

# Two-dimensional pattern matching/compression

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.theory/2006-07/msg00020.html>

---

- *From:* Herbert Glarner <[herbert.glarner@xxxxxxxxxx](mailto:herbert.glarner@xxxxxxxxxx)>
  - *Date:* Fri, 07 Jul 2006 20:09:44 +0200
- 

Good evening,

I am not quite sure if this is the right place to /discuss/ aspects of two-dimensional pattern matching. If not, I'd thankfully take pointers to a group which suits better.

-----

There exist  $q$  two-dimensional arrays  $A$  with an individual number of rows  $r$  and columns  $c$  each. Each array is a 2-dimensional pattern.

The largest "common area" of any array shall merge with any other array into a single new superarray such, that they become part of a superarray and could be extracted individually without any loss of information, as long as their start coordinates within the superarray and their dimensions are known. Obviously, if a pattern is not completely part of another one, this involves enlarging original array dimensions, possibly generating 'unused space'. The goal is to get a final superarray with a minimal overall area.

It is not a requirement, but a possibility, that each pattern is fed in one after the other without the actual one knowing about the future ones. However, it is also legal to assume that all patterns already exist such, that they all are individually accessible.

-----

Start of example for clarification. Needs fixed-width font to make sense.

As an example, consider the following patterns, consisting of an alphabet of only 2 symbols "x" and "-". The original arrays  $A$  can only consist of these alphabet symbols "x" and "-", but after merging patterns into a superarray, a metasymbol "." (unused) can become part of a compound superarray, indicating that a part of another pattern may occupy this space with its symbols 'x' and '-'.

(For sake of easy reference the patterns are given the shape of letters. They may just be of any content, though.)

Given are 4 arrays:

"P": "B": "m": "h":

```

XXXXXXXX- XXXXXX- X-XX--XX- X-----
X-----X X-----X XX--XX--X X-----
X-----X X-----X X---X---X X-XX-
X-----X X-----X X---X---X XX--X
XXXXXXXX- XXXXXX- X---X---X X---X

```

## Two-dimensional pattern matching/compression

```

X----- X----- X X---X---X X---X
X----- X----- X X---X---X X---X
X----- X----- X X---X
X----- XXXXXX- X---X

```

"B" and "P" do match at two of their opposite edges, "m" and "h" as well. They can be merged to a superarray "BP" and "hm", which can be merged further to "BPhm" with an overall size of 15x16:

"BP" "hm" "BPhm"

```

XXXXXXXX- X----- .... X----- .....
X-----X X----- .... X----- .....
X-----X X-XX--XX- X-XX--XX- .....
X-----X XX--XX--X XX--XX--XXXXXX-
XXXXXXXX- X---X---X X---X---X-----X
X-----X X---X---X X---X---X-----X
X-----X X---X---X X---X---X-----X
X-----X X---X---X X---X---XXXXXX-
XXXXXXXX- X---X---X X---X---X-----X
X----- .....X-----X
X----- .....X-----X
X----- .....XXXXXX-
X----- .....X-----
.....X-----
.....X-----
.....X-----

```

But also "P" and "m" could be merged, and "B" with "h". The two intermediate supermatrices don't have a common part they share, though (as far as I see \*g\*). This combination produces a final superarray, which is larger than the previous one, and thus should be discarded in favor of the former one.

"mP" "hB" "mPhB"

```

X-XX--XX-..... X-----..... X-XX--XX-.....
XX--XX--XXXXXX- X-----..... XX--XX--XXXXXX-
X---X---X-----X X-XX-..... X---X---X-----X
X---X---X-----X XX--XXXXXX- X---X---X-----X
X---X---X-----X X---X-----X X---X---X-----X
X---X---XXXXXX- X---X-----X X---X---XXXXXX-
X---X---X-----X---X-----X X---X---X-----X
.....X----- X---XXXXXX- X-----...X-----
.....X----- X---X-----X X-----...X-----
.....X----- ...X-----X X-XX-...X-----
...X-----X XX--XXXXXX-....
...XXXXXX- X---X-----X....
X---X-----X....
X---X-----X....
X---XXXXXX-....
X---X-----X....
...X-----X....
...X-----X....
...XXXXXX-....

```

## Two-dimensional pattern matching/compression

Note, that from any superarray all the composing patterns can be extracted 1:1, as long as their start coordinates and dimensions are known. This was assumed.

End of example.

---

Now, first of all I'd be interested, if anyone has knowledge of publications or ongoing research papers about this kind of matching/compression problems. Possibly there exists also a different terminology other than "Two-dimensional pattern matching" under which I could do a new Google search. Pointers would be appreciated.

Otherwise: Does anyone have an idea on how to solve especially the comparison problem? Brute-force obviously will take eternities, because of the multitude of required shift windows into two dimensions. Contrary to the example, the real input can consist of thousands of such arrays, and as a theoretical discussion I feel that we should not make any assumptions about the dimensions of these arrays.

I was thinking about translating all possible shift windows (subarrays) originating at edges (since matches can only occur at opposite edges, unless an array will be completely a part of the superarray) one after one into bitstreams and compare those against the so far final superarray's bitstreams of same size. However, that would require that after a match the superarray's bitstreams would need to be recalculated to a possibly large extent. Even if this would enable me to find the largest common area, it won't solve the positioning problem, though: intuitively I am quite sure, that with this approach I would miss out better (more compact) arrangements.

I look forward for thoughts and possibly a discussion about different aspects. Thanks for your patience.

Kind regards  
//Herbert Glarner

.