EcoLab

Создание компонента, реализующий алгоритм Гномьей сортировки

Выполнил:

Веряскин Данила Валерьевич

21ПИ1

Содержание:

1. [Алгоритм](#_1qwev9tmrgb5)
2. [Оценка сложности](#_x9ruumfimhx9)
3. [Улучшения](#_7rja42jccoi4)
4. [Результаты тестирования](#_j9n934ctl09l)

## Алгоритм

Гномья сортировка (англ. Gnome Sort) - алгоритм сортировки, также известный как “глупая сортировка”, похожий на сортировку вставками, но, в отличие от последней, перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком. Название происходит от предполагаемого поведения садовых гномов при сортировке линии садовых горшков.

Гномья сортировка выполняет по меньшей мере столько же сравнений, сколько и сортировка вставками, и имеет те же асимптотические характеристики времени выполнения. Алгоритм концептуально простой и не требует вложенных циклов. Среднее время выполнения равно O(n²), но стремится к O(n), если список изначально почти отсортирован.

Алгоритм находит первое место, где два соседних элемента стоят в неправильном порядке, и меняет их местами. Он пользуется тем фактом, что обмен может породить новую пару, стоящую в неправильном порядке, только до или после переставленных элементов. Он не допускает, что элементы после текущей позиции отсортированы, таким образом, нужно только проверить позицию до переставляемых элементов.

***Псевдокод:***

| **procedure** gnomeSort(a[]):  pos := 0  **while** pos < length(a):  **if** (pos == 0 **or** a[pos] >= a[pos-1]):  pos := pos + 1  **else**:  swap a[pos] **and** a[pos-1]  pos := pos - 1 |
| --- |

## Оценка сложности

Гномья сортировка проходит через весь массив, сравнивая каждый элемент с его предыдущим. Если элементы находятся в неправильном порядке, они обмениваются местами, и указатель текущей позиции сдвигается на одну позицию назад. Этот процесс продолжается до тех пор, пока указатель не достигнет конца массива.

***Лучший случай:***

В лучшем случае гномья сортировка будет иметь сложность O(n). Это происходит, когда на вход алгоритму подается уже отсортированный массив. В таком случае алгоритм пройдеться линейно по всему массиву, не встретив элементов которые следует поменять местами и завершит свою работу.

***Средний случай:***

В среднем случае гномья сортировка будет иметь сложность O(n²). Это произойдет, когда массив содержит случайные данные и требует множества обменов между элементами для правильной сортировки. Количество операций сравнения и обмена в среднем будет пропорционально квадрату размера массива.

***Худший случай:***

В худшем случае гномья сортировка также будет иметь сложность O(n²). Это происходит, когда массив отсортирован в обратном порядке. В таком случае каждый элемент будет сравниваться и обмениваться с предыдущим до начала массива, что приводит к квадратичной сложности.

Таким образом, гномья сортировка имеет квадратичную сложность в обоих среднем и худшем случае, что делает ее неэффективной для больших массивов данных. Однако, для небольших массивов или массивов, в которых небольшое количество элементов находится не на своем месте, она может быть быстрой и простой в реализации.

## Улучшения

В результате оптимизации гномья сортировка трансформируется в сортировку вставками. Каждый раз, когда «гном» наталкивается на новый номер, все значения слева от «гнома» уже отсортированы.

***Псевдокод сортировки вставками:***

| **procedure** insertionSort(a[]):  for i := 2 to n do:  x := a[i]  j = i  **while** j > 1 and a[j - 1] > x:  a[j] = a[j - 1]  j = j - 1  a[j] = x |
| --- |

## Результаты тестирования

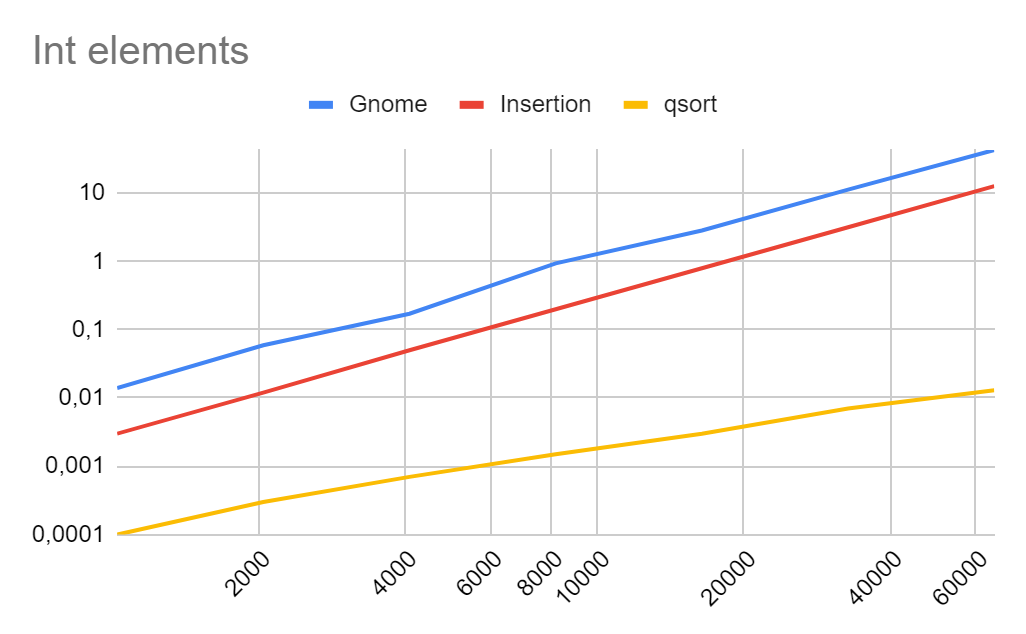
Тестирование реализованного компонента производилось на входных данных различных типов и размеров.  
 Тестируемые типы: int, float, double, char.

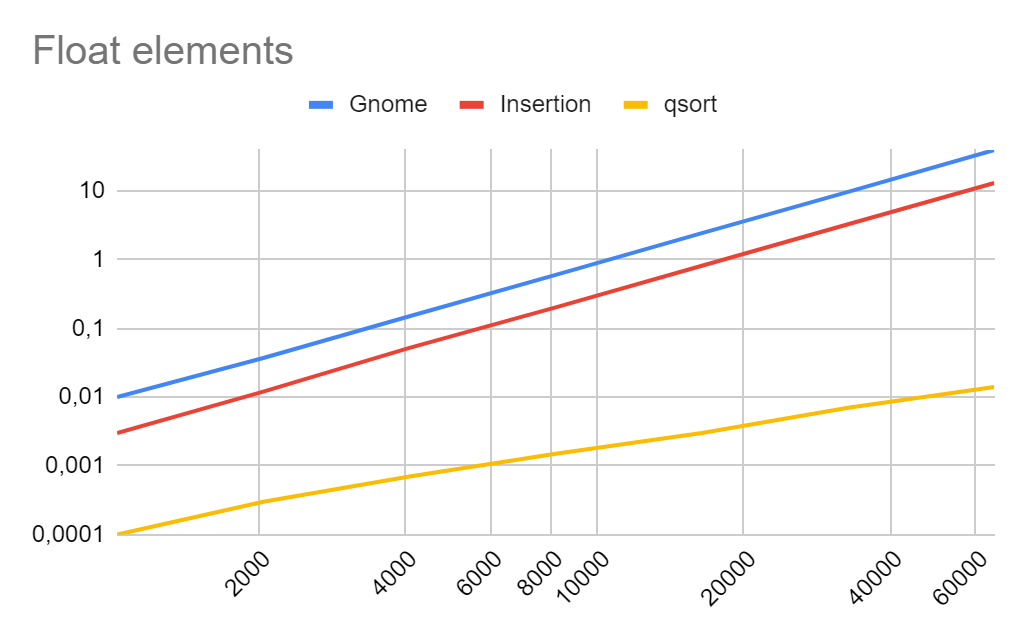
Тестируемые размеры: от 210 до 216 включительно с увеличением в 2 раза на каждом шаге.

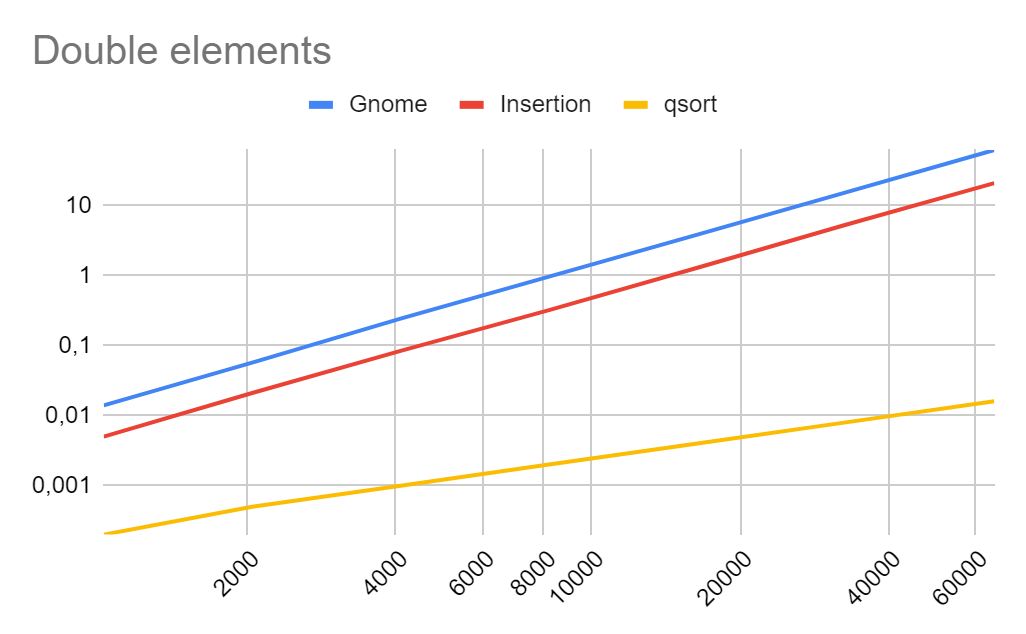
Во всех тестах массивы заполнялись псевдослучайными элементами с помощью функция rand().

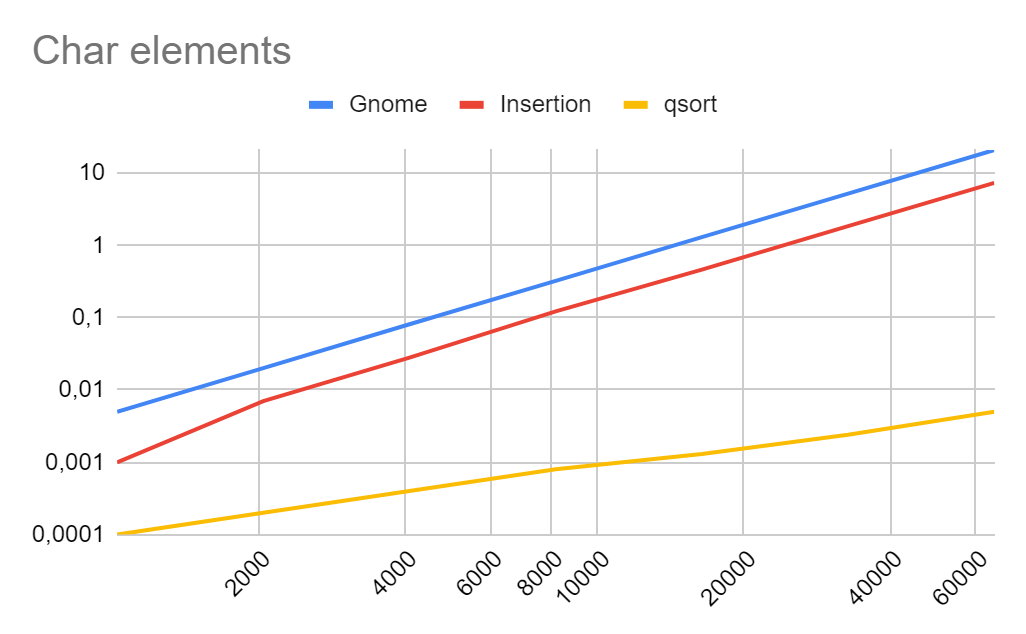
Для сравнения, время работы также измерялось для добавленного компонента реализующего сортировку вставками и библиотечной функций qsort.

Результаты тестирования приведены ниже в виде графиков.









Исходя из полученных результатов можно отметить, что гномья сортировка хуже сортировки вставками и быстрой сортировки во всех тестовых случаев. На графиках видно, что сортировка вставками действительно является улучшенной версией гномьей сортировки, так как разница во времени на всех тестах пропорциональна.

Наиболее быстро все сортировки работают с элементами типа char, а наиболее долго с double. Это можно объяснить размерами типов элементов в памяти. В случае с double это 8 байт, а в случае с char это 1 байт.

Также из графиков можно заметить, что время работы qsort растет более плавно. Это происходить из-за того, что библиотечная быстрая сортировка имеет лучшую сложность O(nlogn), а две других исследуемых сортировки O(n²).