Отчет о выполнении задания 1  
«Алгоритмы сортировки»

**Чедуров Александр Николаевич**

Группа ИС-641

# Описание алгоритмов

В задании 1 требовалось реализовать и исследовать эффективность трех алгоритмов сортировки: Counting Sort, Bubble Sort и MergeSort. Все алгоритмы реализованы на языке C под операционной системой GNU/Linux.

***Counting Sort*** - алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов. Применение сортировки подсчётом целесообразно лишь тогда, когда сортируемые числа имеют (или их можно отобразить) диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством, например, миллион натуральных чисел меньших 1000.

Код на С:

|  |
| --- |
| int countingsort (int mas[], int n)  {  int i, min, max;  min = mas[0];  max = mas[0];  for (i = 1; i < n; i++) {  if ( mas[i] < min ) {  min = mas[i];  } else if ( mas[i] > max ) {  max = mas[i];  }  }  int j, z;  int range = max - min + 1;  int cow[range];  for (i = 0; i < range; i++) {  cow[i] = 0;  }  for (i = 0; i < n; i++) {  cow[mas[i] - min]++;  }  for (i = min, z = 0; i <= max; i++) {  for (j = 0; j < cow[i - min]; j++) {  mas[z++] = i;  }  }    return 0;  } |

***Bubble sort*** - Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде, отсюда и название алгоритма).

Код на С:

|  |
| --- |
| int bubblesort (int mas[], int n)  {  int i, temp, s = 1;  while (s) {  s = 0;  for (i = 1; i < n; i++) {  if (mas[i] < mas[i - 1]) {  temp = mas[i];  mas[i] = mas[i - 1];  mas[i - 1] = temp;  s = 1;  }  }  }    return 0;  } |

***Merge Sort*** - алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «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. Наконец, их решения комбинируются, и получается решение исходной задачи.

Код на С:

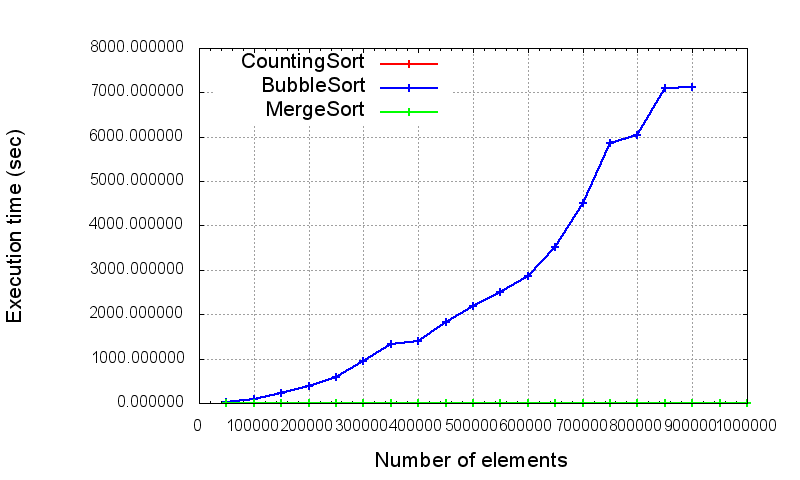
|  |
| --- |
| int mergehelp (int mas[], int mashelp[], int l, int m, int r)  {  int i, left, right;  for (i = l; i < r + 1; i++) {  mashelp[i] = mas[i];  }  left = l;  right = m + 1;  i = l;  while ((left <= m) && (right <= r)) {  if (mashelp[left] <= mashelp[right]) {  mas[i] = mashelp[left];  left++;  } else {  mas[i] = mashelp[right];  right++;  }  i++;  }  while (left <= m) {  mas[i] = mashelp[left];  left++;  i++;  }  while (right <= r) {  mas[i] = mashelp[right];  right++;  i++;  }    return 0;  }  int mergesort (int mas[], int mashelp[], int l, int r)  {  int m;  if (l < r) {  m = (l + r) / 2;  mergesort (mas, mashelp, l, m);  mergesort (mas, mashelp, m + 1, r);  mergehelp (mas, mashelp, l, m, r);  }    return 0;  } |

# Организация экспериментов

* Эксперименты проводились на ноутбуке Lenovo G5045   
  (AMD A8-6410 APU with AMD Radeon R5 Graphics 2.00GHz, RAM 4 Gb
* Операционная система Ubuntu x86\_64 (компилятор gcc 4.8.2)
* Ключи компиляции программы (см. README): -Wall -O2

# Результаты экспериментов

На рис. показаны графики зависимости кол-ва элементов в массиве от времени:



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Counting** | **Bubble** | **Merge** |
| **Количество элементов в массиве** | **Время выполнения алгоритма, с** | | |
| 50000 | 0.003000 | 24.755508 | 0.017001 |
| 100000 | 0.004008 | 94.859106 | 0.036006 |
| 150000 | 0.004999 | 218.771032 | 0.056505 |
| 200000 | 0.005000 | 375.784826 | 0.074510 |
| 250000 | 0.008002 | 585.399488 | 0.089996 |
| 300000 | 0.009012 | 954.078733 | 0.109496 |
| 350000 | 0.009987 | 1324.143048 | 0.132530 |
| 400000 | 0.009999 | 1388.028133 | 0.148503 |
| 450000 | 0.011510 | 1816.459826 | 0.168022 |
| 500000 | 0.013004 | 2175.689448 | 0.209040 |
| 550000 | 0.014497 | 2492.418979 | 0.231045 |
| 600000 | 0.015018 | 2866.805056 | 0.247552 |
| 650000 | 0.015022 | 3522.962840 | 0.250530 |
| 700000 | 0.016985 | 4512.010687 | 0.270080 |
| 750000 | 0.018002 | 5868.203043 | 0.292518 |
| 800000 | 0.020046 | 6039.258765 | 0.308553 |
| 850000 | 0.022485 | 7105.847892 | 0.331543 |
| 900000 | 0.022517 | 7112.422818 | 0.354543 |
| 950000 | 0.023002 |  | 0.373561 |
| 1000000 | 0.023504 |  | 0.395549 |

Из результатов экспериментов видно, что алгоритм ***Counting Sort*** работает быстрее алгоритма ***Merge Sort***, в свою очередь алгоритм ***Bubble Sort*** работает намного медленнее обоих алгоритмов. Это объясняется тем, что алгоритмы имеют различную вычислительную сложность, так:

***Counting Sort*** в худшем случае O(n + k), в лучшем O(n);

***Bubble Sort*** в худшем случае O(n2), в лучшем O(n);

***Merge Sort*** вычислительную сложность O(n\*log(n)).

***Контрольные вопросы:***

1. *Что такое вычислительная сложность алгоритма (computational complexity)?*

Ответ: Кол-во элементарных операций.

1. *Что означают записи f(n) = O(g(n)), f(n) = Θ(g(n)), f(n) = Ω(g(n))?*

Ответ: Оценка алгоритма сортировки, задающий верхний предел времени, который тратит функция на сортировку.

1. *Какой алгоритм сортировки называется устойчивым (stable)?*

Ответ: Алгоритм, не меняющий местами элементы с одинаковыми значениями.

1. *Какой алгоритм сортировки называется сортировкой «на месте» (in place)?*

Ответ: Алгоритм, не использующий еще один массив данных.

1. *Какая вычислительная сложность в худшем случае у алгоритмов, которые Вы реализовали?*

Ответ: Counting Sort в худшем случае O (n + k);

Bubble Sort вычислительную сложность O (n2);

Merge Sort вычислительную сложность O(n\*log(n)).

1. *Объясните поведение кривых на графиках, которые вы построили. Согласуются ли экспериментальные результаты с оценкой вычислительной сложности алгоритмов?*

Ответ: При увеличении числа n (кол-во элементов) графики возрастают. Counting возрастает, примерно, под углом 45 градусов, угол возрастания Bubble увеличивается в геометрической прогрессии, а у Merge угол возрастания немного больше, чем у Counting т.к. у них разная вычислительная сложность.

1. *Какие алгоритмы сортировки с вычислительной сложностью O(nlogn) для худшего случая вам известны?*

Ответ: Heap Sort и Merge Sort.

1. *Известны ли вам алгоритмы сортировки, работающие быстрее O(nlogn) для худшего случая?*

Ответ: Radix Sort.