به نام خدا

جابر بابکی 96131020

پروژه دوم

Multi-Field Range Encoding for Packet Classification in TCAM

بزرگ فکر کن ، هوشمندانه تصمیم بگیر، اما کوچک شروع کن

1396.11.5

مسیر پروژه:

جدول کل توابع کاربردی در انجام پروژه

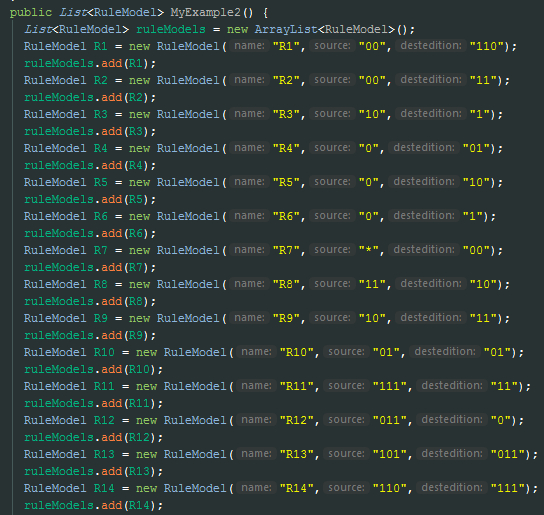
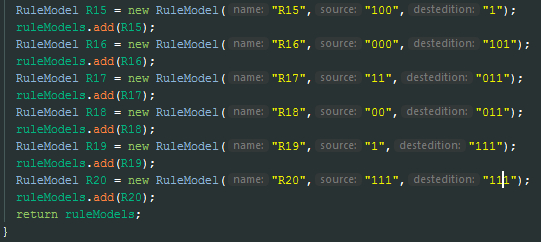
|  |  |  |
| --- | --- | --- |
| شماره | نام تابع | عملکرد |
| 1 | init() | در ابتدا ارایه دو بعدی که original 2-D مقدار دهی اولیه می شود. |
| 2 | setLimted() | برای ترسیم زئومتریک باید محدوده مناسب را با توجه بهrule ها بدست بیاوریم. |
| 3 | setRule() | در این تابع در واقع با توجه به محدوده تعیین شده rule ها و region ها بر روی original 2-D تعیین می شود. |
| 4 | setElemtryRegion() | این تابع با تحلیل روی original 2-D و با توجه به قوانین گفته شده ElemtryRegion را تعیین می کند. |
| 5 | getCodeWord() | این تابع با توجه به محدوده تعیین شده و original 2-D کد ها را تولید می کند |
| 6 | getVest() | برای تعیین terneray code باید ابتدا region را تعیین بکنیم |
| 7 | getTernary() | در این تابع با توجه به جدول codeWord ها و Vest باید ternary کد تولید شود |
| 8 | showOrginal2D() | این تابع جهت نمایش محتویات original 2-D است |
| 9 | showVest() | در تابع محتویات جدول Vest را نمایش می دهد |
| 10 | showCodeWord() | در تابع محتویات جدول CodeWord را نمایش می دهد |
| 11 | showTernary () | در تابع محتویات جدول Ternary را نمایش می دهد |

جدول کلاس مدل

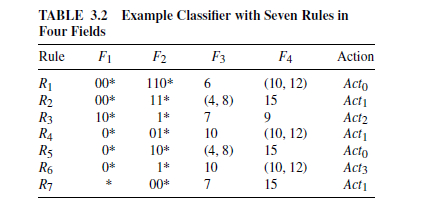
|  |  |  |
| --- | --- | --- |
| شماره | نام مدل | عملکرد |
| 1 | RuleModel | این مدل در واقع برای ساخت classifier و rule ها استفاده می شود. |
| 2 | LimitedModel | محدوده های بدست آمده برای هر rule در این کلاس مدل نگه داری می شود. |
| 3 | VestModel | دیتا مدلی برای نگه داری Vest ها می باشد |
| 4 | CodeWordModel | دیتا مدلی برای نگه داری CodeWord های بدست آمده می باشد. |
| 5 | TernaryModel | جدولی برای نگه داری Ternary های بدست امده می باشد. |

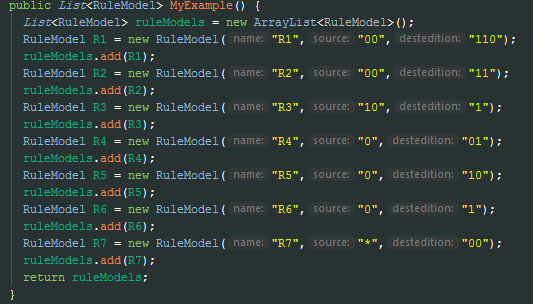
توضیحات و عملکرد هر تابع

1. ایجاد جدول original 2-D

به طور کلی هدف از این مقاله کم کردن حجم استفاده از حافظه TCAM با استفاده از کوتاه کردن داده های ذخیره شده در TCAM است . در مثال های مقاله از هیچ classifier استفاده نکرده و با فرض اینکه Rule ها و ماتریس ابتدایی را داریم شروع به انجام عملیات می کند، اما برای پیاده سازی نیاز است ابتدا جدول classifier برای خود مثال بزنیم و براساس آن ماتریس و ناحیه را تشکیل داد بر اساس چیزی که در صورت سوال هست گفته شده یک مجموعه قوانین 20 سطری ایجاد شود که من در کلاس ExampleClassifier این را ایجاد کردم :

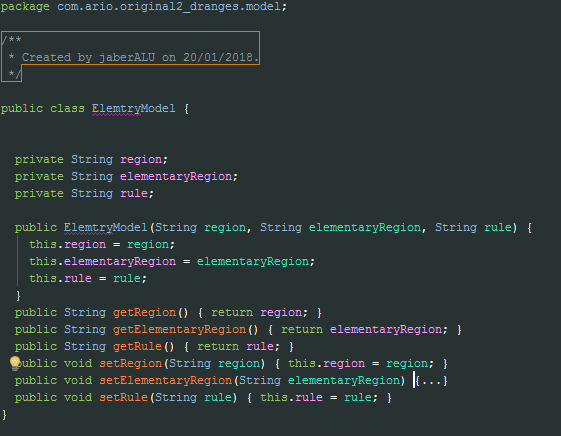
همچنین برای تست کردن و نشان دادن عملکرد صحیح برنامه از classifier که در کتاب High Performance Switches and Routers-1 صفحه ی 80 استفاده نکردم چون بر اساس این classifier ماتریس و ناحیه بندی را انجام می دهد و میتوان درستی اجرا را مقایسه کرد و در ادامه با استفاده از الگوریتم های مقاله پیش می رویم.

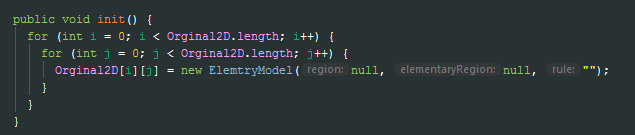


و فایل این clssaifier در کلاس ExampleClassifier قرار داده شده است.

پس برای تست از این Classifire استفاده می کنیم، هر چند با هر ورودی دیگری عملیات انجام درسیت و کامل انجام می شود.

برای تشکیل original 2-D که در مقاله نام ماتریس های هست که rule های دو فیلد در نظر گرفته شده ابتدا کلاس دیتا مدل برای هر درایه این ماتریس را بررسی می کنیم :

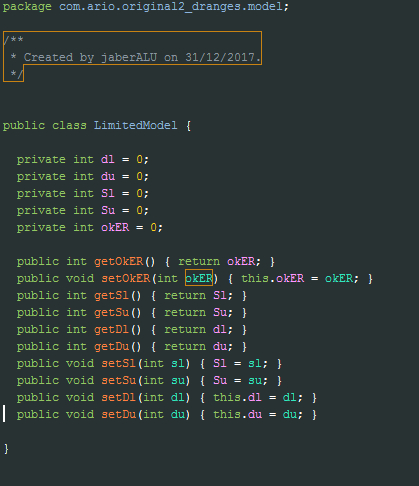


همانطور که دیده می شود هذ درایه حاوی region و elemrntraryRegion و rule می باشد، و در تابع زیر مقدار دهی اولیه می شوند

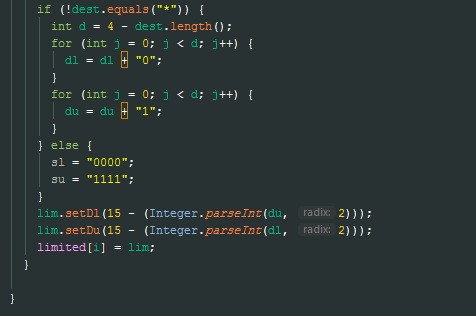
همانزور که دیده می شود در یک آرایه دو بعدی ما می آییم کلاس مدل ElemtrayModel را new می کنیم . و تمامی عملیاتی که در مقاله گفته شده بر روی این ماتریس انجام می شود.

1. نحوه تعیین محدوده بر اساس Rule ها

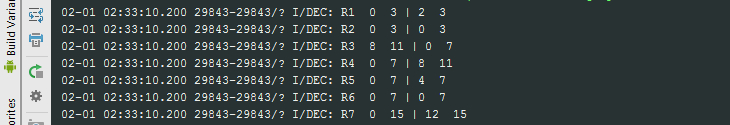
در این تابع باید مشخص شود چه جاهایی باید rule و region و elementray ست شود ابتدا دیتا مدل این استفاده شده در این تابع را بررسی می کنیم :



برای تعیین محدود ما از این دیتا مدل استفاده می کنیم برای soure یه حد پایین در نظر گرفتم و همچنین برای destnetion هم یه حد پایین و بالا در نظر گرفتم.

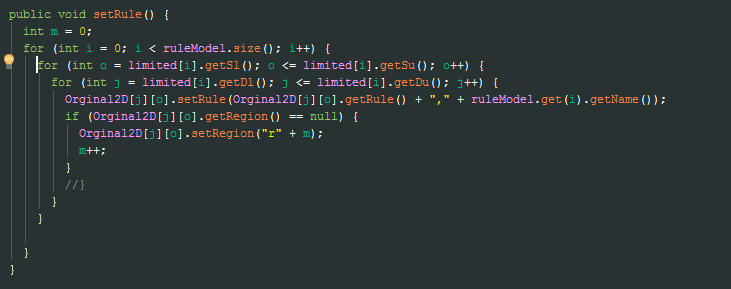


همانطور که دیده می شود ما ابتدا source را و destenition را از example که ساختیم می خوانیم و سپس source هر چند بیت که هست از 4 بیت کم میکنیم (می تونیم بر اساس بیشترین بیت میدا یا مقصد این 4 بیت متغییر باشد ) سپس به اندازه های بیت های باقی مانده ما یکبار صفر اضافه می کنیم و یک بار یک تا حد بالا و پایین بدست بیاید و همین کار را برای destnition انجام می دهیم، من برای اینکه ارایه ی دوبعدی را در سیستم مختصات دکارتی ببرم باید منهای 15 میکردم و در نهایت این محدوده های را که خیلی مهم هستند در ارایه به نام limited ذخیره می کنم.

خروجی این قسمت به صورت زیر است .

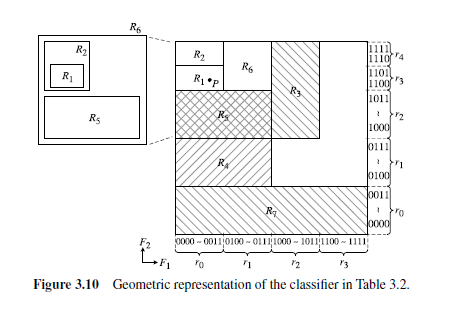
همانطور که دیده می شود در واقع ما مبدا و مقصد را به صورت دسیمال و حد بالا و پایین نشان دادیم که برای مراحل بعدی و ستن کردن rule ها مهم هستند.

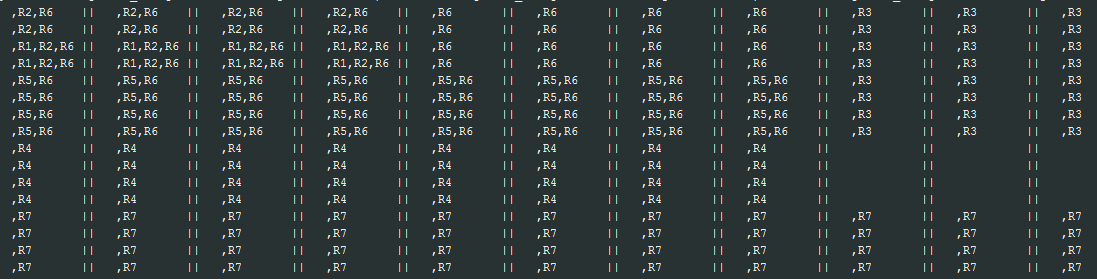
1. ست کردن Rule ها و Region ها در original 2-D

در این مرحله باید ما باید rule ها را و region ها را روی ماتریس ست بکنیم:

همانطور که دیده می شود ما بر اساس محدودی ای که تعیین کردیم می آییم rule را ست می کنیم و region ها رو هم ست می کنیم .

خروجی به صورت زیر می باشد:

قبلا از اینکه خروجی این تابع را ببینیم ابتدا ناحیه بندی که در کتاب High Performance Switches and Routers-1 و در صفحه ی 90 هست را مشاهده می کنیم تا بهتر بتوانیم مقایسه بکنیم

و اما خروجی تابع من :

R3

R7

R4

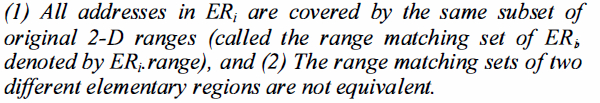
R6

R2

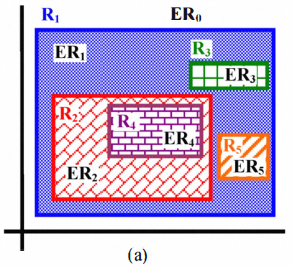
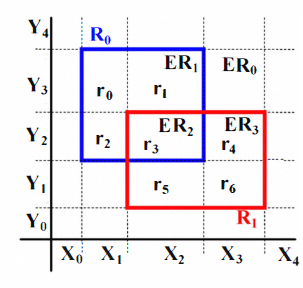
R1

همانطور که دیده می شود کاملا شبیه زئو متریک هست که در کتاب هست. نکته مهم اینکه region ها همانطور که در تابع بود تعیین شدند اما جهت شلوغ نشدن نشان ندادم .

1. تعیین Elementary region

بر اساس مواردی که در مقاله گفته شده است، و دو مثالی که در مقاله نشان داده شده است ما می آییم تابع را اجرا میکنیم، در مقاله برای تعیین Elementary region به موارد زیر اشاره کرد.

و بر اساس دو مثال گفته شده که به شکل زیر است :

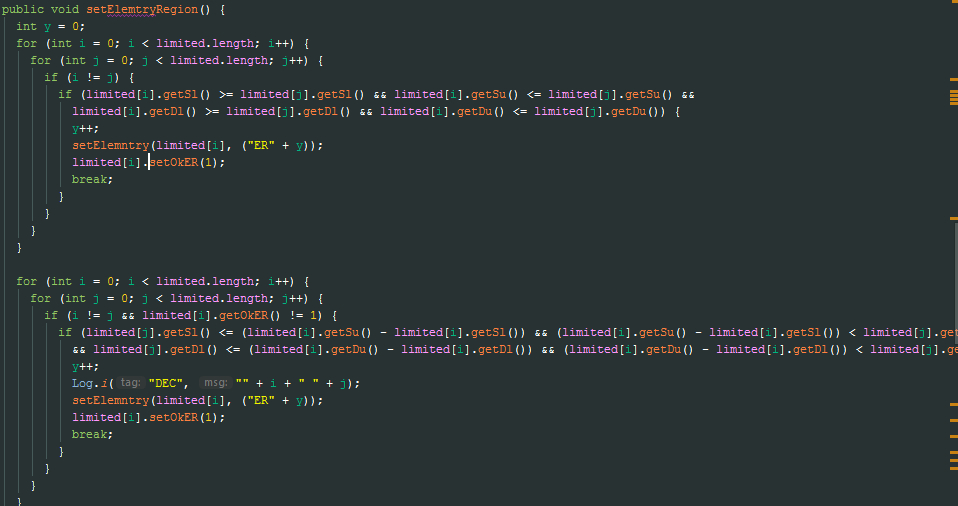


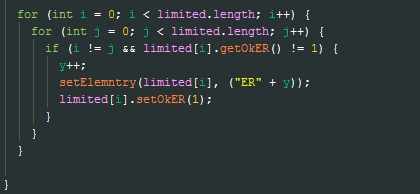
نتیجه می گیرم که تابع تعیین کننده ER باید سه مرحله زیر را بررسی و برچسب ER را ست کند

اول اینکه اگر rule زیر مجموعه rule دیگر بوده باید region های rule داخلی برچسب غیر از بیرونی داشته باشند

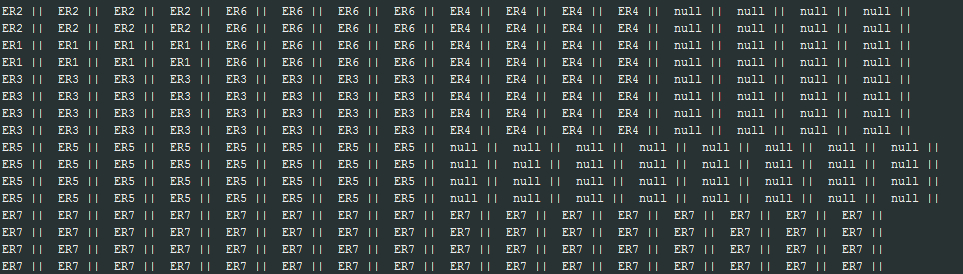
دوم اینکه اگر دو تا rule با هم رابطه جزیی داشته باشند باید اون region ها برچسب ER جدید بخورند

سوم اینکه بقیه rule که با هم جدا هستند باید ER بخورند و همچنین باقی مانده elemtery ها باید ER صفر بخوردند

با توجه به این قوانین تابع زیر عمل می کند:

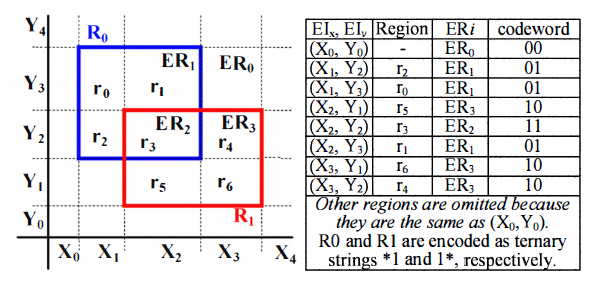


در این تابع سه تا for اصلی قرار دارد که هرکدوم موارد و قوانین گفته شده را اجرا می کنند، من این سه تا قوانین را با تحلیل روی محدوده هر rule این کار را انجام می دهیم یعنی بررسی میکنیم اگر حد پایین یک rule از حد پایین rule دیگه ای بزرگتر بوده و حد بالاش کوچکتر بوده و همینوطور برای مقصد هم چک می کنیم و نتیجه میگیریم زیر مجموعه هست و فیلد setOkER را یک میکنیم و تا برای قانون بعد چک نشود .

خروجی این مرحله به صورت زیر است :

همانطور که دیده می شود به دلیل زیر مجموعه بودن بخشی از Rule ها عملکرد تابع درست بوده و برای قسمت هایی که هیچ ارتباطی با هم نداشته برچسب متفاوت زده شده است توجه شود که شماره ER به صورت تصادفی ست می شود.

1. تعیین CodeWord

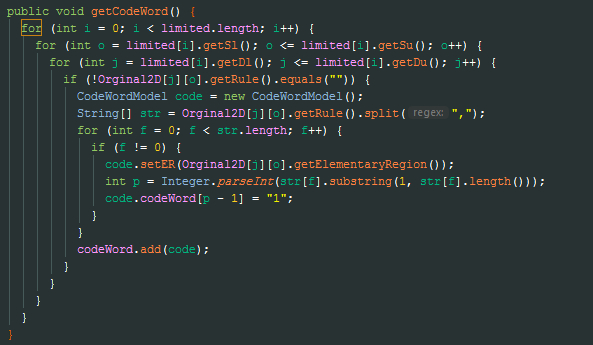
برای تعیین codeword بر اساس مثال اولی که زده شده عمل میکنیم

همانطور که در تصویر دیده می شود برای دو تا rule دو بیت در نظر گرفته شده است و در هر مختصات اگر آن rule وجو داشت برای آن 1 می گذارد و اگر وجود نداشت 0 میگذارد.

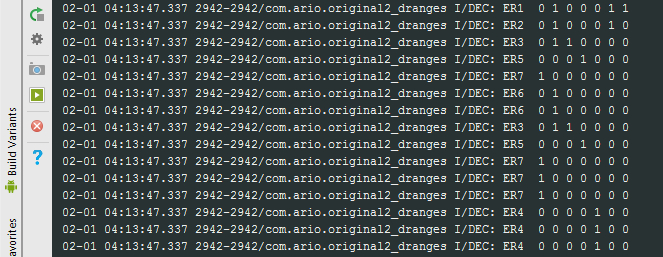
در تابع زیر نیز با توجه به محدوده هایی که داریم که و region هایی که داریم که در مجموع 20 تا هست (البته بعضی region ها ruel وجود ندارد که در نهایت 15 تا می باشد) codeWord را با تابع زیر ایجاد می کنیم.

ابتدا باید دیتا مدل codeWord را بررسی بکنیم :

در این دیتا مدل codeWord و برچسب ER ذخیره می شود و همانطور که دیده می‌شود ابتدا همه rule ها مقدار صفر دارند

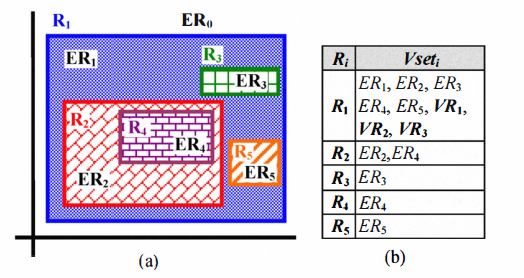


با استفاده از این تابع می آییم codeWord ها را تعیین می کنیم با استفاده از محدوده ها بر روی ناحیه ها بررسی می کنیم که آیا در اون ناحیه آیا rule قرار دارد یا نه اگر قرار دارد، اگر قرار دارد برای آن 1 قرار داده می شود در غیر اینصورت که بطور پیش فرض صفر است .

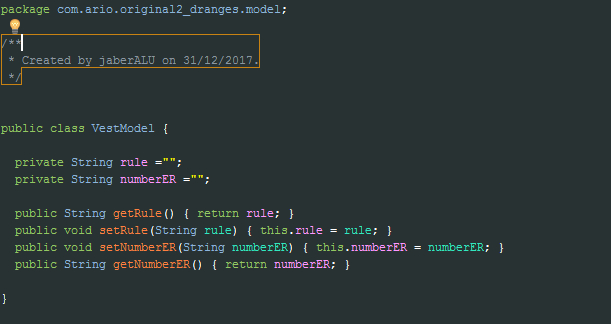
خروجی به صورت زیر است :

البته به طور پیش فرض ER صفر شامل 0000000 می باشد که من در ارایه ذخیره نکردم.

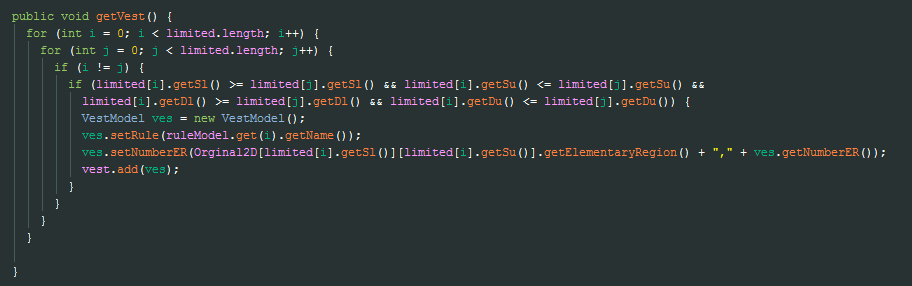
1. تابع Vest مورد نیاز برای تعیین Ternary

با توجه به شکل زیر در تابع vest زیر مجموعه ها را تعیین می کنیم و در جدولی قرار می دهیم .

همانطور که دیده می شود در هر rule بررسی می کند چند تا ER وجود دارد

ابتدا دیتا مدل مربوط به Vest را بررسی می کنیم :

همانطور که دیده می شود یک rule و ER ها را نگه می دارد.

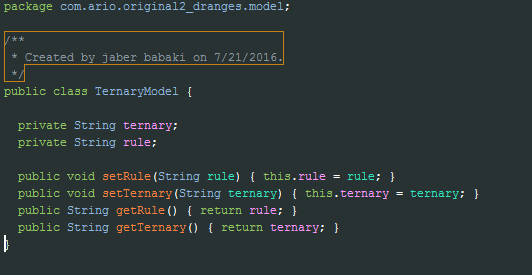


در تابع getVest ما می آییم ER هایی که زیر مجموعه Rule هستند را بررسی می کنم و یک نمونه vest مدل میسازم و rule و تمام اون ER ها را درون ان می ذارم و در ارایه داینامیک global که vest نام دارد قرار می دهم .

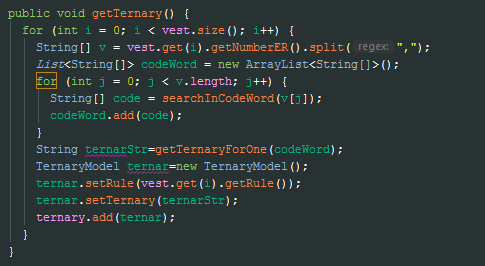
خروجی تابع به صورت زیر است :

1. خروجی نهایی TCAM

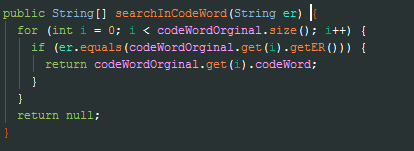
کل مقاله و تمام الگوریتم های برای رسیدن به TCAM که حجمش کمی کمتر شده، برای ایجاد این TCAM ما نیاز به دو جدول CodeWord و Vest داشتیم که بالا تر ایجاد کردیم حالا با استفاده از vest اون ER هایی که در یک مجموع هستند را باید کوتاه بکنیم به این صورت بیت که در اون مجموع یکسان هست را دقیقا همان می نویسیم و بیت متغییر را \* می گذاریم.

ابتدا دیتا مدل استفاده شده در این تابع را بررسی می کنیم

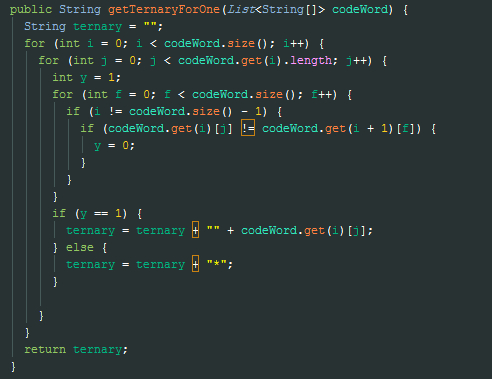
این دیتا مدل دو مقدار را نگه داری میکند کد کوچک شده که ternary نامیده می شود و rule مورد نظر

با استفاده از این تابع این جدول ایجاد می شود:

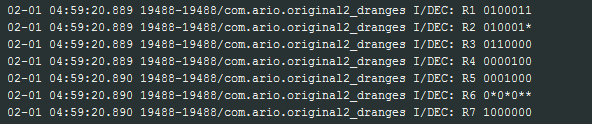
در این تابع ایتدا از هر vest خوانده می شود سپس بر اساس ER ارایه ای تشکیل می شود و در یک for با توجه به آن ER از جدول codeWord و با استفاده از تابع searchInCodeWord می ایم codeWord را میاریم بیرون و درون یک رایه می ریزیم برای تحلیل شباهت، سپس با استفاده از تابع getTernaryForOne می آییم اون codeWord هایی مربوط به یک ER را بررسی ternary را خروجی میدهد سپس دیتا مدل نمونه ای می سازیم و این مقدار های بدست آمده به همراه rule مورد نظر را درون ارایه داینامیک به نام ternary میریزیم

تابع searchInCodeWord

در این تابع میاد بر اساس ER سرچ میزند و اولین codeWord تطبیق داده شده را برمیگرداند

تابع getTernaryForOne

در این تابع سه حلقه for برای بررسی بیت های codeWord استفاده می شود ابتدا براساس یک بیت کل اون codeWord ها بررسی میشود و چناچه یک بیت در هر یک از بیت ها فرق داشته باشد y برابر صفر می شود و در if پایینی تصمیم گرفته می شود بیت ذخیره شود یا \* و در نهایت ternary به عنوان خروجی داده می شود.

خروجی جدول TCAM