | 15217 | 25.9s | `my $answer = decode_json( $json );`<br>`# spent  25.9s making 15217 calls to JSON::XS::decode_json, avg 1.70ms/call` |

| | | | |
|---|---|---|---|
| 28.0s | 15217 | 27.9s | `my $bytes_read = read( STDIN, $json,`<br>`# spent  27.9s making 15217 calls to` |
| 3.79ms | | | `defined $bytes_read or die "Can't rea` |
| 47.7ms | 15217 | 15.9ms | `$json .= <STDIN>;`<br>`# spent  15.9ms making 15217 calls to` |
| 6.29ms | | | `chomp( $json );` |
| 5.75ms | | | `$json .= ']';` |
| 17.6s | | | `$json =~ tr/\n/,/;` |
| | | | |
| 26.0s | 15217 | 25.9s | `my $answer = decode_json( $json );` |

```
my $subs = {
    '$str =~ tr' => sub {
        for ( 1 .. 100 ) {
            my $str = '\n' x 1000;
            $str =~ tr/\n/,/;
        }
    },
};
```

https://perlbanjo.com/170e39eb5e

| | | |
|---|---|---|
| 5.10.1 | 23.8/s | 22.7/s |
| 5.20.3 | 62.5/s | 55.6/s |
| 5.30.3 | 41.7/s | 38.5/s |
| 5.40.0 | 62.5/s | 62.5/s |