A-1 ReporT

We are given a budget of 2400, 6400, 9999, 32000 bits. We are required to implement the branch predictors, useful for the branch prediction.

1) 2400:
   Here, I implemented a *Pap Predictor*, which has a ghr register table, having 64 entries and each entry is a *4-bit Global history register*. Also, the size of the pattern history table(PHR) is 1024 and it uses a saturating counter of 2-bit length.

2) 6400:
   Here, I implemented a *Tournament predictor* in which *Pap* predictor is predictor-1 and *G-share* predictor is predictor-2.

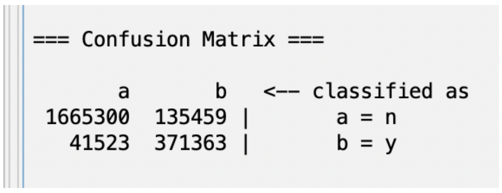
3) 9999:
   Here, too, I implemented a Tournament predictor, in which the *Pap* predictor is predictor-1 and the *G-share* predictor is predictor-2.

4) 32000:
   Here, I used a neural network technique, known as perceptron.
   Perceptron:
   The history length of each row in the table is 59. I contained the information about the last 59 branches I encountered in a global history register. And I implemented a table, which has 8 weights.

The accuracy that I was able to achieve was between 96 - 97.

Machine Learning Techniques:
I chose the weka toolkit to find out the accuracy of branch predictors. I converted all the given traces into .csv.arff format and then ran it on the weka toolkit to find out the accuracy predicted by those ML algorithms in weka.
I used the *REPTree* algorithm which is a tree based algorithm to find out the accuracy prediction of the branches.
Test-options → use training set.

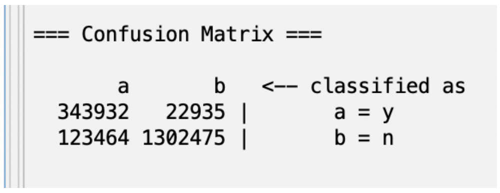
1) *trace1.csv.arff:*
   Accuracy Predicted: 92.005 %
   Confusion Matrix:



```
=== Confusion Matrix ===

      a       b   <-- classified as
 1665300  135459 |       a = n
   41523  371363 |       b = y
```

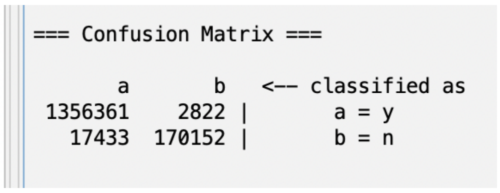
2) *trace2.csv.arff:*
   Accuracy Predicted: 91.8341
   Confusion Matrix:



```
=== Confusion Matrix ===

      a       b   <-- classified as
 343932   22935 |       a = y
 123464 1302475 |       b = n
```

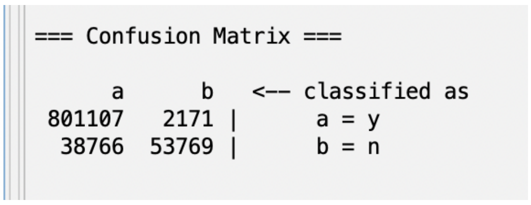
3) *trace3.csv.arff:*
   Accuracy Predicted: 98.6905 %
   Confusion Matrix:



```
=== Confusion Matrix ===

      a       b   <-- classified as
1356361    2822 |       a = y
  17433  170152 |       b = n
```

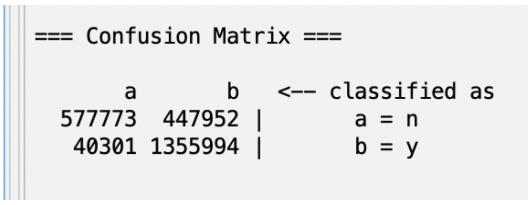
4) *trace4.csv.arff:*
   Accuracy Predicted: 95.4302 %
   Confusion Matrix:



```
=== Confusion Matrix ===

     a      b   <-- classified as
801107   2171 |      a = y
 38766  53769 |      b = n
```

5) *trace5.csv.arff:*
   Accuracy Predicted: 79.8411 %
   Confusion Matrix:



```
=== Confusion Matrix ===

      a       b   <-- classified as
 577773  447952 |       a = n
  40301 1355994 |       b = y
```

Tejas

Tejas is an open-source, Java-based multi-core architectural simulator, which is platform-independent.
It can simulate binaries in any ISA and corresponding to any OS. Tejas is a trace driven simulator.
Tejas consists of an emulator, which is responsible for execution of the program and also generation of traces..

Traces typically contain all the information about the instructions executed and branch outcomes.
The traces are sent to the timing simulator via transfer engine.
Then Instructions are translated to VISA in the simulator.
These streams of instructions are taken as input, and pipelines are simulated.