

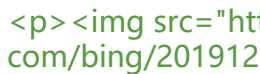

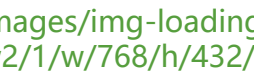
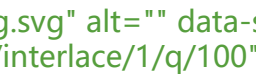


一个 Scheme 的矩阵转置

作者: [plus7wist](#)

原文链接: <https://ld246.com/article/1618714775792>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

matrix 是列表的列表，先假设每个列表都是等长的，例如它是 `'((1 2) (3 4) (5 6))`。把它视作一个二维矩阵，matrix 中每个元素自上而下对应矩阵的一行，而每个元素作为一个列表的元素，自左而右的表示矩阵行中的元素。所以上述的例子对应的矩阵是：

```
<div class="language-math">\begin{pmatrix}
  1 & 2 \\
  3 & 4 \\
  5 & 6 \\
\end{pmatrix}</div>
```

我们的程序需要转置这个矩阵，也就是变成 `'((1 3 5) (2 4 6))`。我完成了一个素的实现，它迭代了两层循环。但我发现了这样的实现：

```
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight-cl">(define (transpose matrix)
</span></span><span class="highlight-line"><span class="highlight-cl">  (if (null? matrix)
</span></span><span class="highlight-line"><span class="highlight-cl">      '())
</span></span><span class="highlight-line"><span class="highlight-cl">      (apply map (l
mbda x x) matrix)))
</span></span></code></pre>
```

首先要注意不要把 `(lambda x x)` 错读成 `(lambda (x) x)`，理解成 identity 函数。上文的 lambda 实际上是将参数列表作为列表返回，也就是说如果这个 lambda 的参数是 `1 2 3` 三个，那么函数返回 `(1 2 3)`。

再有，要记得 map 的签名：`(map p l1 ... ln)`，`l1` 到 `ln` 是等长的列表，长度是 m，那么 `p` 调用 m 次，每次调用有 `n` 个参数，第 i 次调用的参数是 `l1` 到 `ln` 的第 i 个参数，回值是上述 `m` 个返回值组成的列表。

如果 `p` 是返回参数列表的函数，`l1` 到 `ln` 是矩阵的 `n` 行，那么 `p` 调用矩阵的列数 (`m`) 次，每个返回值是第 i 列，整个返回值是列的列表，视作矩阵时即是输入矩阵的转置。

`(apply p a1 .. an '(b1 ... bn))` 相当于 `(p a1 ... an b1 ... bn)` 以上一段的 `map` 调用正是 `(apply map that-lambda matrix)` 最后写好空矩阵的处理即可。