1396.11.5

به نام خدا

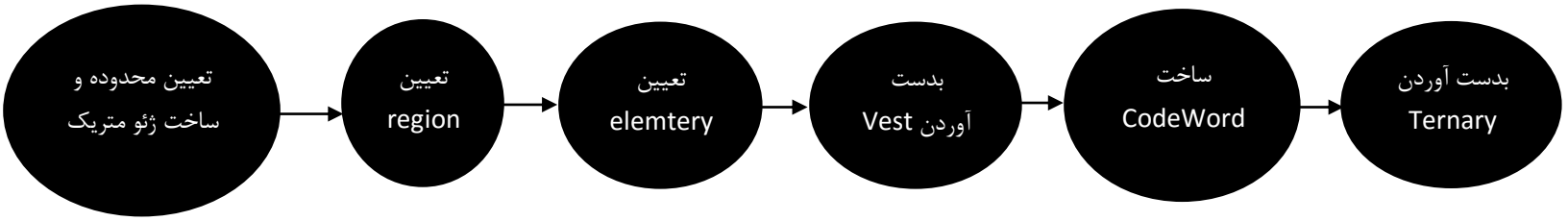
جابر بابکی 96131020

پروژه دوم

Multi-Field Range Encoding for Packet Classification in TCAM

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

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 به rule ها بدست بیاوریم.
3	setRule()	در این تابع در واقع با توجه به محدوده تعیین شده ruleها و regionها 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 CodeWordهای 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 :

```
public List<RuleModel> MyExample2() {
  List<RuleModel> ruleModels = new ArrayList<RuleModel>();
  RuleModel R1 = new RuleModel( name: "R1", source: "00", destedition: "110");
  ruleModels.add(R1);
  RuleModel R2 = new RuleModel( name: "R2", source: "00", destedition: "11");
  ruleModels.add(R2);
  RuleModel R3 = new RuleModel( name: "R3", source: "10", destedition: "1");
  ruleModels.add(R3);
  RuleModel R4 = new RuleModel( name: "R4", source: "0", destedition: "01");
  ruleModels.add(R4);
  RuleModel R5 = new RuleModel( name: "R5", source: "0", destedition: "10");
  ruleModels.add(R5);
  RuleModel R6 = new RuleModel( name: "R6", source: "0", destedition: "1");
  ruleModels.add(R6);
  RuleModel R7 = new RuleModel( name: "R7", source: "*", destedition: "00");
  ruleModels.add(R7);
  RuleModel R8 = new RuleModel( name: "R8", source: "11", destedition: "10");
  ruleModels.add(R8);
  RuleModel R9 = new RuleModel( name: "R9", source: "10", destedition: "11");
  ruleModels.add(R9);
  RuleModel R10 = new RuleModel( name: "R10", source: "01", destedition: "01");
  ruleModels.add(R10);
  RuleModel R11 = new RuleModel( name: "R11", source: "111", destedition: "11");
  ruleModels.add(R11);
  RuleModel R12 = new RuleModel( name: "R12", source: "011", destedition: "0");
  ruleModels.add(R12);
  RuleModel R13 = new RuleModel( name: "R13", source: "101", destedition: "011");
  ruleModels.add(R13);
  RuleModel R14 = new RuleModel( name: "R14", source: "110", destedition: "111");
  ruleModels.add(R14);
  RuleModel R15 = new RuleModel( name: "R15", source: "100", destedition: "1");
  ruleModels.add(R15);
  RuleModel R16 = new RuleModel( name: "R16", source: "000", destedition: "101");
  ruleModels.add(R16);
  RuleModel R17 = new RuleModel( name: "R17", source: "11", destedition: "011");
  ruleModels.add(R17);
  RuleModel R18 = new RuleModel( name: "R18", source: "00", destedition: "011");
  ruleModels.add(R18);
  RuleModel R19 = new RuleModel( name: "R19", source: "1", destedition: "111");
  ruleModels.add(R19);
  RuleModel R20 = new RuleModel( name: "R20", source: "111", destedition: "111");
  ruleModels.add(R20);
  return ruleModels;
}
```

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.

TABLE 3.2 Example Classifier with Seven Rules in Four Fields

Rule	F_1	F_2	F_3	F_4	Action
R_1	00*	110*	6	(10, 12)	Act_0
R_2	00*	11*	(4, 8)	15	Act_1
R_3	10*	1*	7	9	Act_2
R_4	0*	01*	10	(10, 12)	Act_1
R_5	0*	10*	(4, 8)	15	Act_0
R_6	0*	1*	10	(10, 12)	Act_3
R_7	*	00*	7	15	Act_1

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

```
public List<RuleModel> MyExample() {
  List<RuleModel> ruleModels = new ArrayList<RuleModel>();
  RuleModel R1 = new RuleModel( name: "R1", source: "00", destedition: "110");
  ruleModels.add(R1);
  RuleModel R2 = new RuleModel( name: "R2", source: "00", destedition: "11");
  ruleModels.add(R2);
  RuleModel R3 = new RuleModel( name: "R3", source: "10", destedition: "1");
  ruleModels.add(R3);
  RuleModel R4 = new RuleModel( name: "R4", source: "0", destedition: "01");
  ruleModels.add(R4);
  RuleModel R5 = new RuleModel( name: "R5", source: "0", destedition: "10");
  ruleModels.add(R5);
  RuleModel R6 = new RuleModel( name: "R6", source: "0", destedition: "1");
  ruleModels.add(R6);
  RuleModel R7 = new RuleModel( name: "R7", source: "*", destedition: "00");
  ruleModels.add(R7);
  return ruleModels;
}
```

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 :

```java
package com.ario.original2_dranges.model;

/**
 * Created by jaberALU on 20/01/2018.
 */

public class ElemtryModel {


  private String region;
  private String elementaryRegion;
  private String rule;

  public ElemtryModel(String region, String elementaryRegion, String rule) {
    this.region = region;
    this.elementaryRegion = elementaryRegion;
    this.rule = rule;
  }
  public String getRegion() { return region; }
  public String getElementaryRegion() { return elementaryRegion; }
  public String getRule() { return rule; }
  public void setRegion(String region) { this.region = region; }
  public void setElementaryRegion(String elementaryRegion) {...}
  public void setRule(String rule) { this.rule = rule; }
}
```

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

```java
public void init() {
  for (int i = 0; i < Orginal2D.length; i++) {
    for (int j = 0; j < Orginal2D.length; j++) {
      Orginal2D[i][j] = new ElemtryModel( region: null,  elementaryRegion: null,  rule: "");
    }
  }
}
```

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.

2–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 :

```java
package com.ario.original2_dranges.model;

/**
 * Created by jaberALU on 31/12/2017.
 */


public class LimitedModel {

  private int dl = 0;
  private int du = 0;
  private int Sl = 0;
  private int Su = 0;
  private int okER = 0;

  public int getOkER() { return okER; }
  public void setOkER(int okER) { this.okER = okER; }
  public int getSl() { return Sl; }
  public int getSu() { return Su; }
  public int getDl() { return dl; }
  public int getDu() { return du; }
  public void setSl(int sl) { Sl = sl; }
  public void setSu(int su) { Su = su; }
  public void setDl(int dl) { this.dl = dl; }
  public void setDu(int du) { this.du = du; }

}
```

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.

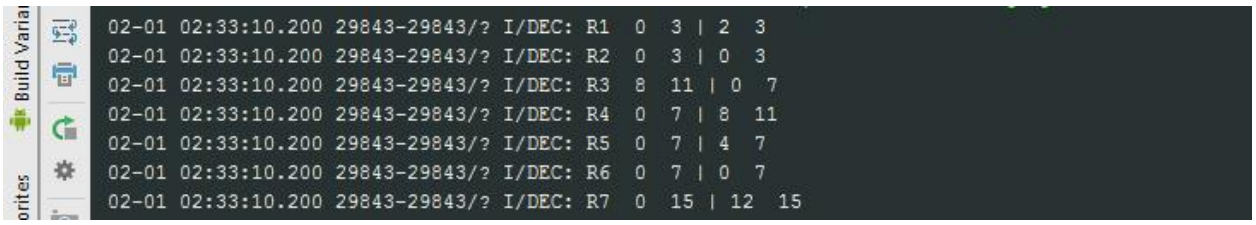
```java
public void getLimted() {
  for (int i = 0; i < ruleModel.size(); i++) {
    LimitedModel lim = new LimitedModel();
    String source = ruleModel.get(i).getSource();
    String dest = ruleModel.get(i).getDestedition();
    String sl = ruleModel.get(i).getSource();
    String su = ruleModel.get(i).getSource();
    if (!source.equals("*")) {
      int s = 4 - source.length();
      for (int j = 0; j < s; j++) {
        sl = sl + "0";
      }
      for (int j = 0; j < s; j++) {
        su = su + "1";
      }
    } else {
      sl = "0000";
      su = "1111";
    }
    lim.setSl(Integer.parseInt(sl,  radix: 2));
    lim.setSu(Integer.parseInt(su,  radix: 2));
    String dl = dest;
    String du = dest;
```

```
        if (!dest.equals("*")) {
            int d = 4 - dest.length();
            for (int j = 0; j < d; j++) {
                dl = dl + "0";
            }
            for (int j = 0; j < d; j++) {
                du = du + "1";
            }
        } else {
            sl = "0000";
            su = "1111";
        }
        lim.setDl(15 - (Integer.parseInt(du,  radix: 2)));
        lim.setDu(15 - (Integer.parseInt(dl,  radix: 2)));
        limited[i] = lim;
    }

}
```

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.

3–**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:

```
public void setRule() {
  int m = 0;
  for (int i = 0; i < ruleModel.size(); i++) {
    for (int o = limited[i].getSl(); o <= limited[i].getSu(); o++) {
      for (int j = limited[i].getDl(); j <= limited[i].getDu(); j++) {
        Orginal2D[j][o].setRule(Orginal2D[j][o].getRule() + "," + ruleModel.get(i).getName());
        if (Orginal2D[j][o].getRegion() == null) {
          Orginal2D[j][o].setRegion("r" + m);
          m++;
        }
        //}
      }
    }

  }
}
```

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

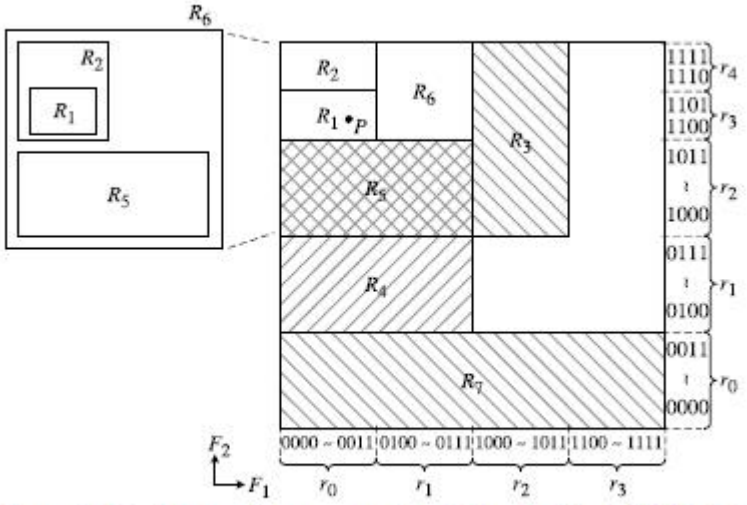
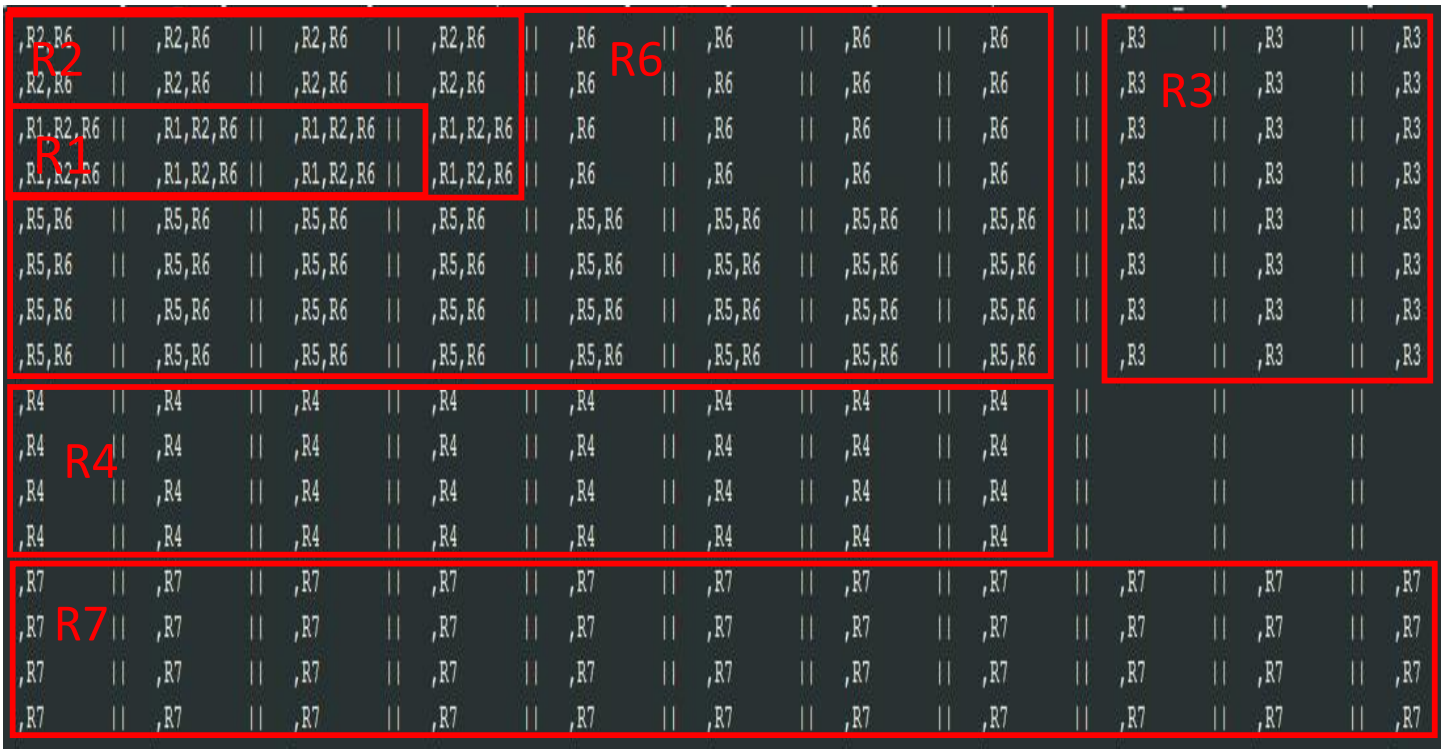


Figure 3.10 Geometric representation of the classifier in Table 3.2.

و اما خروجی تابع من :



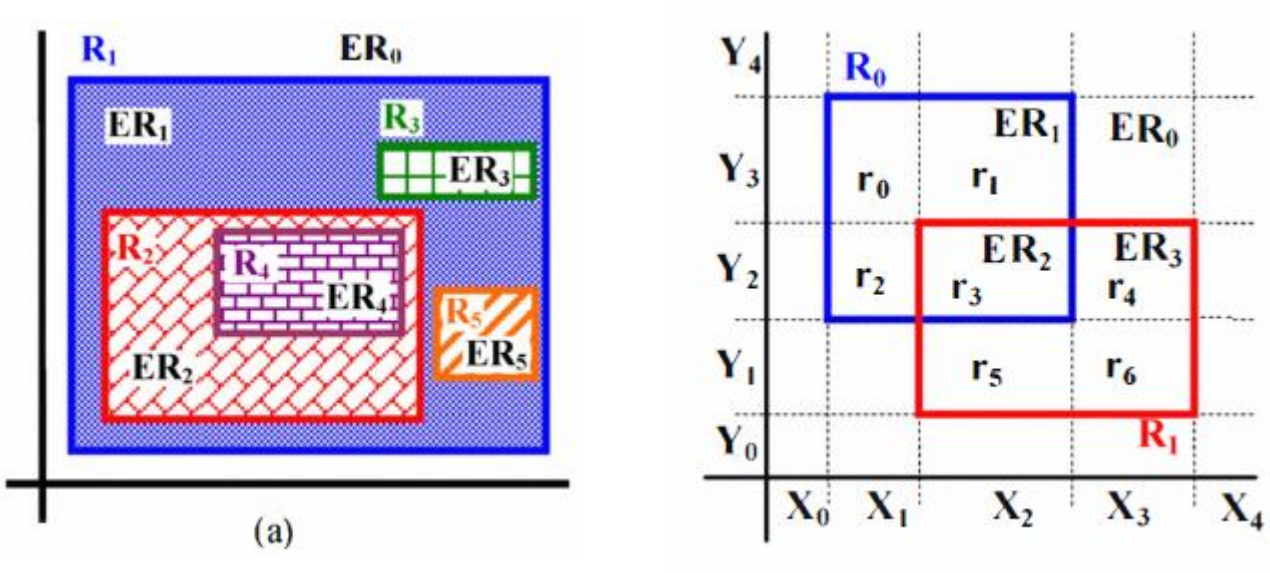
همانطور که دیده می شود کاملا شبیه زئو متریک هست که در کتاب هست. نکته مهم اینکه region ها همانطور که در تابع بود تعیین شدند اما جهت شلوغ نشدن نشان ندادم .

4–**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.

(1) All addresses in ER_i are covered by the same subset of original 2-D ranges (called the range matching set of ER_i, denoted by ER_i.range), and (2) The range matching sets of two different elementary regions are not equivalent.

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 regionهای 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 ruleکه با هم جدا هستند باید ER بخورند و همچنین باقی مانده elemteryها باید ER صفر بخوردند

با توجه به این قوانین تابع زیر عمل می کند:

```
public void setElemtryRegion() {
  int y = 0;
  for (int i = 0; i < limited.length; i++) {
    for (int j = 0; j < limited.length; j++) {
      if (i != j) {
        if (limited[i].getSl() >= limited[j].getSl() && limited[i].getSu() <= limited[j].getSu() &&
          limited[i].getDl() >= limited[j].getDl() && limited[i].getDu() <= limited[j].getDu()) {
          y++;
          setElemntry(limited[i], ("ER" + y));
          limited[i].setOkER(1);
          break;
        }
      }
    }
  }

  for (int i = 0; i < limited.length; i++) {
    for (int j = 0; j < limited.length; j++) {
      if (i != j && limited[i].getOkER() != 1) {
        if (limited[j].getSl() <= (limited[i].getSu() - limited[i].getSl()) && (limited[i].getSu() - limited[i].getSl()) < limited[j].get
          && limited[j].getDl() <= (limited[i].getDu() - limited[i].getDl()) && (limited[i].getDu() - limited[i].getDl()) < limited[j].ge
          y++;
          Log.i( tag: "DEC",  msg: "" + i + " " + j);
          setElemntry(limited[i], ("ER" + y));
          limited[i].setOkER(1);
          break;
        }
      }
    }
  }
```

```java
    for (int i = 0; i < limited.length; i++) {
        for (int j = 0; j < limited.length; j++) {
            if (i != j && limited[i].getOkER() != 1) {
                y++;
                setElemntry(limited[i], ("ER" + y));
                limited[i].setOkER(1);
            }
        }
    }

}
```

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 .

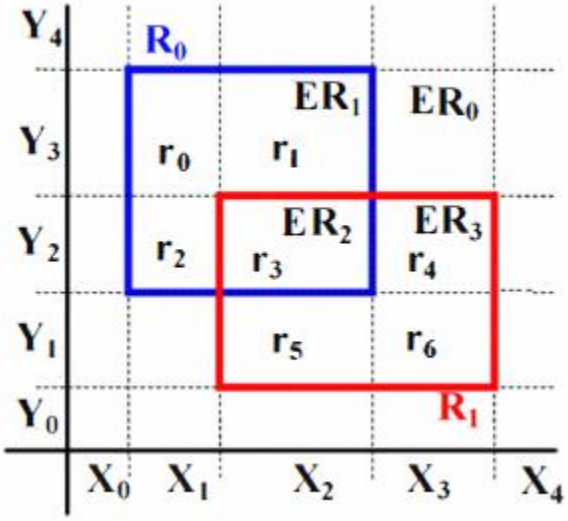
خروجی این مرحله به صورت زیر است :

```
ER2 || ER2 || ER2 || ER2 || ER6 || ER6 || ER6 || ER6 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER2 || ER2 || ER2 || ER2 || ER6 || ER6 || ER6 || ER6 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER1 || ER1 || ER1 || ER1 || ER6 || ER6 || ER6 || ER6 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER1 || ER1 || ER1 || ER1 || ER6 || ER6 || ER6 || ER6 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER3 || ER4 || ER4 || ER4 || ER4 || null || null || null || null ||
ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || null || null || null || null || null || null || null || null ||
ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || null || null || null || null || null || null || null || null ||
ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || null || null || null || null || null || null || null || null ||
ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || ER5 || null || null || null || null || null || null || null || null ||
ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 ||
ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 ||
ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 ||
ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 || ER7 ||
```

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.

5– **تعیین** CodeWord

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



EI_x, EI_y	Region	ER*i*	codeword
(X_0, Y_0)	-	ER_0	00
(X_1, Y_2)	r_2	ER_1	01
(X_1, Y_3)	r_0	ER_1	01
(X_2, Y_1)	r_5	ER_3	10
(X_2, Y_2)	r_3	ER_2	11
(X_2, Y_3)	r_1	ER_1	01
(X_3, Y_1)	r_6	ER_3	10
(X_3, Y_2)	r_4	ER_3	10

Other regions are omitted because they are the same as (X_0, Y_0).
R0 and R1 are encoded as ternary strings *1 and 1*, respectively.

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 regionهایی 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 :

```
package com.ario.original2_dranges.model;

/**
 * Created by jaberALU on 31/12/2017.
 */


public class CodeWordModel {
  public String[] codeWord = new String[]{"0","0","0","0","0","0","0"};
  public String getER() { return ER; }
  private String ER="";
  public void setER(String ER) { this.ER = ER; }
}
```

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

```
public void getCodeWord() {
    for (int i = 0; i < limited.length; i++) {
        for (int o = limited[i].getSl(); o <= limited[i].getSu(); o++) {
            for (int j = limited[i].getDl(); j <= limited[i].getDu(); j++) {
                if (!Orginal2D[j][o].getRule().equals("")) {
                    CodeWordModel code = new CodeWordModel();
                    String[] str = Orginal2D[j][o].getRule().split( regex: ",");
                    for (int f = 0; f < str.length; f++) {
                        if (f != 0) {
                            code.setER(Orginal2D[j][o].getElementaryRegion());
                            int p = Integer.parseInt(str[f].substring(1, str[f].length()));
                            code.codeWord[p - 1] = "1";
                        }
                    }
                    codeWord.add(code);
                }
            }
        }
    }
}
```

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 :

```
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER1  0 1 0 0 0 1 1
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER2  0 1 0 0 0 1 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER3  0 1 1 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER5  0 0 0 1 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER7  1 0 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER6  0 1 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER6  0 1 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER3  0 1 1 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER5  0 0 0 1 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER7  1 0 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER7  1 0 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER7  1 0 0 0 0 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER4  0 0 0 0 1 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER4  0 0 0 0 1 0 0
02-01 04:13:47.337 2942-2942/com.ario.original2_dranges I/DEC: ER4  0 0 0 0 1 0 0
```

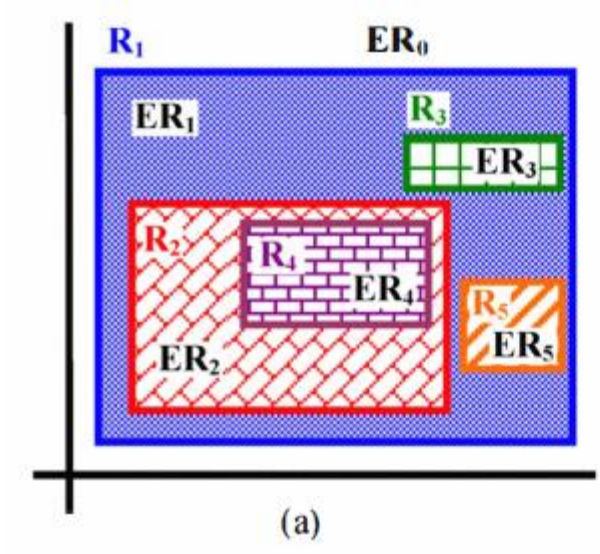
Build Variants

avorites

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

6–**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 .



R_i	$Vset_i$
R_1	ER_1, ER_2, ER_3 $ER_4, ER_5, VR_1,$ VR_2, VR_3
R_2	ER_2, ER_4
R_3	ER_3
R_4	ER_4
R_5	ER_5

(b)

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 :

```
package com.ario.original2_dranges.model;

/**
 * Created by jaberALU on 31/12/2017.
 */


public class VestModel {

  private String rule ="";
  private String numberER ="";

  public String getRule() { return rule; }
  public void setRule(String rule) { this.rule = rule; }
  public void setNumberER(String numberER) { this.numberER = numberER; }
  public String getNumberER() { return numberER; }

}
```

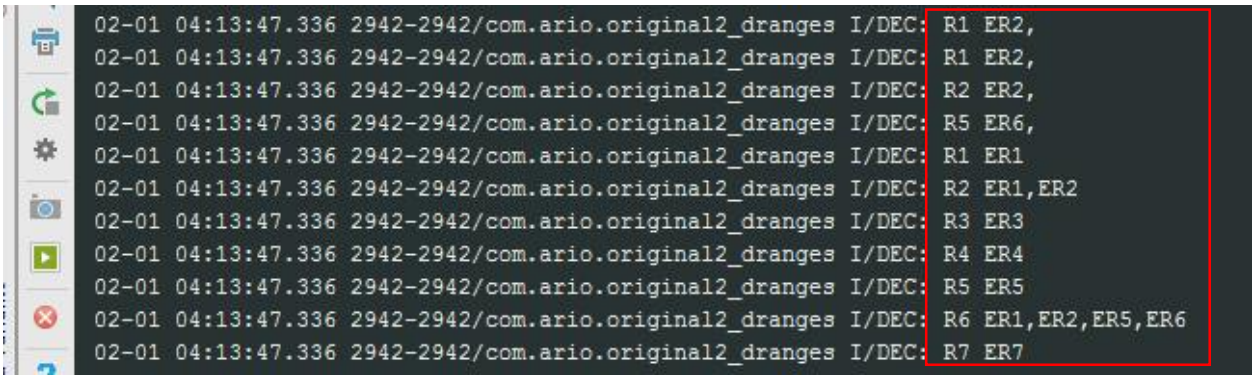
همانطور که دیده می شود یک rule و ER ها را نگه می دارد.

```
public void getVest() {
  for (int i = 0; i < limited.length; i++) {
    for (int j = 0; j < limited.length; j++) {
      if (i != j) {
        if (limited[i].getSl() >= limited[j].getSl() && limited[i].getSu() <= limited[j].getSu() &&
          limited[i].getDl() >= limited[j].getDl() && limited[i].getDu() <= limited[j].getDu()) {
          VestModel ves = new VestModel();
          ves.setRule(ruleModel.get(i).getName());
          ves.setNumberER(Orginal2D[limited[i].getSl()][limited[i].getSu()].getElementaryRegion() + "," + ves.getNumberER());
          vest.add(ves);
        }
      }
    }
  }

}
```

در تابع getVest ما می آییم ER هایی که زیر مجموعه Ruleهستند 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 :



7–**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

```java
package com.ario.original2_dranges.model;

/**
 * Created by jaber babaki on 7/21/2016.
 */
public class TernaryModel {

  private String ternary;
  private String rule;

  public void setRule(String rule) { this.rule = rule; }
  public void setTernary(String ternary) { this.ternary = ternary; }
  public String getRule() { return rule; }
  public String getTernary() { return ternary; }
}
```

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:

```java
public void getTernary() {
  for (int i = 0; i < vest.size(); i++) {
    String[] v = vest.get(i).getNumberER().split( regex: ",");
    List<String[]> codeWord = new ArrayList<String[]>();
    for (int j = 0; j < v.length; j++) {
      String[] code = searchInCodeWord(v[j]);
      codeWord.add(code);
    }
    String ternarStr=getTernaryForOne(codeWord);
    TernaryModel ternar=new TernaryModel();
    ternar.setRule(vest.get(i).getRule());
    ternar.setTernary(ternarStr);
    ternary.add(ternar);
  }
}
```

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 codeWordرا میاریم بیرون و درون یک رایه می ریزیم برای تحلیل شباهت، سپس با استفاده از تابع getTernaryForOne می آییم اون codeWordهایی 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

```
public String[] searchInCodeWord(String er) {
  for (int i = 0; i < codeWordOrginal.size(); i++) {
    if (er.equals(codeWordOrginal.get(i).getER())) {
      return codeWordOrginal.get(i).codeWord;
    }
  }
  return null;
}
```

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

تابع getTernaryForOne

```
public String getTernaryForOne(List<String[]> codeWord) {
  String ternary = "";
  for (int i = 0; i < codeWord.size(); i++) {
    for (int j = 0; j < codeWord.get(i).length; j++) {
      int y = 1;
      for (int f = 0; f < codeWord.size(); f++) {
        if (i != codeWord.size() - 1) {
          if (codeWord.get(i)[j] != codeWord.get(i + 1)[f]) {
            y = 0;
          }
        }
      }
      if (y == 1) {
        ternary = ternary + "" + codeWord.get(i)[j];
      } else {
        ternary = ternary + "*";
      }

    }
  }
  return ternary;
}
```

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

```
02-01 04:59:20.889 19488-19488/com.ario.original2_dranges I/DEC: R1 0100011
02-01 04:59:20.889 19488-19488/com.ario.original2_dranges I/DEC: R2 010001*
02-01 04:59:20.889 19488-19488/com.ario.original2_dranges I/DEC: R3 0110000
02-01 04:59:20.889 19488-19488/com.ario.original2_dranges I/DEC: R4 0000100
02-01 04:59:20.890 19488-19488/com.ario.original2_dranges I/DEC: R5 0001000
02-01 04:59:20.890 19488-19488/com.ario.original2_dranges I/DEC: R6 0*0*0**
02-01 04:59:20.890 19488-19488/com.ario.original2_dranges I/DEC: R7 1000000
```