Vol.2014-EMB-34 No.5
2014/9/17

ロード値予測ミスの偏りを利用したロード値予測器の検討

森脇　信啓[1]　孟 林[†1]　小柳　滋[1]

概要：近年のスーパースカラプロセッサでは，性能を向上のために周波数を上げることは限界に近づいている．そこで，命令レベルの並列性を向上することが必要となっている．命令レベルの並列性の向上を阻害する要因の一つとしてデータ依存があり，これを軽減する手法として，ロード値予測がある．しかし，ロード値予測ミスが存在し，近年パイプライン段数の増加と命令発行幅の増加により，予測ミスのペナルティが増加してしまう．本研究は，ロード値予測ミスを分析し，予測ミスが少数のロード命令に集中して発生することを発見し，これらのロード命令を検知して専用なハードウェアを，従来のロード値予測器に追加し予測を行う手法を提案し，ロード値予測の精度を向上することを目指す．

1. はじめに

命令レベル並列性は，マルチコア技術や，GPGPU による汎用処理があるこの現代においても，プロセッサの性能の向上する方法の一つである．その理由の一つは，マルチコア技術や GPGPU は，命令レベル並列性に依存するスーカースカラプロセッサで構築されるからである．さらに，GPGPU やマルチコアは大きなハードウェアと大量の電力を必要とする．よって，組み込みシステムで構築することが実現しにくい．従って，スーパースカラプロセッサの性能を改善する方法を決定するために，命令レベル並列性を再考する必要がある．命令レベル並列性の向上の最大の問題は，真のデータ依存である．さらに，ロード命令の時に，生じたロード遅延の削減も大きな課題となる．

ロード値予測は，ロード遅延を軽減させる効果的な手法である．ロード値予測器は，ロード値の履歴を書き込み，ロード命令がフェッチされたときにロードの値を予測する機構である．これまでにいくつかの予測器が提案されている [1], [2], [3], [4], [5]．しかし，ロード値予測ミスが存在し，近年パイプライン段数の増加と命令発行幅の増加により，予測ミスのペナルティが増加してしまう．このように，ロード値予測では予測率を上げるだけでなく，予測精度の向上も必要となる．

本稿では，従来のロード値予測器の動作を分析することにより，ロード値の予測ミスが一部分のロード命令に集中して発生すると考えた．ここで，予測ミスが集中して発生するロード命令のみに対処するロード値予測機構を従来の予測器に付加することにより，性能を向上させる予測器を目指す．

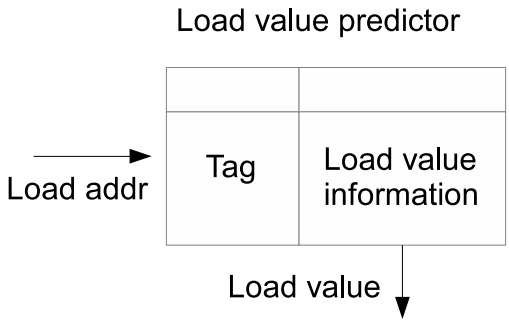

図 1　Load value prediction diagram.

本論文は以下のように構成する．2 章では，先行研究を説明する．3 章では最終値予測器を用いて，ロード値予測ミスが一部のロード命令に集中的に発生することを示す．また，4 章ではロード値予測ミスのパターンを示し，5 章でそのパターンに対応する予測器を提案する．6 章はまとめと今後の課題である．

2. 先行研究

ロード値予測はロード命令の値を予測することより，ロード遅延を削減し，プロセッサの性能向上と繋がる有効な方法である [1], [2], [3], [4], [5]．図 1 は，ロード値予測機構のブロック図を示し，タグ (Tag) とロード値の情報 (Load value information) を保存している．タグは，ロード命令のアドレスの一部であり，ロード値の情報はロード命令の過去の値などの情報を含んでいる．

[1] 立命館大学　情報理工学部
College of Information Science and Engineering, Ritsumeikan University
[†1] 現在，立命館大学　理工学部
Presently with College of Science and Engineering, Ritsumeikan University

2.1 ロード値予測の動作

ロード値予測の動作に，予測と更新が含まれている．

- **予測**：
  ロード命令がフェッチされているときに，ロード命令のアドレスの一部をインデクス (load addr) として，予測器にアクセスする．もし，ロード命令のアドレスが予測器のタグと一致する場合は，ロード値の情報を用いてロード値の予測を行う，一致しない場合はロード値の予測を行わない．
- **更新:**
  ロード命令がコミットされているときに，ロード命令のアドレスの一部をインデクス (load addr) として，予測器にアクセスする．もし，ロード命令のアドレスが予測器のタグと一致する場合は，ロード値の情報を更新する．一致しない場合は．タグとロード値の情報を新しいデータとして予測器に登録する．

2.2 先行研究の例

いくつかのロード値予測の先行研究が提案された．

- **Last Value Predictor** [1], [2]:
  last value predictor は一番シンプルな予測器で，ロード命令の前回の結果を用いて，予測値として使用する．
- **Stride method**[3]:
  ストライド予測は，前回と前々回のロード値の差 (ストライド) を前回のロード値に加算したものを予測値として使用する．
- **Differential Finite Context Method (DFCM)**[4]:
  DFCM はストライドの履歴を保持することで，ストライドのパターンを発見することでロード値を予測する．
- **Two-hop address renaming**[5]:
  2 ホップアドレス名前変え予測は，同一アドレスにアクセスするロード命令とストア命令を関連付ける手法である．予測テーブルにあるロード命令アドレスから，メモリに書き込んだストア命令を取り出すことで，予測値を得る．

3. 予測ミスの偏り

我々の研究では，分岐予測器について，予測ミスの偏りが存在しているため [7]，本稿では，従来のロード値予測器において，予測ミスの特徴を分析する．

まず SimpleScalar Tool Set [8] を用いて，最終値予測を実装して評価を行う．ベンチマークには SPECint2000 から bzip，gcc，gzip，mcf，parser，vortex，vpr の 7 本を使用する．プロセッサの仕様を表 1 に示す．命令セットは SimpleScalar PISA を用いる．そして，予測ミスの命令とそれらの予測ミスの数を数え，ミス全体での割合を計算する．ここで，予測ミスが少数のロード命令アドレスに集中して発生することが分かった．

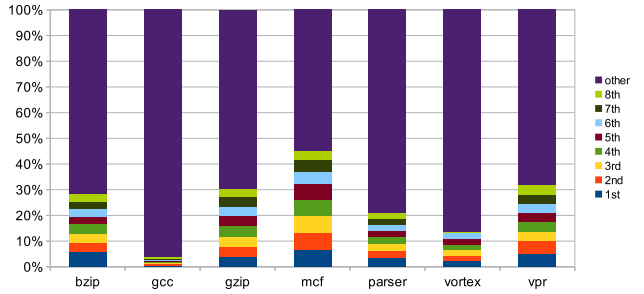

図 2 8 本のよくミスしたロード命令のミス全体における割合

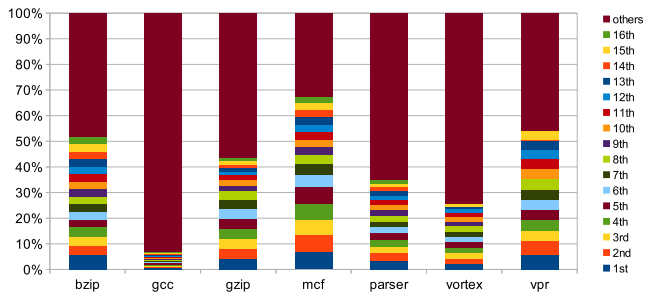

図 3 16 本のよくミスしたロード命令のミス全体における割合

ここから詳細について述べる．これらのベンチマークにおいて，最も多く予測ミスが発生する上位 8 個と 16 個のロード命令アドレスを抽出し，これらの予測ミスの，全体の予測ミスに占める割合を調べる．

図 2 と図 3 では，100M 命令を実行し，8 本と 16 本のよくミスした命令の結果である．横軸はベンチマークであり，縦軸は予測ミスが最も多い上位 8 個と 16 個のロード命令の予測ミスの，全体の予測ミスに占める割合である．これらの結果から見ると，予測ミスが一部のロード命令に偏っていることが分かった．例えば，Bzip,gzip,Mcf,vpr の最も予測ミスした 1 番の命令が，ミスの全体の 5%ぐらいを占めている．最も偏っているのが，mcf で，8 本のよく

表 1 プロセッサの構成

Pipeline	5 stages: 1 Fetch, 1 Decode, 1 Execute 1 Memory Access, 1 Commit
Fetch, Decode, Dispatch	4 instructions
Issue	Int: 4, fp: 2, mem: 2
Window	Dispatch queue: 256, Issue queue: 256,
BTB	2K-entry 4-way associative BTB, 32-entry RAS
Memory	64KB, 4-way associative, 1-cycle instruction and date caches 2MB, 8-way associative, 10-cycle L2

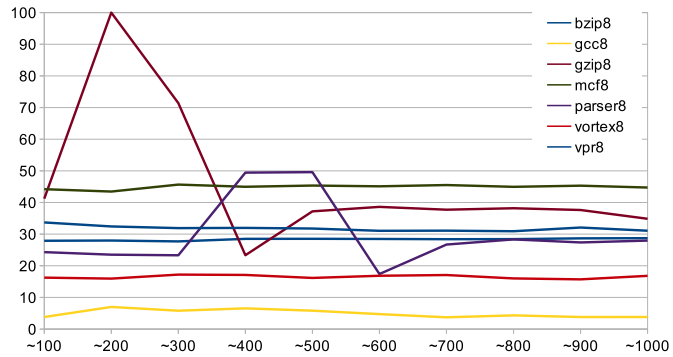

図 4 予測ミス上位 8 個が占める予測ミスの偏り

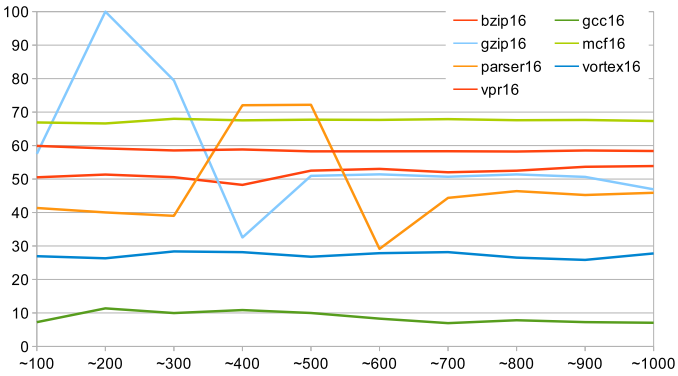

図 5 予測ミス上位 16 個が占める予測ミスの偏り

ミスしたものが全体のミスの 40%以上となっている．

更なる分析を行うために，1M 毎で，予測ミスの割合を調べている．図 4 と図 5 では，予測ミスが最も多い上位 8 個と 16 個のロード命令の予測ミスが，全体の予測ミスに占める割合を示す．なお，予測ミスはプログラムの実行状況に応じて変化するため，調べ方については，1M 命令ごとに最もミスの多い上位 8 個（あるいは 16 個）のロード命令を抽出し，それらのミスが全体のミスに占める割合を調べ，これを 10 回繰り返して 10M 命令まで評価する．

図 4 と図 5 の縦軸は，各ベンチマークにおける全体の予測ミスに対する，割合を示す．横軸は，1M ごとの命令数を示す．図より，予測ミス上位 8 個と 16 個では gcc と vortex のベンチマークを除き，上位 8 個では全体の約 25%，上位 16 個では全体の約 50%の予測ミスを占めていることが分かる．これより，特定のロード命令アドレスが全体の予測ミスを占めていると考えられる．

4. 予測ミスの偏りの分析

3 章では，ロード値予測ミスが少数のロード命令に偏るという特性を持つことを示した．この章では，全体の予測ミスを占めるロード命令の，値の遷移を分析することで，その特徴を見出す．実行環境は 3 章と同じく Simple Scalar Tool Set を用いて，最終値予測において，予測ミス上位 16 個までのロード命令アドレスにおいて，どのように値が遷移するのかを調査した．その結果，およそ 4 つのパターンを発見することが出来た．

- **カウンタ型**
  カウンタ型では，ロード値が 100 の次は 101，102 のように，カウンタのように増えていくように予測ミスが発生していた．
- **1 組の値の繰り返し**
  1 組の値の繰り返しでは，最初に 272，次に 368，そしてまた 272 というように，1 組の値を繰り返しロード値とするものである．
- **2 回連続で同一の値を繰り返す**
  これは，ロード値が 110，110，45，45，115，115 のように，一つのペアを繰り返していくものである．2 回連続で同一の値を取った後にロード値が別の値となっている．
- **複数の値でパターンを構成しているもの**
  これは，ロード値が 2097，2094，2103，2096，2097，2103，2105，2108，2203，2096，2203，2096，…と続き，最初の 2097 に戻るといったパターンである．

本稿では，4.1 から 4.3 までの，カウンタ型，1 組の値の繰り返し，2 回連続で同一の値を繰り返すものに着目する．表 2 に SPECint2000 の bzip ベンチマークにおいて，実際にどのようなパターンがあったのかを示す．

表 2 より，bzip のベンチマークでは，上記の 3 パターンが多く見られた．特に，上位 1 位と 3 位から 7 位までは，ミスのパターンは 1 つで構成されていた．この内，上位 3 位と 5 位の予測ミスの原因は全てカウンタ型であった．また，上位 2 位の予測ミスではカウンタ型と 2 回連続で同一の値を繰り返すものだけで構成されていた．

さらに，他のベンチマークでも上記の 3 パターンを含むミスを発見することが出来た．これらより，カウンタ型，1 組の値の繰り返し，2 回連続で同一の値を繰り返すものを予測器に付加すれば，予測ミスを減らすことが期待出来る．

5. 値予測ミス偏りを利用したロード値予測付加機構

3 章と 4 章では，ロード値予測ミスが少数のロード命令に偏るという特性をもつことを示した．我々はこの特性を利用し，ロード値予測ミスの多いロード命令について，いくつかのパターンが予測できようなローカル予測器を，ベース予測器に付加するハードウェア機構を提案する．

図 6 に，last value 予測器をベースにした提案するハードウェア機構のブロック図を示す．ベース予測器に付加される部分は拡張された Last Value 予測器 ELVP (Extended Load Value Predictor) と MBB(Miss Bias Buffer) により構成される．ELVP はロード値予測を行いながら，予測ミスの多いロード命令を検出する機構である．MBB は検出された予測ミスの多いロード命令のローカル履歴を利用し

表 2　bzip ベンチマークにおける予測ミスの詳細

上位 8 個 (addr)	パターン		
	カウンタ型	1 組の値の繰り返し	2 回連続で同一の値を繰り返すもの
1st(4270584)		272-368-272-368-...	
2nd(4270256)	199-200-201-202-...		782-782-783-783-784-...
3rd(4254408)	230-231-232-233-...		
4th(4270424)			231-231-0-0-233-233-...
5th(4255400)	598-599-600-601-...		
6th(4270496)			110-110-45-45-115-115-...
7th(4270560)			110-110-45-45-115-115-...
8th(4255496)			

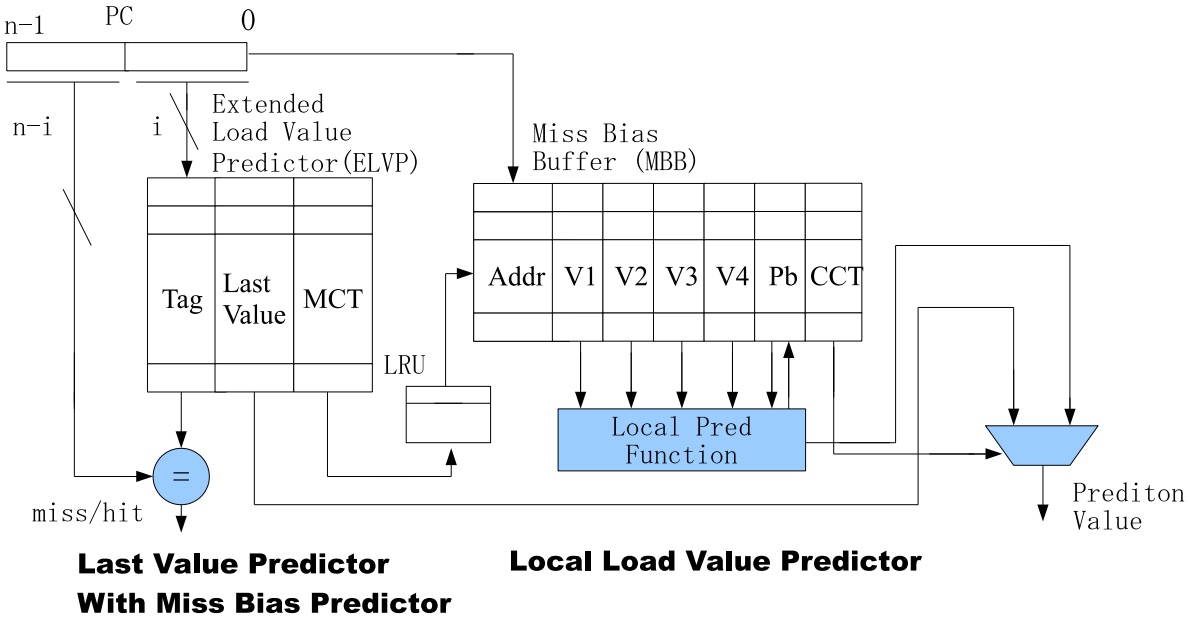

図 6　提案するロード値予測器のブロック図

て，値予測を行う機構である．

予測の流れとしては，まず ELVP では，拡張された Last Value 予測器を利用し，予測ミスの多発するロード命令を検出する．そして，検出されたロード命令のローカル履歴などの情報を MBB(Miss Bias Buffer) に記憶する．

さらに，MBB では，予測ミスの多発するロード命令について，該当するロード命令のローカル履歴を用いて，ロード値予測のパターンを決め，Load Pred Function を用いて，ロード値を予測する．最後に， Selector で，MBB により予測された結果とベース予測器により予測された結果のいずれかを選択する．MBB の予測結果が優先的に使用する．

5.1　ELVP による予測ミスの多発するロード値の予測

ELVP は，従来のロード値予測機構を利用して各エントリに飽和カウンタの MCT(Miss Counter) を追加し従来のロード値予測器の予測ミスを数える．ELVP では従来のロード値予測器と同じように，タグ (Tag) と最後のロード値（Last Value）が設けられている．MBB は，予測ミスの多発するロード命令を格納するもので，Addr，V1, V2, V3, V4（Value1 Value4），Pb(Pattern bit), CCT(Correct Counter) により構成される．

- Addr はロード命令アドレスである．
- V1~ V4 は該当するロード命令の 4 つのロード値の履歴で，シフトレジスタで構成される．
- Pb は該当するエントリのパターンビットである．パターンビットが 0 のときに未使用を意味し，1 の時に繰り返しパターンで，2 のときにカウンターパターンで，3 のときにその他のパターンである．
- CCT は予測成功の履歴を保存している 2 ビット飽和カウントである．成功するときに，インクリメントし，失敗するときにデクリメントを行う．

前章により，多くのベンチマークでは 8 個あるいは 16 個のロード命令に予測ミスが集中的に発生しているため，MBB のサイズは 8 あるいは 16 に設定する．MBB の具体的な動作を以下で説明する．

- **MBB への登録:**
  ロード命令がコミットされるときに，ロード命令のア

ドレスを用いて ELVP を検索する．ELVP がヒットし，かつ予測ミスの場合は，ELVP の MCT をインクリメントする．ELVP がヒットしない場合は，従来のロード値予測と同じように Tag の更新を行い，予測ミスした時に MCT を 1 にし，予測成功した時に MCT を 0 にする．ELVP エントリの MCT が閾値に到達すると，MBB にロード命令のアドレスを登録する．そして，該当する ELVP のエントリの MCT をリセットする．

MBB への登録は，以下のように行われる．まず登録しようとしたロード命令のアドレスが MBB に存在するかどうかをチェックする．このアドレスが MBB に存在しない場合には，MBB の Pb が 0 となるエントリが存在するならば，そのエントリに登録し，Pb を 3 にする．もし，MBB の Pb がすべて 0 ではない場合は，LRU(Least Recently Used)[9] ロジックを利用して，最も 最近利用されていないエントリを選択し，登録する．なお，LRU ロジックにおいて，MBB が正しく予測でき，かつ ベース予測器が正しく予測できない場合が MBB を利用したものと判定する．

- **MBB の更新:**
  ロード命令がコミットされるとき，当該ロード命令が MBB に登録されているときは MBB の更新が行われる．V1~ 4 の更新について，V1,V2,V3,V4 をシフトし，最新のロード値を V1 に登録する．また，Pb の更新について，V1，V2，V3，V4 の値を用いて行う．V1~ V4 が異なる場合は，カウンターパターンと判断し，2 にする．そうではない場合は，カウンターパターンと判断し，1 にする．
  CCT の更新について，成功する場合は 1 をインクリメントし，失敗する場合は，デクリメントを行う 2 ビット飽和カウンタと同じ動作である．

5.2 MBB によるロード値予測

MBB は予測ミスの多発するロード命令の保存と予測を行うものである．予測ミスの多発するロード命令毎に個別のローカル値履歴を保存する．

ロード命令がフェッチされるときに，フェッチされたロード命令のアドレスを用いて，ELVP を検索する．ELVP に存在する場合は，ELVP を用いて値予測を行う．それと同時に，フェッチされたロード命令のアドレスを用いて，MBB を検索する．MBB 内に，ロード命令のアドレスが存在する場合は，Load Pred Function を用いて，ロード値の予測を行う．

ロード予測値の生成について，以下となっている．

- Pb が 1 のときに，繰り返しパターンと判断し，V1 の値が MBB の予測値とする．
- Pb が 2 のときに，カウンタパターンと判断し，V4+ パターンの値が MBB の予測値とする．
- その他の場合は，MBB の予測値がなし．

さらに，Selector において，MBB の予測結果とベース予測の予測結果のいずれかを選択する．具体的には MBB の CCT が 2 以上のときに，MBB の値を使用する．そうではない場合は，ELVP の予測値が，予測値として使用する．

6. おわりに

近年のプロセッサではパイプライン段数の増加と命令発行幅の増加により，ロード値予測ミスのペナルティが増加するため，ロード値予測の精度向上がプロセッサの性能向上に繋がる．本稿では，ロード値予測ミスの特徴について，調査と分析を行った．ここで，ロード値予測ミスが少数のロード命令に偏っている特徴を発見し，さらに，分析により，ミスのパターンも存在している．これらの発見に基づいて，従来の予測器に，よくミスしたロード命令の専用予測機構を付加するロード値予測器を提案した．また，このハードウェアに利用して，ロード値予測ミスの削減と繋げると考えられる．今後の課題として，既存の予測器に実際に付加することで，性能向上を評価を行う．また．gcc のベンチマークでは，特定のロード命令アドレスが全体の予測ミスを占めるということは示しなかったため，提案手法が実際の適用出来るとは考えられない．よって，これを検討して，更なる予測ミスを軽減することが課題である．

参考文献

[1] Mikko H. Lipasti, Christopher B. Wilkerson, and John Paul Shen, ”Value Locality and Load Value Prediction”, the Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.138-147, 1996.
[2] Martin Burtscher and Benjamin G.Zorn, ”Exploring last n value prediction”, the International Conference on Parallel Architectures and Compilation Techniques, pp.66-76, 1999.
[3] Yiannakis Sazeides and James E. Smith, ”The predictability of data values”, In the 30th International Symposium on Microarchitecture, pp.248-258, 1997.
[4] Bart Goeman, Hans Vandierendonck and Koen De Bosschere, ”Differential FCM: Increasing Value Prediction Accuracy by Improving Table Usage Efficiency”, the Seventh international Symposium on High Performance Computer Architecture (HPCA’01), pp.207-216, 2001.
[5] 佐藤寿倫, ”2 ポップアドレス名前替えを用いたロード命令の投機的実行”, 情報処理学会論文誌, vol.40, No.5, pp.2109-2118, 1999.
[6] Erik Jacobson, Eric Rotenberg and J.E.Simth, ”Assigning confidence to conditional branch predictions”, the Proceedings of 29th Int’l Symposium on Microarchitecture, pp.142-152, 1996.
[7] 孟林, 小柳滋,”分岐予測ミスの偏りを利用した分岐予測器の提案,”情報処理学会論文誌,Vol.4/ No.4, pp.85-95,2011.
[8] D. Burger and T. M. Austin, ”The SimpleScalar Tool Set

Version2.0", Technical Report, University of Wisconsin-Madison Computer Sciences 1997.
[9] C.C.Kavar and S.S.Paramar, "Performance Analysis of LRU Page Replacement Algorithm with Reference to different Data Structure", International Journal of Engineering Research and Application, Vol.3, No.1, pp.2070-2076, 2013.