コンピュータと教育　61−5
（２００１．１０．１９）

初等アセンブラプログラミング評価支援システムのための事例ベース構築法

渡辺 博芳　　荒井 正之　　武井 惠雄
帝京大学理工学部

本稿では，初等アセンブラプログラミングを対象とした事例に基づくプログラム評価支援システムのための事例ベース構築法について述べる．事例に含まれるプログラムリストは，対象となるプログラミング言語自身で表現されるべきである．事例のプログラムリストが特別な形式で表現されていると，評価支援システムのユーザとしての教員がその特別な表現形式を習得しなければならないからである．一方，対象とするプログラミング言語そのままの形式で表現されたプログラムリストを持つ事例では，適用範囲が狭いという欠点がある．そこで，プログラムリストの一般化情報を用いて，事例へのインデックスを構築し，事例の適用範囲を広げることが考えられる．本論文では，CASL による初等アセンブラプログラミングを対象として，階層構造を持つインデックスの構築法を提案する．更に，実際の授業で提出されたプログラムを用いた実験を行い，提案した手法の有効性を示す．

A Method of Constructing Case-Bases for Evaluation Assistant of Novice Programs Written in Assembly Language

Hiroyoshi Watanabe,　Masayuki Arai　and　Shigeo Takei
School of Science and Engineering, Teikyo University

This paper presents a method of indexing cases for case-based evaluation assistant systems of novice programs written in an assembly language. Program lists in evaluation cases should be represented in intact target programming language, because special forms of program lists put heavy burdens on teachers who are users of the systems. However, intact forms of program lists cannot cover that many variations. Therefore, indexes to cases should be constructed by using information of generalized program lists in order to expand the variations of program lists covered by one case. We propose a three-level index of evaluation cases. Retrieval experiments with submitted programs in actual classes demonstrated the effectiveness of the proposed index method.

１．　はじめに

典型的な初等プログラミング演習授業では, 教員が提示した問題の題意を満たすようなプログラムを学生が作成し, 提出されたプログラムやレポートを教員が評価するという形態をとることが多い. このような授業において, 全ての学生に題意を満たすプログラムを完成させようとした場合, 題意を満たすプログラムを全学生が提出するまで再提出を繰り返させるので, 教員は延べ学生数以上のプログラムを評価することになる. 従って, 教員が学生のプログラムを評価する作業の負荷は非常に大きくなる.

我々は, CASL による初等アセンブラ・プログラミングを対象として, 提示した課題に対して学生が作成したプログラムを教員が評価する作業を支援するシステムを開発した[1,2]. 本システムは, 学生が提出したプログラムに対して, あらかじめ用意したテストデータを用いた動作の評価と, 事例に基づく実現方法の評価を行う. 開発したシステムを実際の授業で使用することで, 我々のアプローチが教員の評価作業負荷の軽減に大きな効果があることを明らかにした. 一方で, 開発したシステムの性能を更に向上させ, より実用的にするためには, 事例ベース構築方法を改善する必要があることがわかった.

本稿では, 事例ベース活用率の向上を目的とした事例の階層的なインデックス構成法を提案する.

２．対象とする評価作業と評価支援システム

2.1 対象とする評価作業

対象とするプログラム評価は, 提示した課題に対して学生が作成した(提出した)プログラムに対する以下の 2 つの作業である.

(1) 提出されたプログラムが題意を満たしているかどうかの判定:教員が問題を提示する際には, 学生に習得させたい概念等に関する教育的な意図がある. 教員は提出されたプログラムを分析し, 意図していた概念を学生が習得したかどうかの評価を行い, 学生にアドバイスをしたり, 場合によってはプログラムの再提出を求める. 本研究が対象とする評価作業の 1 つは, このような, 学生が作成したプログラムが提示した問題の題意を満たしているかどうかを判定する作業である. 以降では, 題意を満たしている場合は「合格」, そうでないときは「不合格」といい, この題意を満たすかどうかの評価作業を「合否判定」と呼ぶことにする.「不合格」という表現は, 現在判定対象となっている, そのプログラムが不合格であることを示し, 提示された問題について学生が不合格となるのとは異なることに注意されたい. また, 学生がプログラムを提出するのは 1 度だけでなく, 合格に至るまで何度も提出を繰り返すことを想定している.

(2) 提出されたプログラムに対するアドバイスの作成:本研究で対象とする 2 つめの評価作業は, アドバイス文の作成である. アドバイスは, プログラムが題意を満たすか否かに関わらずに, 必要に応じて与える. プログラムが題意を満たさない場合, どのような点が題意を満たしていないかアドバイスが必要である. また, 題意を満たす場合でも, よりよいプログラムにするためのアドバイスを与える.

2.2 プログラム評価処理の流れ

実現したシステムを用いたプログラム評価の流れを図 1 に示す. 学生がプログラムを提出すると, システムは最初に複数組のテストデータを用いてプログラムの動作を評価する. 動作が正しくない場合, どのようなデータに対して動作しないかを示し, 提出が受理されなかった旨のメッセージを提示する. 動作が正しい場合は, 事例に基づく実現方法の評価を行う. 教員はシステムの出力(評価結果)を編集し, 最終的な評価を行う. その結果は自動的に電子メールで学生に通知されるとともに, 事例ベース更新のための情報として利用する.

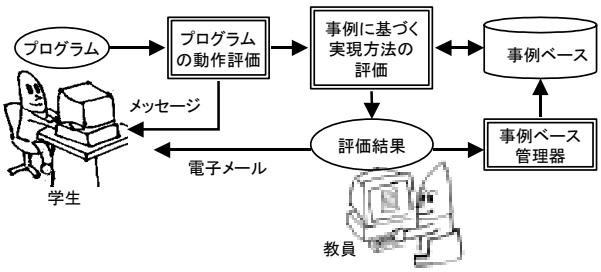

図 1 プログラム評価処理の流れ

2.3 事例に基づくプログラム評価

１つの事例は, 問題記述として評価対象となるプログラムリスト, 解記述として合否判定結果(合格または不合格)とアドバイス文から構成される．また，事例識別子，事例作成者，事例更新年月日, などの事例管理のための情報も持たせる.

評価処理においては, 学生が提出したプログラムに照合するプログラムリストを持つ事例を検索し，それらの中から，学生のプログラムに最も類似する事例を選択する. 照合条件を満たす事例が存在すれば，その事例の評価結果(合否判定とアドバイス文)を評価対象に適用する．合否判定結果はそのまま適用する．つまり，事例の合否判定結果が合格であるなら，評価対象も合格となる．また，アドバイス文については, プログラム照合で求めた対応関係情報を利用して，ラベル名，レジスタ番号や行番号の置き換え程度の簡単な修正を施す.

評価結果を教員に提示する際には, 学生のプログラムと事例のプログラム照合結果に基づいて, システムの生成した評価結果の確信度を出力する．完全に照合し，ほとんど同じと見なせる場合は確信度を Surely とする．完全照合ではないが, 対応関係をつけることができる場合は確信度を Probably とする．確信度が Surely の場合は，システムの評価結果を教員がチェックせずに直接学生に送ることもできる.

３．事例のインデックス法

3.1 問題点

我々が最初に実現したシステム[1]は，フラットな事例ベースを用いていたが, 事例ベースの活用率を向上させる工夫として, 事例に含まれるプログラムを一般化した表現で記述していた．プログラムの一般化表現は，１つのプログラムリストで複数のプログラムと照合できるように，我々が独自に定義した形式である．このシステムでは，以下のような問題がある.

(1)事例照合処理の計算コスト大：フラットな事例ベースでは, 事例照合処理において評価対象のプログラムを事例ベース中の全ての事例と照合することになる. これでは明らかに効率が悪い.

(2)プログラムリストの一般化表現による問題:事例中のプログラムを一般化表現とすることで, システムのユーザとしての教員が一般化表現を習得する必要がある．(事例の評価結果が適用可能な範囲に一般化レベルを設定する必要があるため, 教員が手動で一般化表現に変換する必要がある．) また，問題の題意などの環境条件の変化[3]が生じると，一般化レベルを調整する必要があり, 事例ベース管理処理が複雑になる.

3.2 基本方針

前節で述べた問題を解決し, 更に事例ベース活用率を向上させることを目的として, プログラムリストの一般化情報を用いて, 階層構造のインデックスを構築するアプローチをとった．その基本方針について述べる.

プログラムリストの一般化表現による問題を回避するために, 事例中のプログラムリストは，一般化表現とせずに，CASL の文法に従った表現をとる. 事例中のプログラムリストを一般化しない場合, そのままでは 1 つの事例の適用範囲が狭くなってしまうが, プログラムリストの一般化レベルに基づいて, 事例へのインデックスを階層化することで対応する. プログラムリストの一般化においては, 次の 3 つの問題

を解決しておかなければならない.

(1)命令の種類に関する一般化：一つの操作が 2 種類以上の命令によって実現可能なことがある．例えば,「あるレジスタに定数を設定する操作」は，CASL では，LD(load)命令とLEA(load effective address)命令の両方で実現できる. このような使用する命令の違いによるバリエーションをカバーしなければならない．これは，我々が独自に定義した一般化表現と一般化ルール[1,2]を用いて行うことができる．2 種類以上の命令で実現できる場合に，それらを同一視してよいかどうか(確信度を surely とするかどうか)は，問題の題意によって異なることもあり得るので, 問題毎に教員がカスタマイズ可能とする. カスタマイズの方法については，3.4 で述べる．

(2)命令の順序に関する一般化：同一の命令群を用いたプログラムでも, 個々の命令の記述順序が異なる場合がある. 命令の順序に関して, ある命令がループの中にある場合と外にある場合で, 評価結果やアドバイスが異なる可能性があるので，命令の順序が異なる場合は，基本的には確信度を Probably とする．ただし，以下の場合は,「順序の違いは些細である」として順序が異なるとは見なさないこととする.

- 順序が異なる範囲に含まれる命令のオペコードが全て等しい.
- 順序が異なる範囲に含まれる命令が指標レジスタ修飾を伴わない，レジスタへの値の設定か，レジスタ主記憶間のデータの転送に関する命令(LEA,LD,ST)のみから構成される.

(3)冗長命令の有無に関する一般化：ある命令の有無に関わらず, プログラムの動作が同一である場合, その「命令は冗長である」という．そのような冗長命令の有無に関するバリエーションをカバーしなければならない. 冗長命令が含まれることで，可読性が悪くなる場合と，逆に良くなる場合もある．そこで, 冗長命令の有無で評価結果やアドバイスが異なる可能性があるので, 冗長命令に関する情報が異なる場合は，確信度を Probably とする.

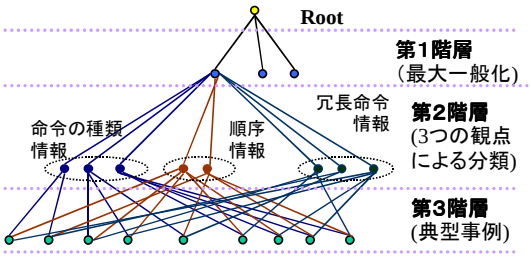

図 2 事例ベースインデックスの概要

3.3 インデックスの階層

提案する手法によって構成した事例ベース構造の概要を図 2 に示す．個々の問題ごとに，図 2 のようなネットワーク構造の事例ベースを構成する．事例ベースは次の階層*から成る.

ルート ： インデックスのルートは，検索の際のスタート地点となる．ルートノードに，第 1 階層のノードへのインデックスを持たせる．インデックス情報は，第 1 階層の個々のノードがカバーするプログラムリストにおける各命令数(26 命令分)の上限値と下限値を持つ.

第 1 階層 ： プログラムリストを可能な限り一般化したノード(プログラムリストに，本システムが持っている全ての一般化ルールを適用して得られる一般化表現を表すノード)である.

第 2 階層 ： 前節で述べた 3 つの観点ごとに，3 種類のノードのグループを形成する．つまり，命令の種類に着目したときに同じと判断される事例は，命令の種類情報グループ内にある同一のノードにリンクする．また，命令の順序関係が同じと判断される事例は，順序情報グループの同一のノードにリンクする．冗長命令についても同様である. このように, 1 つの事例は個々のグループについて必ず 1 ノードとリンク

＊ 文献[4,5]ではルートを第 1 階層とし, 4 階層と表現しているが, 本稿ではルートを階層に含めずに，3 階層と表現した．表現が異なるだけで，内容は同一である.

される.

第 3 階層 ： 典型的な事例を配置する. ここでは，プログラム中のラベル名，レジスタ名が異なる場合も同一と見なす.

3.4 一般化レベルのカスタマイズ

命令の種類に関する一般化について, 2 種類以上の命令での実現方法を同一視してよいかどうかをチェックリスト形式で教員が設定できるようにする. 具体的には, 一般化命令とそれに対応するチェックリスト用の説明文をあらかじめ用意しておき, チェックが付けられた説明文に対応する一般化命令を第 2 階層のノードを生成する際に適用する.

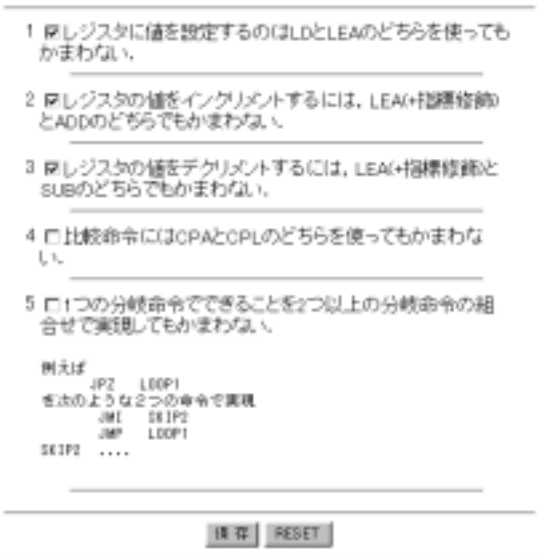

図 3 一般化レベルカスタマイズのためのインターフェースの例

図 3 に 2000 年度版の評価支援システムにおけるチェックリスト設定のためのウェブベースのインターフェースを示す. それぞれの説明文は 1 つ以上の一般化命令に関連付けされている. このようなインターフェースを提供することで, 第 2 階層で適用する一般化ルールを教員が問題ごとに個別に設定可能とする. 図 3 で既にチェックが入っている項目はデフォールト値であり，教員が設定を行わない場合は，それらが使われる.

４．事例検索と事例ベース更新

4.1 冗長命令検出処理

これまで述べた方法に基づいて，2000 年度に実現した評価支援システム[4](idx1 と呼ぶ)では, プログラムリストから 1 ずつ命令を削除しては動作を確認するという方法で冗長命令を検出していた. しかし，この方法では同時には冗長とならない 2 つ以上の命令が存在する場合, それらを全て冗長命令と見なしてしまう. これによって, フラットな事例ベースに比較して事例ベース活用率が低下することがあった.

そこで，2001 年度のシステム[5](idx2 と呼ぶ)では，同時には冗長とならない命令を区別することとした. それによって，冗長命令は複数組になることがあり，その場合，1 つの事例に対して複数のインデックス経路を作成する. 1 事例について複数の経路を作成することで，適切な事例に到達する可能性を高めるためである. idx2 での冗長命令の検出処理は，以下のようになる.

- 最初に，プログラムを構成する命令を 1 つずつ削除しては動作を確認して，冗長命令の候補とする.
- 次に，それら候補中の 2 命令を組合せて同時に削除して動作を確認することで，冗長命令を求める.

以上の処理によって, 同時には冗長とならない命令が存在するケースに対応できるが, それでも，例えば,「ある 2 つの命令と別の 1 つの命令が同時には冗長とならない」といったケースは検出できない. 冗長命令を完全に検出するには, 相当数の組合せについて動作確認を行う必要があり, 検出処理の負荷が膨大になる. 検出処理の負荷とそれによって向上する事例ベース活用率とのトレードオフを考慮して, 上で述べたような処理を行うこととした.

事例検索アルゴリズム：

初期化(*CASE*←空，*DIFF*←最大値)

個々の冗長命令の組に対して以下を行う*1.

1. *PL1* から冗長命令を削除して *PL2* とする．

2. インデックスのルートの命令数情報を用いて照合する可能性がある第 1 階層ノードの候補*2 を求める．これを *NL1* とする．

3. *NL1* 中の個々のノード *N1* について以下を行う．

3.1 *N1* と *PL2/PL1* の照合を行う．*3

3.2 もし，照合条件*4 を満たすなら，以下を行う．

3.2.1 *N1* の下位の第 2 階層ノード情報を基に，最も類似する事例を選択し，これを N3 とする．

3.2.2 *N3* と *PL1* の照合を行う．*5

3.2.3 もし，*N3* と *PL1* が完全照合するなら，*CASE* ← *N3* とし，検索処理を終了

3.2.4 もし，*N3* と *PL1* が類似照合するなら，*N3* と *PL1* の差異を求め，それが *DIFF* よりも小さければ *CASE* ← *N3* とし，*DIFF* を更新する．

変数の説明：

PL1：評価対象のプログラムリスト

PL2：*PL1* から冗長命令を削除したもの

CASE：求める事例(これまでで最も類似する事例)

DIFF ： これまでで最も類似する事例との差異

N1 ： 第 1 階層ノード

N3 ： 候補事例(第 3 階層ノード)

注釈：

***1** 冗長命令がない場合は，冗長命令を空として検索処理を 1 度だけ行う．

***2** 最初は，*PL2* の命令数が上限と下限を満たすノードを選択．2 回目は，*PL1* の命令数が下限のみを満たすノードを選択．

***3** 最初は *PL2* と照合，2 回目は *PL1* と照合．

***4** 最初の条件は，「順序関係は無視して全ての命令が 1 対 1 対応になること．」2 回目の条件は，「*N1* のプログラムの命令が全て *PL1* に対応先を持つこと．」

***5** 直接照合できない場合上位ノードを介して照合する．

図 4 検索処理の流れ

4.2 事例検索処理

図 4 に事例検索処理のための基本的なアルゴリズムを示す．2000 年度のシステムの事例検索では図 4 の処理を 1 度だけ行うが，これでは，「プログラムの命令部分が同一でも，データ部分の順序の違いにより冗長命令が異なるケース」に対応できないことがわかった．そこで，2001 年度のシステム(idx2)では，図 4 の処理を 2 回行うこととした．

最初は，評価対象のプログラムリストから冗長命令を削除したものを用いて，事例ベースのインデックスを辿ることで，事例を検索する．これで，完全照合する事例を得た場合は，2 回目の処理は行わない．

完全照合の事例を得られない場合は，2 回目の検索処理を行う．ここでは，冗長命令も含めた(提出されたそのままの)プログラムリストを用いて事例ベースのインデックスを辿る．2 回目の検索処理では第 1 階層のノードとの照合条件をゆるめることで，適切な事例に到達する可能性を高める．ただし，第 2 階層以下の照合条件は最初の検索処理と同様である．

4.3 事例ベース更新処理

教員の最終的な評価結果の情報を用いて事例ベースを更新する．まず，事例の検索を行い，完全照合の事例が存在し，その事例の評価結果が今回の教員の評価結果と異なる場合は，評価結果部分を上書きする．完全照合の事例が存在しない場合は，新事例を追加し，インデックスを更新する．事例ベース更新処理における事例検索は，前節で述べた 2 回の検索処理のうち，最初の処理のみを行う．

インデックス更新において，事例検索の結果，第 1 階層と第 2 階層に照合するノードが存在すれば，それらからのリンクを付加する．第 1 階層と第 2 階層のノードのうち，照合するノードが存在しない部分については，冗長命令の削除と一般化ルールの適用を行って新しいノードを生成し，ルートからのリンクを付加する．

5．実験

5.1 実験条件

1999 年度と 2000 年度に提示した 17 問に対して提出されたプログラムを用いて事例ベース活用率の評価実験を行った. 事例ベースにインデックスを構築する実験システムとして, 先に述べた idx1，idx2 の他，フラットな事例ベースを持つ実験システムを flat と表す.

実験システムは, 個々の問題に対して提出されたプログラムを学籍番号順(同じ学生が複数提出した場合は提出順)に読み込み，事例の検索を行う．事例の検索結果を記録した後, 新しく事例を追加する必要があれば, 新事例の追加を行う．提出されたプログラム全てに対して, 上の処理を行った結果,「全プログラム数」に対する「照合する事例を得たケース」の割合を事例ベース活用率(%)とする.

表 1 事例ベース活用率(%)の比較

問題	Flat	idx1	idx2	問題	flat	idx1	idx2
1	89.3	90.7	94.7	10	44.1	47.1	52.0
2	87.3	88.6	91.1	11	46.6	55.7	69.3
3	78.1	71.9	79.2	12	77.2	79.7	86.1
4	67.6	69.6	73.5	13	88.9	91.4	91.4
5	59.0	55.1	66.7	14	83.3	85.1	85.1
6	78.6	83.3	91.7	15	95.0	93.8	95.0
7	84.8	88.0	90.2	16	61.7	75.2	82.7
8	62.5	63.6	81.8	17	63.9	68.0	69.7
9	44.6	50.5	72.3	平均	71.3	74.0	80.7

(注) No.1,No.6,No.12，及び No.2,No.7,No.13 はそれぞれ同じ問題であるが，年度とクラスが異なっているので，提出されたプログラムは異なる.

5.2 実験結果と考察

5.2.1 事例ベース活用率

実験の結果, 事例ベース活用率の比較を表 1 に示す. 表 1 のデータを基に, flat と比較して, どれだけ，事例ベース活用率が増加，あるいは減少したかを図 5 に示す.

idx1 は, flat と比較して, 事例ベース活用率は概ね向上しているが，No.3,5,15 の問題においては減少した. その原因は, 次の通りである.

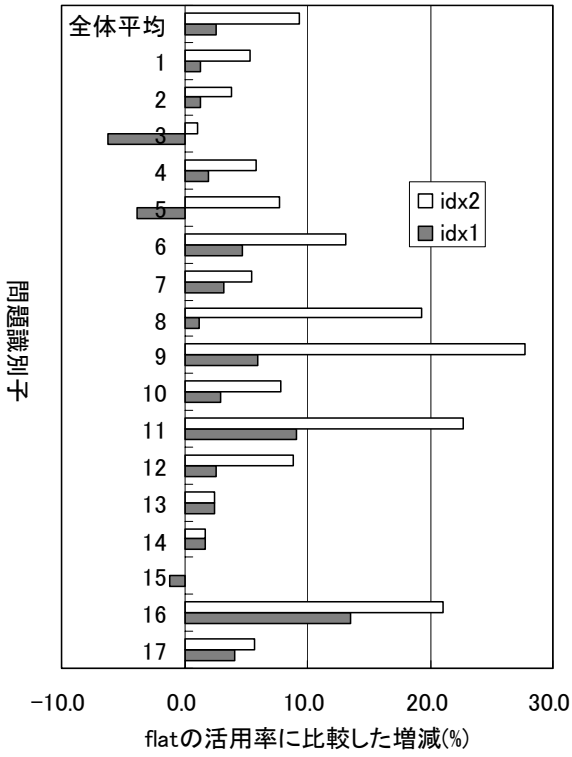

図 5 flat に比較した事例ベース活用率の増減

- 同時には冗長とならない命令を全て冗長命令と見なしてしまうため，適切な事例が存在しても，それに到達できない場合があったこと.
- アセンブラプログラムでは，プログラム部分が全く同じでも，データ部分の配置順序が異なると動作結果が異なる場合がある．冗長命令はプログラム動作結果に基づいて検出するので，プログラム部分が同じ場合でも，冗長命令が異なる場合があったこと.

idx2 においては，これらの問題点が改善されているため，事例ベース活用率を flat と比較すると，No.15 で等しい他は，全て向上している．また，いくつかの問題では事例ベース活用率を大幅に向上させている. 全体の平均では, idx2 は flat と比較して 9.4%, idx1 と比較して 6.7%，事例ベース活用率を向上させており, 提案手法が事例ベース活用率の向上について，有効であることが明らかである.

5.2.2 検索処理時間

1 つの評価対象プログラムに対する事例検索処理にかかる時間(平均値)の比較を図 6 に示す．これは flat を 1 としたとき，idx1，idx2 がそれぞれ何倍になるかを計算したものである．図 6 を見ると，idx1，idx2 とも，2 つの問題を除き, flat よりも検索時間が短縮されているので, 提案手法で構築する事例ベースインデックスが, 検索処理の効率化にも有効であることがわかる．

idx1 と idx2 を比較すると, idx1 の方が処理時間は短い．これは，idx2 の方が，検索処理が複雑になっているためである．ただし，Pentium3(1GHz)搭載 PC を使用した今回の実験で，1 つの評価対象に対する検索時間は，ほとんどが 1 秒以下であり, 最も検索時間がかかったケースで 2 秒であった. 従って, flat, idx1, idx2 とも検索時間は実用範囲内であり，検索処理の効率化よりも, 事例ベース活用率の向上を優先すべきであるといえる．

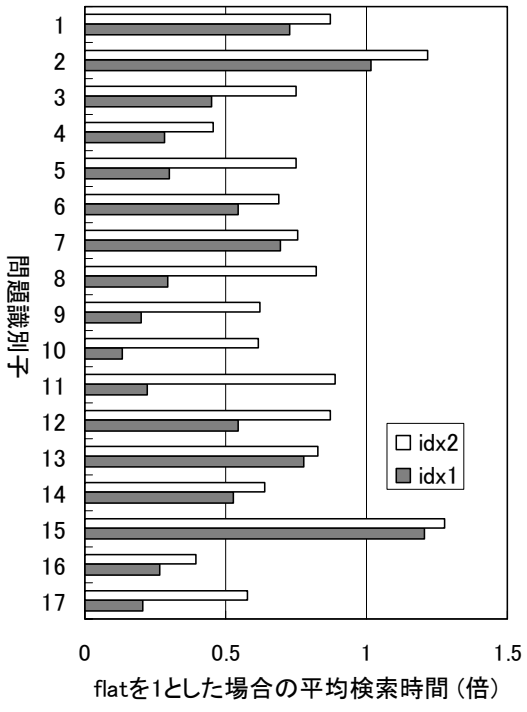

図 6 検索処理時間の比較

6．むすび

CASL を対象としたプログラム評価支援システムにおいて, 事例ベース活用率を向上させるための事例ベース構築法を提案し, 実験により，提案手法の有効性を示した．

本手法を用いて，CASL II に対応したプログラム評価支援システムを開発中である. 今後, そのシステムを実際の授業で使用して評価を行いたい．

謝辞 本研究の一部は科学研究費補助金 No. 12780293，人工知能研究振興財団 研究助成 No. 12AI320-1 の補助による．

参考文献

[1] 渡辺博芳，荒井正之，武井恵雄：事例に基づく初等アセンブラプログラミング評価支援システム，情処論, Vol.42，No.1, pp.99 〜 109, 2001.

[2] Watanabe,H., Arai,M. and Takei,S. : Case-Based Evaluation of Novice Programs, Proc. of AI-ED 2001, San Antonio pp.55 〜 64, 2001.

[3] 渡辺博芳，荒井正之，武井恵雄：事例ベース推論によるプログラム評価支援，情処研報，Vol.2000-CE-56，pp.9 〜 16, 2000.

[4] 渡辺博芳，荒井正之，武井恵雄 ：初等アセンブラプログラミングを対象とした事例に基づくプログラム評価 のための事例ベース構築法，情処学会第 61 回全大, 3P-6, 2000.

[5] 渡辺博芳，荒井正之，武井恵雄 ：初等アセンブラプログラミングを対象とした事例に基づくプログラム評価のための事例ベース構築法の評価， 情処学会第 63 回全大，5X-7，2001.