Семестровая №2

на тему “Сортировка слиянием”

Выполнил: студент 1 курса ИТИС 11-002

Асфандьяров И.

Проверил: преподаватель практики АиСД

Зиятдинов М.Т.

Алгоритм сортировки слияниями был изобретён венгеро-американским математиком Джоном фон Нейманом в 1945 году. Он является одним из самых быстрых способов сортировки. Слияние - это объединение двух или более упорядоченных массивов в один упорядоченный. Сортировка слияниями является одним из самых простых алгоритмов сортировки (среди быстрых алгоритмов). Особенностью этого алгоритма является то, что он работает с элементами массива преимущественно последовательно, благодаря чему именно этот алгоритм используется при сортировке в системах с различными аппаратными ограничениями (например, при сортировке данных на жестком диске). Кроме того, сортировка слиянием является алгоритмом, который может быть эффективно использован для сортировки таких структур данных, как связанные списки. Данный алгоритм применяется тогда, когда есть возможность использовать для хранения промежуточных результатов память, сравнимую с размером исходного массива. Он построен на принципе «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Далее их решения комбинируются, и получается решение исходной задачи. Процедура слияния требует два отсортированных массива. Заметим, что массив из одного элемента по определению является отсортированным. Введём понятие упорядоченного отрезка или серии - это последовательность элементов, для которых не нарушается условие упорядоченности. Количество элементов данной последовательности называется длиной серии. Серия, состоящая из одного элемента, упорядочена всегда. Если длина серии фиксирована, то слияние называется простым. При естественном слиянии всегда сливаются две наиболее длинных серии. Фаза - это последовательность действий, необходимых для однократной обработки всех элементов. Проход - это наименьший процесс, реализация которого составляет алгоритм сортировки. Двухфазная сортировка - это сортировка, в которой отдельно реализуется две фазы: распределение и слияние. Однофазная сортировка - это сортировка, в которой объединены фазы распределения и слияния в одну. Исходные данные разбиваются на серии, или упорядоченные отрезки, и распределяются на два и более вспомогательных массива. Это распределение идет поочередно: первая серия записывается в первый вспомогательный массив, вторая - во второй и т.д. После того, как произошла запись серии в последний вспомогательный массив, следующая по счёту серия записывается опять в первый вспомогательный массив. После распределения всех серий они объединяются в более длинные упорядоченные отрезки: из каждого вспомогательного массива берется по одной серии, и серии сливаются. Если в каком-то массиве серия заканчиваются, то следующая серия пока не рассматривается. Сформированный более длинный упорядоченный отрезок записывается либо в исходный массив, либо в какой-то из вспомогательных. Далее происходит распределение этих длинных серий во вспомогательные массивы с последующим их слиянием. До тех пор пока все данные не будут упорядочены. В сортировке слияниями выделяют две основные характеристики: Количество вспомогательных массивов. (В случае если вспомогательных массивов два, распределение называется двухпутевым, а вся сортировка - двухпутевым слиянием. ) Количество фаз (шагов, этапов) в реализации сортировки.

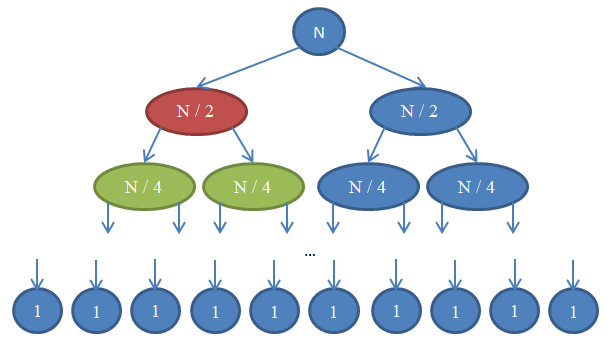
Алгоритм сортировки слияниями:

Шаг 1. Разбить имеющиеся элементы массива на пары и осуществить слияние элементов каждой пары, получив отсортированные цепочки длины 2 (кроме, быть может, одного элемента, для которого не нашлось пары).

Шаг 2. Разбить имеющиеся отсортированные цепочки на пары, и осуществить слияние цепочек каждой пары.

Шаг 3. Если число отсортированных цепочек больше единицы, перейти к шагу 2.

**Сложность:**

****

Глубина рекурсии будет logn, но на каждом уровне в любом случае будут сливаться в сумме n элементов (на втором уровне 2\*(N/2), на третьем 4\*(N/4), это всё отчётливо видно на картинке. То есть 2\*(N/2) будет N, поэтому на каждом уровне в сумме выходит слияние n элементов, просто в более маленьких группах.

Исходя из этого мы можем понять, что n элементов массива будут сливаться logn раз, отсюда и сложность nlogn

Достоинства:

* Работает даже на структурах данных последовательного доступа.
* Хорошо сочетается с подкачкой и кэшированием памяти.
* Неплохо работает в параллельном варианте: легко разбить задачи между процессорами поровну, но трудно сделать так, чтобы другие процессоры взяли на себя работу, в случае если один процессор задержится.
* Не имеет «трудных» входных данных.
* Устойчивая - сохраняет порядок равных элементов (принадлежащих одному классу эквивалентности по сравнению).

Недостатки:

* На «почти отсортированных» массивах работает столь же долго, как на хаотичных. Существует вариант сортировки слиянием, который работает быстрее на частично отсортированных данных, но он требует дополнительной памяти, в дополнении ко временному буферу, который используется непосредственно для сортировки.
* Требует дополнительной памяти по размеру исходного массива.

**Список литературы:**

* https://ru.stackoverflow.com/questions/443313/cортировка-слияниeм-и-сложность
* https://ru.wikipedia.org/wiki/Сортировка\_слиянием

**Код программы:**

1. **#include <stdio.h>**
2. **#include <stdlib.h>**
3. **#include <ctime>**
4. **// Функция сортировки двухпутевым слиянием**
5. **void merge(int \*a, int n)**
6. **{**
7. **int mid = n / 2; // находим середину сортируемой последовательности**
8. **if (n % 2 == 1)**
9. **mid++;**
10. **int h = 1; // шаг**
11. **// выделяем память под формируемую последовательность**
12. **int \*c = (int\*)malloc(n \* sizeof(int));**
13. **int step;**
14. **while (h < n)**
15. **{**
16. **step = h;**
17. **int i = 0; // индекс первого пути**
18. **int j = mid; // индекс второго пути**
19. **int k = 0; // индекс элемента в результирующей последовательности**
20. **while (step <= mid)**
21. **{**
22. **while ((i < step) && (j < n) && (j < (mid + step)))**
23. **{ // пока не дошли до конца пути**
24. **// заполняем следующий элемент формируемой последовательности**
25. **// меньшим из двух просматриваемых**
26. **if (a[i] < a[j])**
27. **{**
28. **c[k] = a[i];**
29. **i++; k++;**
30. **}**
31. **else {**
32. **c[k] = a[j];**
33. **j++; k++;**
34. **}**
35. **}**
36. **while (i < step)**
37. **{ // переписываем оставшиеся элементы первого пути (если второй кончился раньше)**
38. **c[k] = a[i];**
39. **i++; k++;**
40. **}**
41. **while ((j < (mid + step)) && (j<n))**
42. **{ // переписываем оставшиеся элементы второго пути (если первый кончился раньше)**
43. **c[k] = a[j];**
44. **j++; k++;**
45. **}**
46. **step = step + h; // переходим к следующему этапу**
47. **}**
48. **h = h \* 2;**
49. **// Переносим упорядоченную последовательность (промежуточный вариант) в исходный массив**
50. **for (i = 0; i<n; i++)**
51. **a[i] = c[i];**
52. **}**
53. **}**
54. **int main()**
55. **{**
56. **int a[8];**
57. **//введём функцию, чтобы рандомность чисел привязывалась ко времени, это даст нам на каждый запуск новые рандомные числа**
58. **srand (time(NULL));**
59. **// Заполнение массива случайными числами**
60. **for (int i = 0; i<8; i++)**
61. **a[i] = rand() % 20 + 1;**
62. **// Вывод элементов массива до сортировки**
63. **for (int i = 0; i<8; i++)**
64. **printf("%d ", a[i]);**
65. **printf("\n");**
66. **merge(a, 8); // вызов функции сортировки**
67. **// Вывод элементов массива после сортировки**
68. **for (int i = 0; i<8; i++)**
69. **printf("%d ", a[i]);**
70. **printf("\n");**
71. **getchar();**
72. **return 0;**
73. **}**

Входных данных нет, все данные генерируются автоматически каждый запуск программы.

Ссылка на репозиторий: https://github.com/demonrussich/AiSD