УДК 51:155.001.57:681.3.06

В.П. КЛИМЕНКО, Ю.С.ФИШМАН, С.В. КОНДРАШОВ, Д.А. ШАТКОВСКИЙ, Т.Н. ШВАЛЮК

ОБ ОЦЕНКЕ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ ЛИНЕЙНЫХ СПИСКОВ ПРИ РЕАЛИЗАЦИИ СИСТЕМ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

Abstract: *The paper is devoted to the problems of memory organization while implementing the computer algebra systems. The alternative approaches to the memory organization based upon the principle "heap" and the nested and chained lists are analyzed. The bounds of effective using the indicated methods are estimated theoretically. The "heap" memory organization is proved to be more effective by the processing speed and the memory usage. The way the effectiveness depends on the used method operation factors and technical resources is shown. By the example of the language Analytic-2000 implementation the way in which the results depend on the garbage collection implementation is described.*
Key words: *symbol objects, objects of arbitrary length, lists, consecutive lists, heap, nested lists, nodal lists, chained lists, texts, algorithmic language, Analytic, memory organization, performance criteria, effectiveness, computer algebra, numerical-analytical methods, analytical methods, formula conversion, artificial intelligence, recognition, garbage collection.*

Анотація: *Стаття присвячена проблемі організації пам'яті при реалізації систем комп'ютерної алгебри. Розглядається альтернативний підхід організації на базі принципу "купа" та на базі гніздових і ланцюгових списків. Теоретично оцінюються межі ефективного застосування вказаних методів. Доведено більшу ефективність організації пам'яті за принципом "купа" по швидкодії й по використанню ресурсів пам'яті. Наведено залежність такої ефективності від параметрів методів та технічних ресурсів, що використовуються. На прикладі реалізації мови Аналітик-2000 показана залежність результатів від реалізації програми очищення пам'яті.*
Ключові слова: *символьні об'єкти, об'єкти довільної довжини, списки, списки послідовні, купа, списки гніздові, списки вузлові списки ланцюгові, тексти, алгоритмічна мова, Аналітик, організація пам'яті, критерій ефективності, ефективність, комп'ютерна алгебра, чисельно-аналітичні методи, аналітичні методи, формульні перетворення, штучний інтелект, розпізнавання, очищення сміття.*

Аннотация: *Статья посвящена проблеме организации памяти при реализации систем компьютерной алгебри. Рассматривается альтернативный подход организации на базе принципа "куча" и на базе гнездовых и цепных списков. Теоретически оцениваются границы эффективного применения указанных методов. Доказана большая эффективность организации памяти по принципу "куча" по быстродействию и по использованию ресурсов памяти. Приведена зависимость такой эффективности от параметров применяемых методов и используемых технических ресурсов. На примере реализации языка Аналитик-2000 показана зависимость результатов от реализации программы очистки памяти.*
Ключевые слова: *символьные объекты, объекты произвольной длины, списки, списки последовательные, куча, списки гнездовые, списки узловые, списки цепные, тексты, алгоритмический язык, Аналитик, организация памяти, критерий эффективности, эффективность, компьютерная алгебра, численно-аналитические методы, аналитические методы, формульные преобразования, искусственный интеллект, распознавание, очистка мусора.*

1. Введение

Развитие ЭВМ, увеличение объемов и удешевление машинной памяти позволили с 50-х годов развернуть работу по автоматизации символьных выкладок. Появились достаточно развитые средства обработки символьной информации такие, как ЛИСП, внутренняя память которого была организована с использованием списковых структур, получивших название цепных списков [1]. Многие достаточно сложные программы символьных преобразований использовали ЛИСП в качестве языка реализации [2, 7]. Но ЛИСП был, в основном, ориентирован на работу с текстами, а ограниченные ресурсы памяти в то время требовали для реализации математических программ, а затем и систем программирования чрезвычайно экономного подхода.

В 60-х годах при реализации универсальной системы программирования АНАЛИТИК, ориентированной на решение задач с применением символьных преобразований, разработчики по соображениям экономии памяти отказались от списковой структуры организации памяти. При этом была выполнена аппаратная реализация многих специфических для математики преобразований и создан транслятор, интерпретирующий язык высокого уровня. Такой подход, особенно на этапах проектирования и предварительного научного обсуждения, послужил поводом для острых дискуссий. В частности, спорным оказался вопрос отказа от списковых структур в организации памяти.

Большой успех ЭВМ МИР (машина для исследовательских и инженерных расчетов) эмпирически решил эту проблему. Авторам неизвестны какие-либо теоретические работы, связанные с этой, как казалось бы, потерявшей остроту задачей. Появление новых масштабных разработок систем компьютерной алгебры (СКА), таких как REDUCE, Mathematica, Maple, реализованных на списковых структурах, показало, что подобного рода исследования не потеряли своей актуальности.

Цель статьи – теоретически оценить границы эффективного применения принципа организации памяти “heap” (“куча“) в СКА. Для этого рассматриваются два критерия: объем занимаемой данными памяти и скорость работы программ.

2. Оценка критериев эффективности

При оценке этих критериев предполагается, что данные в выкладках обычно используются многократно, поэтому их уничтожение недоспустимо.

Согласно [1] далее в выкладках под списковой структурой организации памяти понимается наиболее общий случай – узловой список. При организации памяти узловым списком данные разбиваются на гнезда (последовательно расположенные в памяти ячейки) с фиксированной или произвольной длиной и с произвольным количеством адресов, которые связывают любые части объекта между собой. Так, используемый в ЛИСП цепной список является частным случаем узлового списка, получаемым при использовании одного машинного слова для данных и одного адреса. С помощью цепных списков ЛИСП реализована СКА Reduce [3]. С использованием узловых списков связана реализация СКА Mathematica и Maple [4, 5].

Под “кучей” (heap) в статье понимается определенный [1] линейный список, представляющий собой непрерывный участок памяти, хранящий объект целиком.

Рассматриваемая оценка эффективности по занимаемой памяти определяется формулой

$$\eta_{П} = \frac{V_k}{V_c}, \qquad (1)$$

где $\eta_{П}$ – коэффициент, характеризующий эффективность использования памяти; V_k V_c – объем памяти одного и того же объекта, записанного с использованием линейного списка (“кучи”), и любой другой списковой структуры (цепной, гнездовой или узловой), содержащей адреса переходов между частями списка.

Очевидно, что в этом случае $\eta_{П} < 1$ и организация памяти линейным списком является более экономной. Доказательство этого факта является тривиальным, так как V_k содержит только данные, а V_c, кроме данных, еще и адреса переходов.

Далее покажем, что и для быстродействия также существует достаточно обширная область большей эффективности при организации памяти по типу “куча” по сравнению со списковыми структурами.

Аналогично (1) критерий оценки эффективности линейной и списоковой организации памяти для быстродействия выглядит

$$\eta = \frac{T_k}{T_c}, \tag{2}$$

где η – оценка критерия эффективности быстродействия; T_k – это время работы программы при организации памяти линейным списком; T_c – время работы программы при организации при помощи списковых структур. При $\eta < 1$ эффективность “кучи” выше.

Очевидно, что время работы программы при организации памяти “кучей” состоит из времени счета и времени уборки мусора. Отсюда из (2) получим

$$\eta = \frac{T_k}{T_c} = \frac{T_k^{'} + T_{yk}}{T_c^{'} + T_{yc}}, \tag{3}$$

где $T_k^{'}, T_{yk}$ – время счета и время работы процедуры уборки мусора при организации памяти “кучей”; $T_c^{'}, T_{yc}$ – время счета и время работы процедуры уборки мусора при организации памяти с использованием списковых структур.

Очевидно, что время уборки мусора при использовании списковых структур является сравнительно малым. Поэтому в дальнейших выкладках оно не учитывается. Дальнейшие упрощения также будут производиться только в сторону уменьшения области более высокой эффективности “кучи”.

Приняв $T_{yc} = 0$, получим

$$\eta = \frac{T_k^{'} + T_{yk}}{T_c^{'}}. \tag{4}$$

Положим, что работа алгоритма в основном состоит из последовательности операций чтения данных из памяти, обработки их в регистрах процессора и записи результата в память. Тогда различие в работе одного и того же алгоритма при организации памяти “кучей” и при помощи списковых структур состоит только в необходимости записи адресов елементов списка. Следовательно,

$$T_c^{'} = T_k^{'} + T_a, \tag{5}$$

где T_a – время работы с адресами.

Тогда соотношение (4) примет вид

$$\eta = \frac{T_k^{'} + T_{yk}}{T_k^{'} + T_a}. \quad (6)$$

Очевидно, если все слагаемые числителя и знаменателя положительны, то при вычитании из числителя и знаменателя одного и того же слагаемого соотношение η с 1 в критерии относительной эффективности не меняется.

Отсюда из (6) имеем

$$\eta = \frac{T_{yk}}{T_a}, \quad (7)$$

где числовое значение η изменилось, но сохранилось отношение к 1.

В конце статьи дано конструктивное доказательство существования алгоритма уборки мусора, из которого видно, что время уборки мусора для “кучи” состоит из времени анализа (поиск и упорядочивание адресов участников полезной информации) и времени плотного переписывания полезных данных в младшие адреса памяти. Отсюда из (7) имеем

$$\eta = \frac{T_{анализа} + T_{переписывания}}{T_a}, \quad (8)$$

где $T_{переписывания}$ – время плотного переписывания данных; $T_{анализа}$ – время анализа.

Время работы с адресами между двумя актами уборки мусора :

$$T_a = \frac{p \cdot S_{пам} \cdot T_{пам}}{S_{гнезда}}, \quad (9)$$

где $S_{пам}$ – объем доступной памяти; $T_{пам}$ – время работы с одним элементом памяти; $S_{гнезда}$ – объем гнезда вместе с адресами (в 32-разрядных словах); p – пропорционально среднему числу обращений к адресу между очередными уборками мусора.

Исходя из (8) и (9), получим

$$\eta = S_{гнезда} \frac{T_{анализа} + T_{переписывания}}{p \cdot S_{пам} \cdot T_{пам}}. \quad (10)$$

Время плотного переписывания данных в младшие адреса памяти:

$$T_{переписывания} = 2 \cdot T_{пам} \cdot q \cdot S_{пам}, \quad (11)$$

где q – доля перемещаемой при уборке мусора памяти (обычно $q << 1$).

Из (10) и (11) получаем

$$\eta = S_{гнезда} \frac{T_{анализа} + 2 \cdot T_{пам} \cdot q \cdot S_{пам}}{p \cdot S_{пам} \cdot T_{пам}}. \quad (12)$$

В реализованной в СКА А-2000 процедуре уборки мусора, описание которой приводится ниже, время анализа оценивается [7]

$$T_{анализа} = n \cdot \log(n) \cdot 5 \cdot T_{пам}, \quad (13)$$

где n – число объектов задачи.

Тогда из (12) и (13) получим

$$\eta = S_{гнезда} \frac{n \cdot \log(n) \cdot 5 + q \cdot S_{пам} \cdot 2}{p \cdot S_{пам}}. \quad (14)$$

Проведя несложные эквивалентные преобразования, можно представить формулу (14) в более удобном для анализа виде:

$$\eta = \frac{S_{гнезда}}{p} \left(\frac{5 \cdot n \cdot \log(n)}{S_{пам}} + 2 \cdot q \right). \quad (15)$$

Из (15) видно, что критерий относительной эффективности η прямо пропорционален размеру гнезда и что η монотонно растет с n и q и убывает вместе с p и $S_{пам}$.

Очевидно, что увеличение длины гнезда списковой структуры, начиная с некоторого $S_{гнезда}$, уже практически не приводит к повышению эффективности списка, а η стремится к 1, т. к. это является предельной ситуацией фактического перехода списка в “кучу”. Кроме того, в системе с произвольным размером гнезда в результате дробления гнезд через некоторое время практически невозможно выделить гнездо с большим объемом, при этом η уменьшается.

Далее из (15) имеем, что η зависит от эффективности процедуры уборки мусора. На малую по своему определению величину q влияет в сторону дальнейшего уменьшения тот факт, что при уборке мусора переписываются только объекты с адресами старше адреса первого занятого мусором участка памяти. Описание алгоритма уборки мусора в А-2000, приведенное ниже, показывает малую зависимость коэффициента η от параметра n (15).

Таким образом, величина η, в основном, зависит от объема гнезда $S_{гнезда}$, объема переписываемых данных q и параметра p, который при применении современных методов компьютерной алгебры не бывает очень большим.

Функциональная зависимость η от основных параметров (15) иллюстрируется приведенными ниже графиками, в которых величина технических параметров ПЭВМ соответствует современным уровням компьютерной техники (объем оперативной памяти $S_{пам}$ принят равным 512 МБ = 1,28*10^8 32-разрядных слов).

Второстепенные параметры, от которых величина η зависит мало, варьируется следующим образом: $n = 100$ и 10000; $p = 2,10,100,1000$.

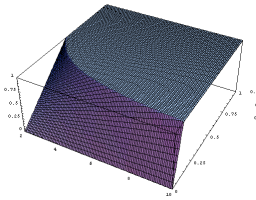

Рис. 1: p =2 n =100

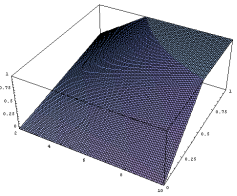

Рис. 2: p =10 n =100

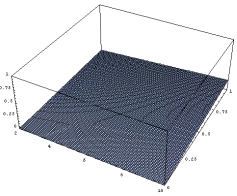

Рис. 3: p =100 n =100

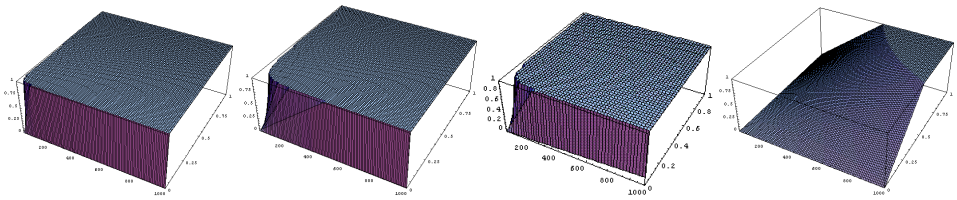

Рис. 4: p =2 n =10000 | Рис. 5: p =10 n =10000 | Рис. 6: p =100 n =10000 | Рис. 7: p =1000 n =10000

Как видно из рис. 1-7, при решении современных задач компьютерной алгебры аналитическими или численно-аналитическими методами большая эффективность организации памяти по методу “кучи” фактически гарантирована. При размерах гнезд до 4 двойных слов “куча” выигрывает безоговорочно. При больших величинах гнезд списковые структуры для некоторых задач могут стать более эффективными. В этом случае результат сильно зависит от параметра p. Для разных задач и методов программирования он может колебаться в значительных пределах. Можно предположить, что диапазон от 2 до 1000 охватывает подавляющее большинство реальных задач компьютерной алгебры. Из графиков также видно, что чем больше параметр p, тем больше гарантированная область эффективности “кучи”. Кроме того, при размерах гнезда больших, чем средний размер объекта, списковые структуры практически представляют собой “кучу”.

Приведенный анализ относительной эффективности организации памяти по принципам “куча” и “список” для полноты решения задачи был проведен в абстрактных терминах (со значениями соответствующих параметров в очень широком диапазоне). В действительности, если при выборе языка реализации в распоряжении разработчиков СКА имеются языки, подобные ЛИСП, с ограниченной вариацией параметров, в частности, параметра $S_{гнезда}$, то это делает выбор между “кучей” и списковыми структурами сильно зависимым от качества и возможности реализации процедуры уборки мусора для “кучи”.

Приведенный ниже алгоритм уборки мусора, реализованный в СКА А-2000, может рассматриваться как конструктивное доказательство возможности создания эффективных процедур такого рода.

Процедура уборки мусора в СКА А-2000 заключается в плотном переписывании данных в младшие адреса памяти, выделенной для системы. Такое уплотнение может осуществляться разными способами. В системах А-2000 уборка мусора состоит из двух этапов:

– упорядочивание адресов занятых участков по возрастанию;

– перемещение занятых блоков в порядке возрастания адресов.

Адреса всех объектов содержатся в соответствующей дескрипторной таблице (каталоге).

Наиболее совершенная процедура упорядочивания этих адресов содержит $n \cdot \log(n)$ шагов [7], на каждом из которых производится обращение к двум адресам памяти, сравнение, обмен и запись в память.

Время чтения и время записи предполагаются одинаковыми. Время сравнения и обмена в сумме меньше, чем одно обращение к памяти, т. к. эти операции осуществляются в регистрах процессора. Поэтому, ухудшая ситуацию для “кучи”, можно принять их сумму равной времени

обращения к памяти. Отсюда время анализа (упорядочивания) описывается формулой (13). Время перемещения полезных данных описано формулой (11).

3. Выводы

1. При реализации языков компьютерной алгебры наиболее экономная организация памяти осуществляется при использовании непрерывных последовательных списков.
2. При реализации языков компьютерной алгебры наиболее эффективной является организация памяти в виде непрерывных последовательных списков (“куча”).
3. Эффективность реализации языка компьютерной алгебры при применении непрерывных последовательных списков сильно зависит от эффективности алгоритма очистки памяти.

СПИСОК ЛИТЕРАТУРЫ

1. Китов А.И. Программирование информационно-логических задач. – М.: Советское радио, 1969. – С. 251–272.
2. Клименко В.П. Основные принципы построения систем и интерпретации языков, проблемно-ориентированных на научные и инженерные задачи // Кибернетика. – 1990. – № 1. – С. 49–56.
3. Hearn A.C. REDUCE. User’s Manual Version 3.4. – Santa Monica: The RAND Corporation, 1991. – 200 p.
4. The Mathematica Book // Wolfram Research – N.Y. – §1.12.2.
5. Документация Maple // http://www.maplesoft.com.
6. http://algolist.manual.ru/sort/knuth3.zip.
7. Ньюэлл А., Шоу Дж., Саймон Г. Эмпирические исследования машины “ЛОГИКА-ТЕОРЕТИКА” (пример изучения эвристик): Сб. “Вычислительные машины и мышление”. – М.: Мир, 1967. – С. 113–144.