Сортировка массивов

Основы программирования

Сортировка массивов

- Сортировка – это упорядочивание набора однотипных данных по возрастанию или убыванию.
- Ключ сортировки – это часть данных, определяющая порядок элементов.

Оценка алгоритмов сортировки

- Существует множество различных алгоритмов сортировки. Все они имеют свои положительные и отрицательные стороны. Общие критерии оценки алгоритмов сортировки:
- Скорость работы алгоритма сортировки. Она непосредственно связана с количеством сравнений и количеством обменов, происходящих во время сортировки, причем обмены занимают больше времени.

Оценка алгоритмов сортировки

- Сравнение происходит тогда, когда один элемент массива сравнивается с другим; обмен происходит тогда, когда два элемента меняются местами.
- Время работы в лучшем и худшем случаях. Оно имеет значение при анализе выполнения алгоритма, если одна из краевых ситуаций будет встречаться довольно часто.

Оценка алгоритмов сортировки

- **Поведение алгоритма сортировки. Поведение алгоритма сортировки называется естественным, если время сортировки минимально для уже упорядоченного списка элементов, увеличивается по мере возрастания степени неупорядоченности списка и максимально, когда элементы списка расположены в обратном порядке. Объем работы алгоритма оценивается количеством производимых сравнений и обменов.**

Оценка алгоритмов сортировки

- Различные сортировки массивов отличаются по быстродействию. Существуют простые методы сортировок, которые требуют порядка n*n сравнений, где n – количество элементов массива и быстрые сортировки, которые требуют порядка n*ln(n) сравнений. Простые методы удобны для объяснения принципов сортировок, т.к. имеют простые и короткие алгоритмы. Усложненные методы требуют меньшего числа операций, но сами операции более сложные, поэтому для небольших массивов простые методы более эффективны.

Оценка алгоритмов сортировки

- Простые методы сортировки можно разделить на три основные категории:
- сортировка методом "пузырька" (простого обмена);
- сортировка методом простого выбора (простой перебор);
- сортировка методом простого включения (сдвиг-вставка, вставками, вставка и сдвиг).

Сортировка методом "пузырька" (простого обмена)

- Алгоритм попарного сравнения элементов массива.
- Алгоритм состоит в повторяющихся проходах по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов.

Сортировка методом "пузырька" (простого обмена)

- Проходы по массиву повторяются до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает – массив отсортирован. При проходе алгоритма элемент, стоящий не на своём месте, "всплывает" до нужной позиции

Сортировка методом "пузырька" (простого обмена)

- Рассмотрим идею метода на примере. Отсортируем по возрастанию массив из 5 элементов: 5 4 8 2 9.
- Длина текущей части массива – n-k+1, где
- k - номер просмотра,
- i – номер первого элемента проверяемой пары,
- номер последней пары – n-k.

Сортировка методом "пузырька" (простого обмена)

- Первый просмотр: рассматривается весь массив.
- i=1 5 4 8 2 9
- > меняем
- i=2 4 5 8 2 9
- < не меняем
- i=3 4 5 8 2 9
- > меняем
- i=4 4 5 2 8 9
- < не меняем
- 9 находится на своем месте.

Сортировка методом "пузырька" (простого обмена)

- Второй просмотр: рассматриваем часть массива с первого до предпоследнего элемента.
- i=1 4 5 2 8 9
- < не меняем
- i=2 4 5 2 8 9
- > меняем
- i=3 4 2 5 8 9
- < не меняем
- 8 – на своем месте

Сортировка методом "пузырька" (простого обмена)

- Третий просмотр: рассматриваемая часть массива содержит три первых элемента.
- i=1 4 2 5 8 9
- > меняем
- i=2 2 4 5 8 9
- < не меняем
- 5 – на своем месте.

Сортировка методом "пузырька" (простого обмена)

- Четвертый просмотр: рассматриваем последнюю пару элементов.
- i=1 2 4 5 8 9
- < не меняем
- 4 – на своем месте.
- Наименьший элемент – 2 оказывается на первом месте. Количество просмотров элементов массива равно N-1.

Сортировка методом "пузырька" (простого обмена)

- Итак, наш массив отсортирован. Этот метод также называют методом «пузырька». Название это происходит от образной интерпретации, при которой в процессе выполнения сортировки более «легкие» элементы (элементы с заданным свойством) мало-помалу всплывают на «поверхность».

Сортировка методом "пузырька" (простого обмена)

```
//Описание функции сортировки методом "пузырька"
void Sort_Obmen (int n, int A[])
{
  int k,i,buf;
  for ( k=0; k<n-1; k++)
    for ( i=0; i<n-k-1; i++)
```

Сортировка методом "пузырька" (простого обмена)

```
if (A[i]>A[i+1])
{
    buf=A[i];
    A[i]=A[i+1];
    A[i+1]=buf;
   }
}
```

Сортировка простым выбором

- Пусть исходный массив A состоит из 10 элементов:
- 5 13 7 9 1 8 16 4 10 2.
- После сортировки массив должен выглядеть так:
- 1 2 4 5 7 8 9 10 13 16.

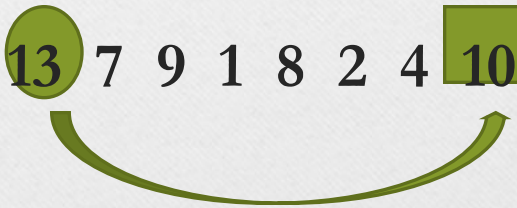
Сортировка простым выбором

- 1-й шаг. Рассмотрим весь массив и найдем в нем максимальный элемент – 16 (стоит на седьмом месте), поменяем его местами с последним элементом – с числом 2.

- 

Сортировка простым выбором

- 2-й шаг. Рассмотрим часть массива – с первого до девятого элемента. Максимальный элемент этой части – 13, стоящий на втором месте. Поменяем его местами с последним элементом этой части – с числом 10.
- 5 13 7 9 1 8 2 4 10 16



- Отсортированная часть массива состоит теперь уже из двух элементов.

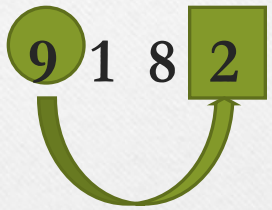
Сортировка простым выбором

- 3-й шаг. Уменьшим рассматриваемую часть массива на один элемент. Здесь нужно поменять местами второй элемент (его значение – 10) и последний элемент этой части – число 4.
- 5 10 7 9 1 8 2 4 13 16

В отсортированной части массива – 3 элемента.

Сортировка простым выбором

- 4-й шаг.
- 5 4 7 9 1 8 2 10 13 16



- 5-й шаг. Максимальный элемент этой части массива является последним в ней, поэтому его нужно оставить на старом месте.
- 5 4 7 2 1 8 9 10 13 16
-

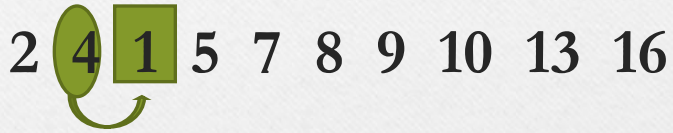
Сортировка простым выбором

- 6-й шаг.

- 7-й шаг.

Сортировка простым выбором

- 8-й шаг.



- 9-й шаг.

Сортировка простым выбором

- **Фрагмент программной реализации:**
- **void vibor(int A[],int n)**
- **{**
- **int i, j, k;**
- **int item;**
- **for (i = 0; i < n-1; i++)**
- **{ item = A[i];**

Сортировка простым выбором

- k = i;
- for (j = i+1; j < n; j ++)
- {
- if (A[j] >= item)
- {
- k = j;

Сортировка простым выбором

- item = A[k];
- }
- }
- A[k] = A[i];
- A[i] = item;
- }
- }

Сортировка простыми вставками

- Сортировка этим методом производится последовательно шаг за шагом. На i-ом шаге считается, что часть массива, содержащая первые i-1 элементов, уже упорядочена, то есть $A[1] \le A[2] \le \ldots \le A[i-1]$. Далее берется i-й элемент, и для него подбирается место в отсортированной части массива, такое, что после его вставки упорядоченность не нарушается, то есть требуется найти такое j ($0 \le j \le i-1$), что $A[j] \le A[i] < A[j+1]$.

Сортировка простыми вставками

- Затем выполняется вставка элемента A[i] на место j. На каждом шаге отсортированная часть массива увеличивается. Для выполнения полной сортировки потребуется выполнить N-1 шаг.

Сортировка простыми вставками

- Пусть требуется отсортировать массив из 10 элементов по возрастанию.
- 1-й шаг.
- 13 6 8 11 3 1 5 9 15 7

Рассматриваем часть массива из одного элемента 13. Вставляем второй элемент массива 6 так, чтобы упорядоченность сохранилась. Так как 6<13, записываем 6 на первое место. Отсортированная часть массива содержит два элемента (6 13).

Сортировка простыми вставками

- 2-й шаг.
- 6 13 8 11 3 1 5 9 15 7
- Возьмем третий элемент массива 8 и подберем для него место в упорядоченной части массива. 8>6 и 8<13, следовательно, его место второе.
- Возьмем третий элемент массива 8 и подберем для него место в упорядоченной части массива. 8>6 и 8<13, следовательно, его место второе.

Сортировка простыми вставками

- 3-й шаг.
- 6 8 13 11 3 1 5 9 15 7
- Следующий элемент – 11. Он записывается в упорядоченную часть массива на третье место, так как 11>8, но 11<13. Продолжаем аналогичным образом пока массив не оказывается отсортированным полностью.

Сортировка простыми вставками

- void vstavka (int A[],int n)
- {
- int i;
- for (i=0; i<n; i++)
- {
- int w=A[i];
- int j=i-1;

Сортировка простыми вставками

- while((j>=0)&&(w<A[j]))
- {
- A[j+1]=A[j];
- j--;
- }
- A[j+1]=w;
- }
- }

Быстрая сортировка Хоара

- Метод предложен Ч. Э. Р. Хоаром в 1962 году. Эффективность метода достаточно высокая, кроме специально подобранных данных, поэтому автор назвал его «быстрой сортировкой».
- Идея метода. В исходном массиве A выбирается некоторый элемент X (барьерный элемент). Нашей целью является запись X «на свое место» в массиве,

Быстрая сортировка Хоара

- пусть это будет место k, такое, что слева от X были элементы массива, меньшие или равные X, а справа – элементы массива, большие X, т. е. массив A будет иметь вид: (A[1], A[2], …, A[k-1]), A[k] (X), (A[k+1], …, A[n]).
- В результате элемент A[k] находится на своем месте и исходный массив A разделен на две неупорядоченные части, барьером между которыми является элемент A[k].

Быстрая сортировка Хоара

- Дальнейшие действия состоят в независимой сортировке полученных частей по той же логике до тех пор, пока не останутся части массива, состоящие из одного элемента, то есть пока не будет отсортирован весь массив.

Быстрая сортировка Хоара

- Пример. Исходный массив состоит из 8 элементов:
- 8 12 3 7 19 11 4 16.
- В качестве барьерного элемента возьмем средний элемент массива (7). Произведя необходимые перестановки, получим:
- (4 3) 7 (12 19 11 8 16), элемент 7 находится на своем месте.

Быстрая сортировка Хоара

- Продолжаем сортировку.
-
- Левая часть: (3) 4 7 (12 19 11 8 16)
- 3 4 7 (12 19 11 8 16)

Быстрая сортировка Хоара

- Правая часть:
- 3 4 7 (8) 11 (19 12 16)
- 3 4 7 8 11 (19 12 16)
- 3 4 7 8 11 12 (19 16)
- 3 4 7 8 11 12 (16) 19
- 3 4 7 8 11 12 16 19

Быстрая сортировка Хоара

- В приведенном алгоритме используется рекурсивная схема реализации, параметрами которой являются нижняя и верхняя границы изменения индексов сортируемой части исходного массива.

Быстрая сортировка Хоара

- Фрагмент программной реализации:
- void hoar_r(int A[],int m, int t)
- {
- int i=m;
- int j=t;
- int x=A[(m+t)/2];
- while (i<=j)

Быстрая сортировка Хоара

- if (A[i]<x) i++;
- else if (A[j]>x) j--;
- else {
- int z=A[i];
- A[i]=A[j];
- A[j]=z;
- i++;

Быстрая сортировка Хоара

- j--;
- }
- if (m<j) hoar_r(A,m,j);
- if (i<t) hoar_r(A,i,t);
- }