jBoxer

I change the directions of small pieces of metal for a living.

The Knuth-Morris-Pratt Algorithm in my own words

Dec 13th, 2009 | Comments

For the past few days, I've been reading various explanations of the Knuth-Morris-Pratt string searching algorithms. For some reason, none of the explanations were doing it for me. I kept banging my head against a brick wall once I started reading "the prefix of the suffix of the prefix of the…".

Finally, after reading the same paragraph of CLRS over and over for about 30 minutes, I decided to sit down, do a bunch of examples, and diagram them out. I now understand the algorithm, and can explain it. For those who think like me, here it is in my own words. As a side note, I'm not going to explain why it's more efficient than na"ive string matching; that's explained perfectly well in a multitude of places. I'm going to explain exactly how it works, as my brain understands it.

The Partial Match Table

The key to KMP, of course, is the partial match table. The main obstacle between me and understanding KMP was the fact that I didn't quite fully grasp what the values in the partial match table really meant. I will now try to explain them in the simplest words possible.

Here's the partial match table for the pattern "abababca":

```
1 char:  | a | b | a | b | a | b | c | a |
2 index: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
3 value: | 0 | 0 | 1 | 2 | 3 | 4 | 0 | 1 |
```

If I have an eight-character pattern (let's say "abababca" for the duration of this example), my partial match table will have eight cells. If I'm looking at the eighth and last cell in the table, I'm interested in the entire pattern ("abababca"). If I'm looking at the seventh cell in the table, I'm only interested in the first seven characters in the pattern ("ababab"); the eighth one ("a") is irrelevant, and can go fall off a building or something. If I'm looking at the sixth cell of the in the table… you get the idea. Notice that I haven't talked about what each cell *means* yet, but just what it's referring to.

Now, in order to talk about the meaning, we need to know about **proper prefixes** and **proper suffixes**.

Proper prefix: All the characters in a string, with one or more cut off the end. “S”, “Sn”, “Sna”, and “Snap” are all the proper prefixes of “Snape”.

Proper suffix: All the characters in a string, with one or more cut off the beginning. “agrid”, “grid”, “rid”, “id”, and “d” are all proper suffixes of “Hagrid”.

With this in mind, I can now give the one-sentence meaning of the values in the partial match table:

The length of the longest proper prefix in the (sub)pattern that matches a proper suffix in the same (sub)pattern.

Let’s examine what I mean by that. Say we’re looking in the third cell. As you’ll remember from above, this means we’re only interested in the first three characters (“aba”). In “aba”, there are two proper prefixes (“a” and “ab”) and two proper suffixes (“a” and “ba”). The proper prefix “ab” does not match either of the two proper suffixes. However, the proper prefix “a” matches the proper suffix “a”. Thus, **the length of the longest proper prefix that matches a proper suffix**, in this case, is 1.

Let’s try it for cell four. Here, we’re interested in the first four characters (“abab”). We have three proper prefixes (“a”, “ab”, and “aba”) and three proper suffixes (“b”, “ab”, and “bab”). This time, “ab” is in both, and is two characters long, so cell four gets value 2.

Just because it’s an interesting example, let’s also try it for cell five, which concerns “ababa”. We have four proper prefixes (“a”, “ab”, “aba”, and “abab”) and four proper suffixes (“a”, “ba”, “aba”, and “baba”). Now, we have two matches: “a” and “aba” are both proper prefixes and proper suffixes. Since “aba” is longer than “a”, it wins, and cell five gets value 3.

Let’s skip ahead to cell seven (the second-to-last cell), which is concerned with the pattern “abababc”. Even without enumerating all the proper prefixes and suffixes, it should be obvious that there aren’t going to be any matches; all the suffixes will end with the letter “c”, and none of the prefixes will. Since there are no matches, cell seven gets 0.

Finally, let’s look at cell eight, which is concerned with the entire pattern (“abababca”). Since they both start and end with “a”, we know the value will be at least 1. However, that’s where it ends; at lengths two and up, all the suffixes contain a c, while only the last prefix (“abababc”) does. This seven-character prefix does not match the seven-character suffix (“bababca”), so cell eight gets 1.

How to use the Partial Match Table

We can use the values in the partial match table to skip ahead (rather than redoing unnecessary old comparisons) when we find partial matches. The formula works like this:

*If a partial match of length **partial_match_length** is found and `table[partial_match_length] > 1`, we may skip ahead `partial_match_length - table[partial_match_length - 1]` characters.*

Let’s say we’re matching the pattern “abababca” against the text “bacbababaabcbab”. Here’s our partial match table again for easy reference:

```
1 char:  | a | b | a | b | a | b | c | a |
2 index: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
3 value: | 0 | 0 | 1 | 2 | 3 | 4 | 0 | 1 |
```

The first time we get a partial match is here:

```
1 bacbababaabcbab
2  |
3  abababca
```

This is a partial_match_length of 1. The value at `table[partial_match_length - 1]` (or `table[0]`) is 0, so we don't get to skip ahead any. The next partial match we get is here:

```
1 bacbababaabcbab
2     |||||
3     abababca
```

This is a partial_match_length of 5. The value at `table[partial_match_length - 1]` (or `table[4]`) is 3. That means we get to skip ahead `partial_match_length - table[partial_match_length - 1]` (or `5 - table[4]` or `5 - 3` or 2) characters:

```
1 // x denotes a skip
2
3 bacbababaabcbab
4     xx|||
5       abababca
```

This is a partial_match_length of 3. The value at `table[partial_match_length - 1]` (or `table[2]`) is 1. That means we get to skip ahead `partial_match_length - table[partial_match_length - 1]` (or `3 - table[2]` or `3 - 1` or 2) characters:

```
1 // x denotes a skip
2
3 bacbababaabcbab
4       xx|
5         abababca
```

At this point, our pattern is longer than the remaining characters in the text, so we know there's no match.

Conclusion

So there you have it. Like I promised before, it's no exhaustive explanation or formal proof of KMP; it's a walk through my brain, with the parts I found confusing spelled out in extreme detail. If you have any questions or notice something I messed up, please leave a comment; maybe we'll all learn something.

Posted by Jake Boxer Dec 13th, 2009

Comments

68 comments

★ 14

Join the discussion…

Best Community Share Login

Richard Fleming • 10 months ago

Near the start of the "How to use the partial match table" section when you say you can skip ahead if "table[partial_match_length] > 1" do u not mean "table[partial_match_length - 1] > 1"?

10 ^ v • Reply • Share ›

ashoka ➔ Richard Fleming • 6 months ago

shouldn't it be >= (greater than or equal to), instead of > (greater than) ?

^ v • Reply • Share ›

KMP ➔ ashoka • a month ago

NOOO!!!! IT SHOULDN'T.. YOU STUPID BITCH.. YOU WHORE DIDN'T LEARNT A THING. WHY THE FUCK DO YOU VISIT SITES LIKE THIS? YOU MOM WAS WHORE SINCE SHE WAS 8 YEARS OLD AND YOUR FATHER FUCKS WITH ANYONE HE BUMPS INTO..

BITCH..

^ v • Reply • Share ›

t ➔ KMP • 6 days ago

wtf :D

1 ^ v • Reply • Share ›

jfk ➔ KMP • 25 days ago

ok thanks

1 ^ v • Reply • Share ›

Jowny Cawsh • 6 months ago

Awesome job man, the Wikipedia article was so verbose and dry I couldn't wrap my head around the concepts. It'd be amazing if you did the same for the Boyer-Moore algorithm and it's good suffix rule :P

6 ^ v • Reply • Share ›

Ruobin Ling • 6 months ago

This saved my butt for my Data Structures test tomorrow! Thanks a lot man!

P.S. I think Richard Fleming is right though, it should be table[partial_match_length-1], from your work it's pretty clear that that is a typo :)

Thanks again!

2 ^ v • Reply • Share ›

KMP • a month ago

To mi kazi

1 ^ v • Reply • Share ›

Dan • 5 months ago

That was a life-saver, thank you very much.
Indeed - there's a mistake - it should say "table[partial_match_length - 1] > 1", but other than that - the article was very clear.

1 ^ v • Reply • Share ›

Guest • 6 months ago

You have no idea how obscure the notion of proper prefix and proper suffix felt looking at algorithms and nebulous variable names.

I thank you from the bottom of my heart.

1 ^ v • Reply • Share ›

Subhra • 6 months ago

Thanks a lot!!!! Finally I understood KMP after browsing through so many sites and so many videos.

1 ^ v • Reply • Share ›

Neha Jatav • 11 months ago

Great! We share the feeling (about banging my head against a brick wall once I started reading “the prefix of the suffix of the prefix of the…”)
But thanks to your blog post, its crystal clear :)

1 ^ v • Reply • Share ›

Madhur Sharma • a year ago

http://en.wikipedia.org/wiki/K...

I was checking the Wiki and couldn't understand the below condition while creating table building. Can you explain?

(second case: it doesn't, but we can fall back)
otherwise, if cnd > 0, let cnd ← T[cnd]

1 ^ v • Reply • Share ›

Arun Nadesh • 2 years ago

//If a partial match of length partial_match_length is found and table[partial_match_length] > 1, we may skip ahead ...//

Shouldn't it be " table[partial_match_length - 1] > 1 " ?

1 • Reply • Share ›

Jake Boxer Mod ➦ Arun Nadesh • 2 years ago
Correct! Great find. Updating now.

• Reply • Share ›

Abhiroop Sarkar • 25 days ago
Like Einstein said "If you can't explain it simply, you don't understand it well enough" Brilliant explanation. Thanks a lot

• Reply • Share ›

Garvit Jain • a month ago
If only I had words to thank you! 2 days and I was at level 0. One article and I feel like the king :P

• Reply • Share ›

Fatima Asif • a month ago
Is there any difference between finding of sp'i of KMP and sp'i of optimized KMP?

• Reply • Share ›

geekgirl • a month ago
really the best kmp explaination till now :) thanxs :) helps a lot

• Reply • Share ›

SHRESHTHA DEEPAK • 2 months ago
can u please explain the time complexy for the worst case?

• Reply • Share ›

Steve • 2 months ago
Freakin Awesome man.. thnx!!

• Reply • Share ›

Geek4IT • 2 months ago
Nice bro.

• Reply • Share ›

Hoang Nguyen • 2 months ago
you sir are a hero

• Reply • Share ›

rock marciano • 2 months ago
Best explanation available. Thanks a lot man.
• Reply • Share ›

Vijay Kumar • 3 months ago
Thank you very much, Awesome explanation!
• Reply • Share ›

aaaa • 3 months ago
great bro....thanks a ton....
• Reply • Share ›

Sumit Poddar • 3 months ago
Great work dude. Thanks a lot. It really helps a lot in understanding the algorithm. ***** to your effort :)
• Reply • Share ›

Ajay • 3 months ago
Thanks a lot for this great explanation!!!
• Reply • Share ›

xuan wang • 3 months ago
Bravo ！ thx man ！
• Reply • Share ›

Karlvin • 5 months ago
Really nice work man! But I'm still curious about finding out the match table in a sufficient method.
• Reply • Share ›

Ayushi • 5 months ago
Thank You so much for this awesome explanation!
• Reply • Share ›

kallol • 5 months ago
thanks a lot dear ... I was also washed away by several verbose tutorials ... u saved me...
• Reply • Share ›

Saiyam Agarwal • 5 months ago
Nice article !!!
• Reply • Share ›

LUISMO • 6 months ago
Ohh bro I finally understad what the Partial Match Table is meant to, I've read tons of papers

full of "prefix of the suffix of the multi-prefix of the anti-sufix" as you mentioned... and now I finally get it thanks to you, this should be on TopCoder Algorithms' Tutorials
Forgive my bad english THANKS a LOT

• Reply • Share ›

Ramakant Bharti • 6 months ago
We can simplify it even more by having the partial match array indexing start from 1 (for easier understanding), that way we don't have to worry about "-1". Now it becomes:

if partial match(of length m), then
if(table[m] > 1)
jump (m-table[m]) in the main text & continue searching

• Reply • Share ›

Rikimazu • 7 months ago
I have a question that in the case , as the dismatch founded here:

ababaa ababab,if we find 'a' dismatch 'b'(index = 5), so we compare 'a'(in ababaa) with 'b'(index = 3).why?we already know it dismatches.because in abababca's partial match table, they are the same character between index 3 and index 5.

thank you.it's really hard for me to understand why.

• Reply • Share ›

Mike Rosoft • 7 months ago
Hey, I finally understand it, thank you!

• Reply • Share ›

Kasper Ziemianek • 7 months ago
Good stuff here

• Reply • Share ›

vishwas garg • 7 months ago
partial_match_length - table[partial_match_length - 1] what is the reason behind doing this

• Reply • Share ›

Kasper Ziemianek ➚ vishwas garg • 7 months ago
I think we are doing this to compute how many characters we can skip

• Reply • Share ›

vishwas garg ➚ Kasper Ziemianek • 7 months ago
that i know.....i got the reason

• Reply • Share ›

Nilerafter24 • 8 months ago

I have a question.
In the partial match table, why is value[0]=0 and value[1]=0 ?
Let's say you're interested in the first letter of the pattern, 'a'... 'a' is a proper prefix of 'a' and it is also a proper suffix of 'a' ..Doesn't this mean that value[0]=1 ?
And for value[1]: 'ab' has prefixes('a' and 'ab') and suffixes('b' and 'ab'). So shouldn't value[1]=2 since the length of the longest prefix that matches a proper suffix, in this case 'ab' is 2?
i.e. value:| 1 | 2 | 1 | 2 | 3 | 4 | 0 | 1 |

Really good explanation by the way. I think I'm almost getting this damned algorithm at last.

• Reply • Share ›

Michael → Nilerafter24 • 8 months ago
Nope. A proper suffix and prefix cant be the string itself, just a substring! Thats why "a" itself, has no proper pre- and suffixes at all. Same with "ab", the only proper prefix is "a" and the only proper suffix is "b", which has no element in common, thats why it is 0.

3 • Reply • Share ›

Joseph Heng → Michael • 3 months ago
Whether I can understand it like this: if the string's suffix and prefix could contain itself, then the longest match length is always the substring itself, and obviously, it make no sense.

• Reply • Share ›

Nilerafter24 → Michael • 8 months ago
I see my mistake now. I forgot one should only consider sub-strings for proper prefixes and suffixes... I was thinking too much.
Thanks for clearing that up. I now completely understand this.
Best explanation on the web.

• Reply • Share ›

hello • 8 months ago
I love you.

• Reply • Share ›

Ken Karasawa • 9 months ago
okay. now THIS i can understand. holy crap, can't believe they didn't explain it like that. thank you!!!

now to figure out how to contruct that partial table using dynamic programming...

• Reply • Share ›

JianFeng • 9 months ago
My English is very bad but i still understand thanks you

• Reply • Share ›

zhengchao xu • 9 months ago

nice explain

• Reply • Share ›

Ram • 9 months ago

Recent Posts

- Why eating sushi is a stressful experience for me
- Converting to the new RSpec 2.11 expectation syntax
- Changing a Rails 3 project name
- BostInnovation's Seth Priebatsch Article is Childish Garbage
- Bash's PS1 Syntax: The Inspiration for Brainfuck?

Latest Tweets

- Status updating...

Follow @jake_boxer