paladin8's blog

Z Algorithm

By **paladin8**, 4 years ago,

This is the first post in my Codeforces blog! I plan to use this as a place where I can write down new algorithms I learned, problems that I found interesting, or stupid mistakes I made during contests.

Today, the topic will be the Z Algorithm, which I learned as a result of failing to solve Problem B of Beta Round 93 (http://codeforces.com/contest/126/problem/B). There are some other solutions like binary search + hashing, but this one is quite nice. Anyway, first, a description of the algorithm and why it works; it is simple and makes a lot of sense (as all good algorithms are).

Algorithm

Given a string S of length n, the Z Algorithm produces an array Z where $Z[i]$ is the length of the longest substring starting from $S[i]$ which is also a prefix of S, i.e. the maximum k such that $S[j] = S[i + j]$ for all $0 \le j < k$. Note that $Z[i] = 0$ means that $S[0] \neq S[i]$. For easier terminology, we will refer to substrings which are also a prefix as prefix-substrings.

The algorithm relies on a single, crucial invariant. As we iterate over the letters in the string (index i from 1 to n - 1), we maintain an interval $[L, R]$ which is the interval with maximum R such that $1 \le L \le i \le R$ and $S[L...R]$ is a prefix-substring (if no such interval exists, just let $L = R =$ - 1). For $i = 1$, we can simply compute L and R by comparing $S[0...]$ to $S[1...]$. Moreover, we also get $Z[1]$ during this.

Now suppose we have the correct interval $[L, R]$ for i - 1 and all of the Z values up to i - 1. We will compute $Z[i]$ and the new $[L, R]$ by the following steps:

- If $i > R$, then there does not exist a prefix-substring of S that starts before i and ends at or after i. If such a substring existed, $[L, R]$ would have been the interval for that substring rather than its current value. Thus we "reset" and compute a new $[L, R]$ by comparing $S[0...]$ to $S[i...]$ and get $Z[i]$ at the same time ($Z[i] = R - L + 1$).
- Otherwise, $i \le R$, so the current $[L, R]$ extends at least to i. Let $k = i - L$. We know that $Z[i] \ge min(Z[k], R - i + 1)$ because $S[i...]$ matches $S[k...]$ for at least $R - i + 1$ characters (they are in the $[L, R]$ interval which we know to be a prefix-substring). Now we have a few more cases to consider.
- If $Z[k] < R - i + 1$, then there is no longer prefix-substring starting at $S[i]$ (or else $Z[k]$ would be larger), meaning $Z[i] = Z[k]$ and $[L, R]$ stays the same. The latter is true because $[L, R]$ only changes if there is a prefix-substring starting at $S[i]$ that extends beyond R, which we know is not the case here.
- If $Z[k] \ge R - i + 1$, then it is possible for $S[i...]$ to match $S[0...]$ for more than $R - i + 1$ characters (i.e. past position R). Thus we need to update $[L, R]$ by setting $L = i$ and matching from $S[R + 1]$ forward to obtain the new R. Again, we get $Z[i]$ during this.

The process computes all of the Z values in a single pass over the string, so we're done. Correctness is inherent in the algorithm and is pretty intuitively clear.

Analysis

We claim that the algorithm runs in $O(n)$ time, and the argument is straightforward. We never compare characters at positions less than R, and every time we match a character R increases by one, so there are at most n comparisons there. Lastly, we can only mismatch

once for each i (it causes R to stop increasing), so that's another at most n comparisons, giving $O(n)$ total.

Code

Simple and short. Note that the optimization $L = R = i$ is used when $S[0] \neq S[i]$ (it doesn't affect the algorithm since at the next iteration $i > R$ regardless).

```
int L = 0, R = 0;
for (int i = 1; i < n; i++) {
  if (i > R) {
    L = R = i;
    while (R < n && s[R-L] == s[R]) R++;
    z[i] = R-L; R--;
  } else {
    int k = i-L;
    if (z[k] < R-i+1) z[i] = z[k];
    else {
      L = i;
      while (R < n && s[R-L] == s[R]) R++;
      z[i] = R-L; R--;
    }
  }
}
```

Application

One application of the Z Algorithm is for the standard string matching problem of finding matches for a pattern T of length m in a string S of length n. We can do this in $O(n + m)$ time by using the Z Algorithm on the string $T\,\Phi\,S$ (that is, concatenating T, Φ, and S) where Φ is a character that matches nothing. The indices i with $Z[i] = m$ correspond to matches of T in S.

Lastly, to solve Problem B of Beta Round 93, we simply compute Z for the given string S, then iterate from i to n - 1. If $Z[i] = n - i$ then we know the suffix from $S[i]$ is a prefix, and if the largest Z value we've seen so far is at least $n - i$, then we know some string inside also matches that prefix. That gives the result.

```
int maxz = 0, res = 0;
for (int i = 1; i < n; i++) {
  if (z[i] == n-i && maxz >= n-i) { res = n-i; break; }
  maxz = max(maxz, z[i]);
}
```

algorithm, **beta round 93**, **string**, **tutorial**, **zalgorithm**

+107 paladin8 4 years ago 61

Comments (61)

Write comment?

thphong

4 years ago, # | +21

The are many good blogs on Codeforces about algorithm. If they are post in same place (as tutorial) is so great.

→ Reply

kvtoraman

4 years ago, # ^ | -11

so you dont like?

→ Reply

wil93

4 years ago, # ^ | +2

He simply said that there should be a place where all useful blog posts like this are categorized

→ Reply

mahmoudhassan → IOI 2016 Russia promo video

code_fille → Find count of subarrays having sum in a given range in less than O(n^2).

Petr → A week with old self

Xellos → Invitation to October Clash on HackerEarth

Alex7 → The new rating calculation system is terrible

aslf010990 → programming competitive books

lperovskaya → Three Subregionals Cup 2015

GuralTOO → COCI

PrinceOfPersia → Codeforces Round #326

bk2dcradle → Coding Calendar | Get Contest times online

MikhailRubinchik → eertree on iwoca 2015

The-Legend → Dose the gym work during contests?

paladin8 → Z Algorithm

dunpeal → How and where to find interesting problemsets with good test suites and editorials?

Errichto → Very short Editorial of SRM #671

flash_7 → 2015-2016 ACM-ICPC, NEERC, Southern Subregional Contest

marek.cygan → Marathon 24 — registration extended by 12 hours

homo_sapiens → Разбор Codeforces Round #325

allllekssssa → HackerRank HourRank 1

Ordan → Help with 588E

raihatneloy → Help needed in a Data Structure Problem

Detailed →

paladin8

4 years ago, # ^ | +3

I agree. Maybe systematically tagging them with "tutorial" would do the trick.

→ Reply

abinash

10 months ago, # ^ | 0

Here I post some tutorial link.

If you know any new tutorial blog post please comment , I will add them .

Thanks **paladin8** for his awesome tutorial.

→ Reply

casher

10 months ago, # ^ | +2

Here also by using Z-algorithm 6780044

→ Reply

Zahra.H

6 months ago, # ^ | 0

100% agree with u . it would be a great source for everybody to learn algorithm :)

→ Reply

ivan.popelyshev

4 years ago, # | 0

e-maxx has a site with algorithms, but it's on russian:

http://translate.google.ru/translate?sl=ru&tl=en&js=n&prev=_t&hl=ru&ie=UTF-8&layout=2&eotf=1&u=http%3A%2F%2Fe-maxx.ru%2Falgo%2F&act=url

→ Reply

Empty

3 years ago, # ^ | 0

That site is awesome, if someone could translate it to english, i think that it would be the best algorithm reference ever.

→ Reply

akash_goel

3 months ago, # ^ | 0

It automatically translates to English for my region! Great!

→ Reply

shahidul_brur

10 days ago, # ^ | 0

Here is the english version of this site: http://e-maxx-eng.github.io

→ Reply

optimistic_fighter

3 years ago, # ^ | ← Rev. 3 0

thanx for sharing this source with us :)

→ Reply

10 days ago, # ^ | 0

Non Russians can use this site to read the tutorials in English: http://e-maxx-eng.github.io

Or if one visit that site using google chrome, then s/he can

shahidul_brur

Or if one visit that site using google chrome, then s/he can automaticaly translate in English with the help of google chrome browser.

Also I have a pdf file in English containing all posts on this site.

→ Reply

Darooha

4 years ago, # | +16

The KMP algorithm computes memo[i] which is the length of the longest prefix of s that matches the tail of s[1], s[2], ... s[i]. What are the advantages of the z-algorithm over KMP? Here's my code using KMP that solves the Codeforces password problem.

```
import java.io.*;
import java.util.*;

public class D {
   public static void main(String[] args) throws IOException{
   BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
   String s = in.readLine();
   char[] c = s.toCharArray();
   int n = c.length;
   int[] memo = new int[n];
   /*  memo[i] will store the length of the longest prefix of s that matches the
tail of s1...si */

   for (int i=1; i<n; i++) {
      int j=i;
      while (j>0 && c[i] != c[memo[j-1]]) j = memo[j-1];
      memo[i] = (j>0) ? memo[j-1]+1 : 0;
   }

   int max = 0;
   for (int i=1; i<n-1; i++) max = Math.max(max, memo[i]);
   /* max =  Maximum internal match to prefix  */
   j = memo[n-1];
   while (j>max) j = memo[j-1];
   System.out.println((j==0)? "Just a legend" : s.substring(0, j));
   }
}
```

→ Reply

paladin8

4 years ago, # ^ | +3

As far as I know, both approaches work fine. I learned KMP before, but it didn't seem very intuitive to me, so I never fully understood string matching. The Z Algorithm is, in my opinion, easier to comprehend (and maybe to code, too). This motivated me to write this blog post :)

→ Reply

MikeMirzayanov

4 years ago, # ^ | 0

Glad to see you here!

Yep, this problem could be solved using KMP. Moreover Z- and prefix-function are equivalent (one can transform Z<->prefix in O(n)). But for me often it is easier to think in terms of Z-function to solve a problem.

→ Reply

4 years ago, # ^ | 0

> one can transform Z<->prefix in O(n)

How?

SkidanovAlex

How?

→ Reply

Manish-Kumar

4 years ago, # ^ | ← Rev. 3 +3

may be this will work `for i = 1 to n p[i + z[i]] = max(p[i + z[i]], z[i])`

and one more travesing from n to 1. doing `p[i] = max(p[i+1] - 1, p[i])`

→ Reply

anfuve.13

6 months ago, # ^ | 0

"and one more travesing from n to 1. doing p[i] = max(p[i+1] — 1, p[i])"

why is this necessary ?

→ Reply

I_love_natalia

3 years ago, # ^ | ← Rev. 3 0

A portion of necroposting.

You can solve 2 problems: 1) Create a sample string with given Z-function over an infinite alphabet. Just follow Z-function generation algorithm and store all the equalities in DST. Next, check the result. 2) Create a sample string with given prefix-function over an infinite alphabet. Just follow prefix function generation algorithm and store all the equalities in DST. Next, check the result.

(in Russian: http://contest.samara.ru/ru/problemset/735/)

→ Reply

liruqi

2 years ago, # ^ | 0

I dont know if your Z<-> prefix transformation is mutual. I've written a post to try to construct z from kmp, http://liruqi.info/post/62511983824/kmp-z (In Chinese). Beside, you may check my code here, https://raw.github.com/liruqi/topcoder/master/HackerRank/SaveHumanity.cpp

→ Reply

Burunduk1

3 years ago, # ^ | ← Rev. 2 0

What are the advantages of the z-algorithm over KMP?

No advantages =)

But if we compare Z-function and Prefix-function then:

1) "monotony"

If z[i] = x, it means substrings [0..j) [i..i+j) are equals for all j from 0 to x.

If p[i] = x, it means [0..j) = (i-j..i] for all j = x, p[x], p[p[x]] and so on.

2) LCP (Largest Common Prefix)

Z-function in fact calculates LCP[0,j] for all j. It can be used for not only substring searching

only substring searching.

I also have two examples of problems which, I hope, show advantages Z-function over Prefix-function.

1) Determine number (No.) of the string in its suffix array in O(n).

2) Determine number (amount) of substrings, which have at least two occuarances in the string in $O(n^2)$ (of course, you can solve it in O(n) using suffix tree).

→ Reply

17 months ago, # ^ | ← Rev. 2 +3

Your comment is valuable. But you have some error, I will show them, to prevent others from misunderstanding. (Someone reopen the post, so I read your comment)

1) "monotony"

kien_coi_1997

If z[i] = x, it means substrings **[0..j) [i..i+j)** are equals for all j from 0 to x-1.

If p[i] = x, it means **[0..j) = (i-j..i]** for all j = x, p[x], p[p[x]] and so on.

→ Reply

17 months ago, # ^ | +8

Burunduk1

Thanks. Fixed.

→ Reply

3 weeks ago, # ^ | 0

shubham1100

KMP requires only the smaller string to be known to allow for computing the prefix function. It is a stream based algorithm,you can work even if characters are given you one by one for the main string. This may be an advantage in certain scenarios.

→ Reply

4 years ago, # | +1

vlade087

Test your skill using Z algorithm http://www.spoj.pl/problems/QUERYSTR . Good Luck

→ Reply

4 years ago, # | ← Rev. 2 0

Adrian2010

Very good algorithm

→ Reply

4 years ago, # | 0

simp1eton

Wondering, is there any reason why the z-algorithm is not as famous as KMP? This is the first time I heard about it XD. It seems that z-algorithm is easier to understand.

→ Reply

4 years ago, # | 0

Sorry to revive this old topic, but I have a question. Let's say that i < R and z[i-L] > R-i+1. Wouldn't this mean that z[i] = R-i+1? Here is my reasoning.

s[R+1] != s[R-L+1], because if they were equal then z[L] would be greater than R-L+1.

In order for z[i] to be greater than R-i+1, s[R+1] must equal s[i-L+R-i+1].

This can be rewritten as s[R+1] must equal s[R-L+1], but it doesn't because

aquamongoose

This can be rewritten as s[R'+1] must equal s[R'-L'+1], but it doesn't because of the first statement.

Therefore, assuming i < R, if z[i-L] != R-L+1, then it equals min(z[i-L],R-i+1). Otherwise, it is still necessary to loop and update R and L values.

Is this true?

→ Reply

paladin8

4 years ago, # ^ | 0

That argument seems to work. Doesn't change the complexity though :)

→ Reply

brunoja

3 years ago, # ^ | ← Rev. 5 0

Hi, I think that assumption is wrong, take this test case:

```
string input: ddcdddc
z array     : 0102310
your assumpt: 0102110
```

Is it right or I misunderstood what he said? :o

(sorry for editing, I am new to Markdown :p)

→ Reply

forthright48

3 years ago, # | +4

Amazing tutorial. Thanks to you and e-maxx I have learned a new algorithm today :D

→ Reply

TarifEzaz

3 years ago, # | +3

Thanks for this amazing tutorial. Keep writing! :)

→ Reply

piyush006

3 years ago, # | 0

Very nice explanation of the algorithm , thanks !

→ Reply

Vivekscripts

3 years ago, # | 0

There seems to be one more condition missing in the routine, actually not missing but Algo is doing more work for that case. So, there can be three cases when I < R and there are as follows

1) Z[k] < R-I+1 in this case Z[I] = Z[k] //Z[k] Length will be strictly less than R-I+1 2) Z[K] > R-I+1 in this case Z[I] = R-I+1 //This is the missing case. 3) Z[k] = R-I+1 in this case Z[I] = R-I +1 + [keep computing the Z values starting from the position R] // This is because we know Z[K] is strictly = R-I+1 (Beta) but Z[I] can still match with Patterns next character thus start computing the Z values for this Z[I] starting position from R until the mismatch is found.

→ Reply

saadtaame

17 months ago, # | -8

One of my favorite posts. Quick question: the algorithm seems to ignore `z[0]`; shouldn't it be the case that `z[0] = n`? Thanks.

→ Reply

Prestige
17 months ago, # ^ | 0
By the algorithm we consider that Z[0] is underfined
→ Reply

shamir0xe
17 months ago, # ^ | 0
you can add it end of your code, if necessary :)
→ Reply


akeilwilliams
14 months ago, # ^ | 0
So, an algorithm is s type of code?
→ Reply

Colonelmo
5 months ago, # ^ | 0
ghabool dari nemikhaym z[0] ?
→ Reply


Pheonix_0
17 months ago, # | -9
can you please elaborate on what S[i] stores .
→ Reply

saadtaame
17 months ago, # ^ | +5
`S` is the string. `S[i]` is the symbol at position `i`. If `S = "abcd"`, then `S[0] = 'a'`, `S[1] = 'b'` and so on.
→ Reply


akeilwilliams
14 months ago, # | 0
So where do we Type out an algorithm? C++?
→ Reply

Everybody_Lies
9 months ago, # | -8
Is it possible to use this algorithm to solve UVA 10298 problem. If yes, would you please describe the process?
→ Reply

saadtaame
9 months ago, # | 0
Just a silly optimization. In the line `while (R < n && s[R-L] == s[R]) R++;`, isn't the `R < n` redundant for strings? Because `s[R-L] == s[R]` will be false when we hit the `\0` character. This could also work if we are using a sequence of numbers instead of strings (we have to add a number at the end that is different than all others).
→ Reply

dalex
9 months ago, # ^ | 0
Not everyone uses C++ char arrays to store strings
→ Reply

9 months ago, # ^ | 0
that's the point. I'm saying that you can use sequences of

saadtaame

that's the point. I'm saying that you can use sequences of any type provided that you insert a sentinel element at the end of your sequence. Then you can save the `R < n`.

→ Reply

AlexDmitriev

9 months ago, # ^ | 0

afair, std::string allows s[s.size()] and returns 0

→ Reply

beatoriche

2 months ago, # ^ | 0

it's guaranteed only for C++11 or later

→ Reply

Decrypto

6 months ago, # | +3

Guys, this link at youtube might come in handy for a deeper insight and intuition into the algorithm. https://www.youtube.com/watch?v=MFK0WYeVEag

→ Reply

eagle93

6 months ago, # | 0

Thank you for this awesome tutorial.

this link contains animation for Z-Algorithm, it may be helpful.

→ Reply

Colorless_coder

5 months ago, # ^ | 0

awesome animation..... Thank you **eagle93**

→ Reply

unrealsoul007

2 months ago, # ^ | 0

Thanks for the link! :)

→ Reply

sierra101

4 months ago, # | +3

can any one tell me how to solve Problem B of Beta Round 93 with KMP by building Longest common prefix/suffix array :D

→ Reply

ducanhvn

4 months ago, # | ← Rev. 5 0

my code for make z function :)

```
void make_z()
{
  for (int i = 1, l = 0, r = 0, _new; i < s.size(); i++) {
    if (i <= r) z[i] = min (r - i + 1, z[i - l]);
    while ((_new = i + z[i]) < n && s[z[i]] == s[_new])
z[i]++;
    _new--;
    if (_new > r) l = i, r = _new;
    }
}
```

→ Reply

kipawa

3 months ago, # | 0

If you want a much more detailed explanation with examples and complexity analysis, this link will be quite helpful!

→ Reply

2 months ago, # | +9

i_am_not_real

Now we are computing the value at `i = k`. The values L and R are as shown in the image. Now in the case `Z[k] ≥ R - i + 1`, since `r + 1` and `r' + 1` values are not same(if they were to be same, the interval would have been `L` and `R + 1` instead of the current values), why do we need to check the values after `r + 1`?

→ Reply

dipteshlce3

2 months ago, # ^ | 0

Helpful.

→ Reply

i_am_not_real

3 weeks ago, # ^ | 0

Can anyone answer my question above?

→ Reply

yashv

5 weeks ago, # | 0

Can the longest sub-string and prefix overlap?

→ Reply

yiruma_ludovico_moyeen

15 hours ago, # | 0

The O(n) complexity made me interested. Such a beautiful algorithm. Thank you :)

→ Reply