Лабораторная работа 4. Усовершенствованные методы сортировки

Дисциплина : Численные методы и программирование

лектор

Шустрова Марина Леонидовна

Цель работы:

Приобрести понимание принципов работы усовершенствованных методов сортировки, научиться их реализовывать на практике

Задание:

- 1)Изучить теоретический материал,
- 2) Решить предложенные задачи в соответствии со своим вариантом:

  один из методов 1-3 на выбор

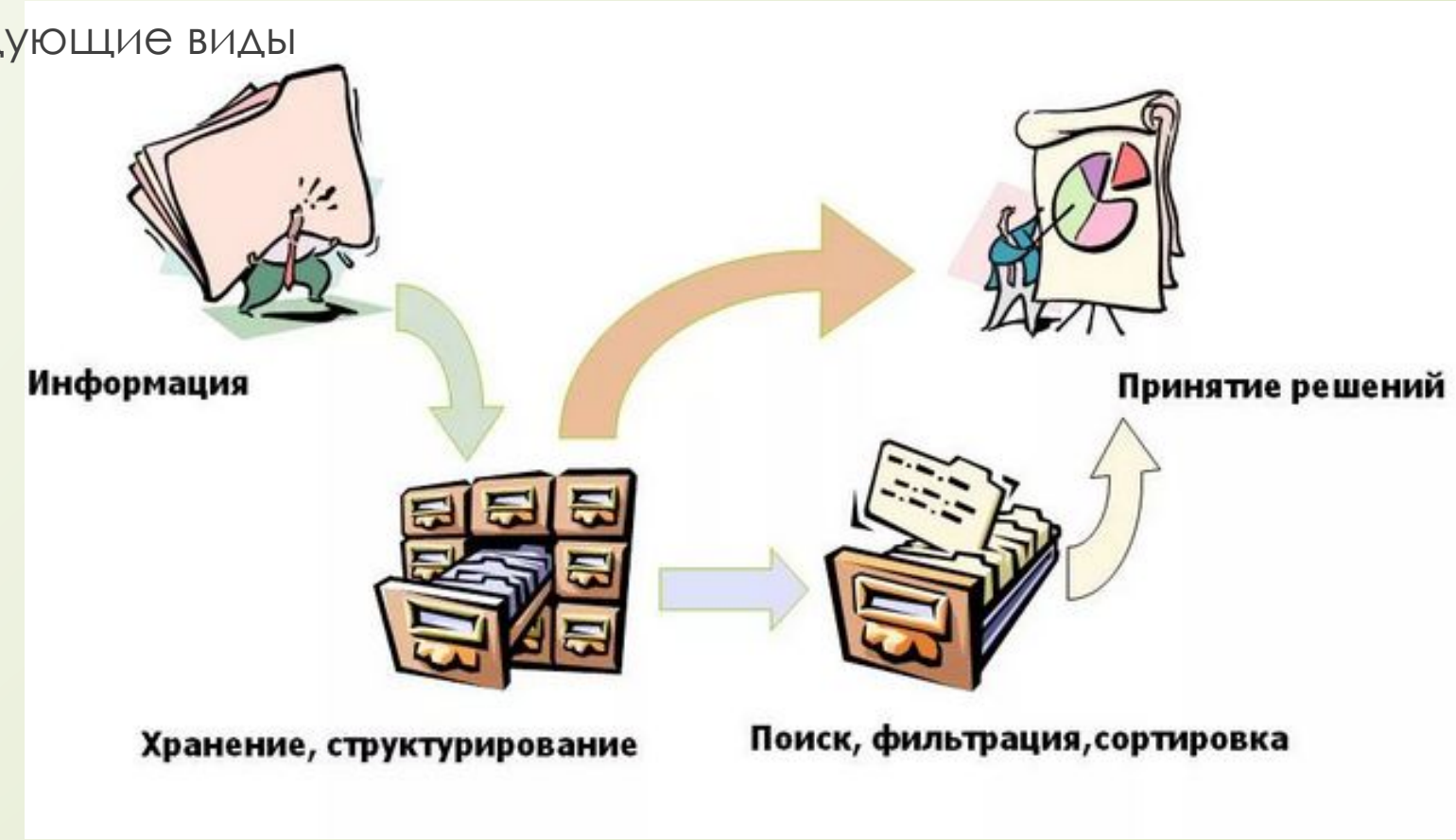
  один из методов 4-6 на выбор
- 3) Заполнить сравнительную таблицу эффективности методов
- 4)Сформировать отчет по лабораторной работе

Сортировка

Сортировка – расположение информации в определенном порядке (упорядочивание)

Чаще всего используются следующие виды сортировки:

- по алфавиту
- по номеру
- в хронологическом порядке

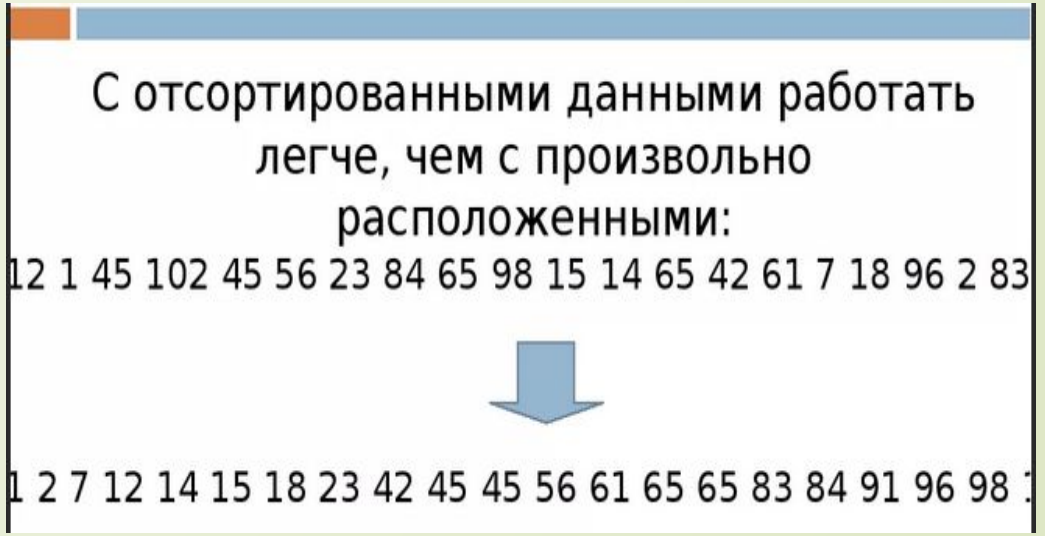


Алгоритм сортировки

- **Алгоритм сортировки** – это алгоритм для упорядочивания элементов в списке.
- В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется **ключом** сортировки.

На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Мы будем рассматривать алгоритмы сортировок на массивах числовых данных



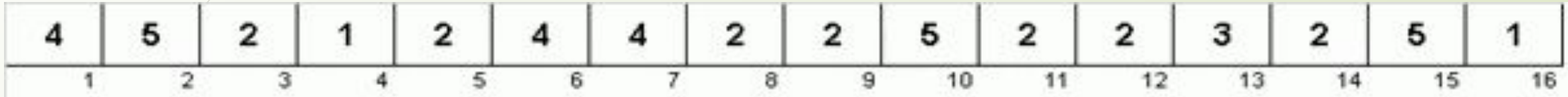
Алгоритмы сортировки

В настоящее время разработано великое множество различных алгоритмов сортировки. В данном курсе мы рассмотрим некоторые из них. Сейчас нас интересуют простые методы (слева).

- **Мы уже попробовали реализовать некоторые простые методы сортировки:**
  - Сортировка пузырьком
    - Шейкерная сортировка
    - Сортировка расческой
    - Сортировка чётно-нечётная
  - Простая сортировка
    - Сортировка вставками
    - Сортировка выбором

- **Теперь попробуем реализовать их усовершенствованные варианты**
  - Бинго-сортировка
  - Двойная сортировка выбором
  - Гномья сортировка
  - Быстрая сортировка
  - Сортировка слиянием
  - Timsort

1. Модификации сортировка выбором: Бинго - сортировка



В неупорядоченной части запоминается не только максимальный элемент, но и определяется максимум для следующей итерации. Это позволяет при повторяющихся максимумах не искать их заново каждый раз, а ставить на своё место сразу как только этот максимум в очередной раз встретили в массиве.

Алгоритмическая сложность осталась та же. Но если массив состоит из повторяющихся чисел, то бинго-сортировка справится в десятки раз быстрее, чем обычная сортировка выбором.

01:23

Задание1.1:

- Проанализировав алгоритм Бинго-сортировки, составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 1.2

- Составьте программу, реализующую метод Бинго-сортировки.

  !!!Массивы берем те же, что и в прошлой работе

- Сколько итераций потребовалось вам для сортировки вашего массива?

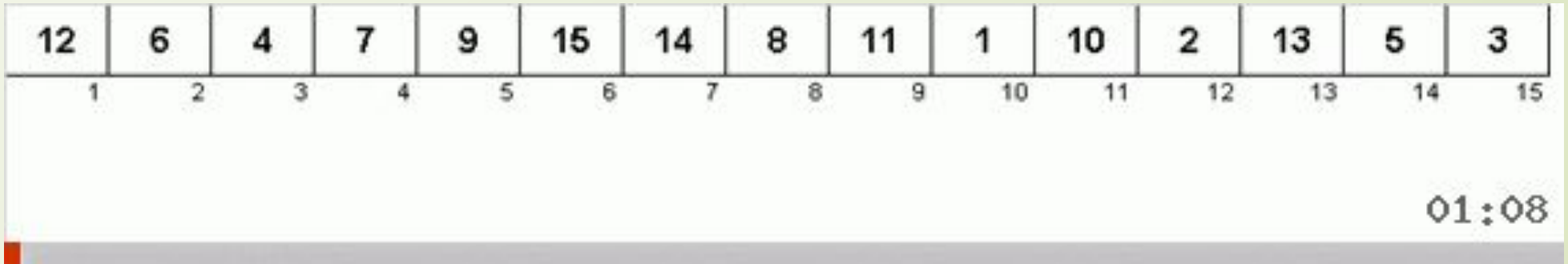
  Ответ на вопрос, листинг программы и поитерационный результат расчета занесите в отчет по лабораторной работе

2. Модификации сортировки выбором: двухсторонняя сортировка выбором **Double selection sort**

Проходя по неотсортированной части массива, мы кроме максимума также попутно находим и минимум.

Минимум ставим на первое место, максимум на последнее. Таким образом, неотсортированная часть при каждой итерации уменьшается сразу на два элемента.

На первый взгляд кажется, что это ускоряет алгоритм в 2 раза — после каждого прохода неотсортированный подмассив уменьшается не с одной, а сразу с двух сторон. Но при этом в 2 раза увеличилось количество сравнений, а число свопов осталось неизменным. Двойной выбор лишь незначительно увеличивает скорость алгоритма, а на некоторых языках даже почему-то работает медленнее.



Задание 2.1:

- Проанализировав алгоритм двойной сортировки выбором, составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 2.2

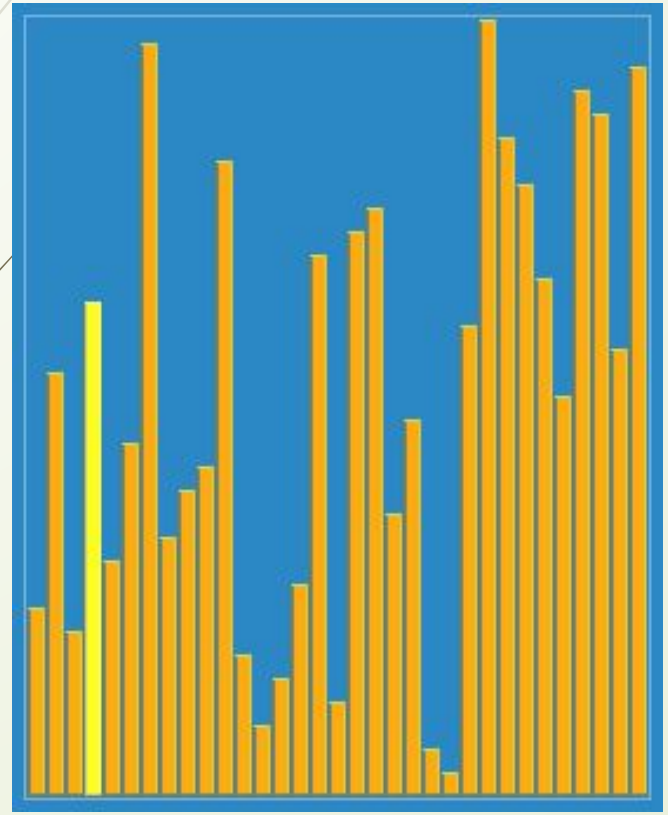
- Составьте программу, реализующую метод двойной сортировки выбором.

Массив нужно использовать тот же, что и в предыдущем задании

Сколько итераций теперь потребовалось вам для сортировки вашего массива?

Ответ на вопрос, листинг программы и результат расчета занесите в отчет по лабораторной работе

3. Гномья сортировка

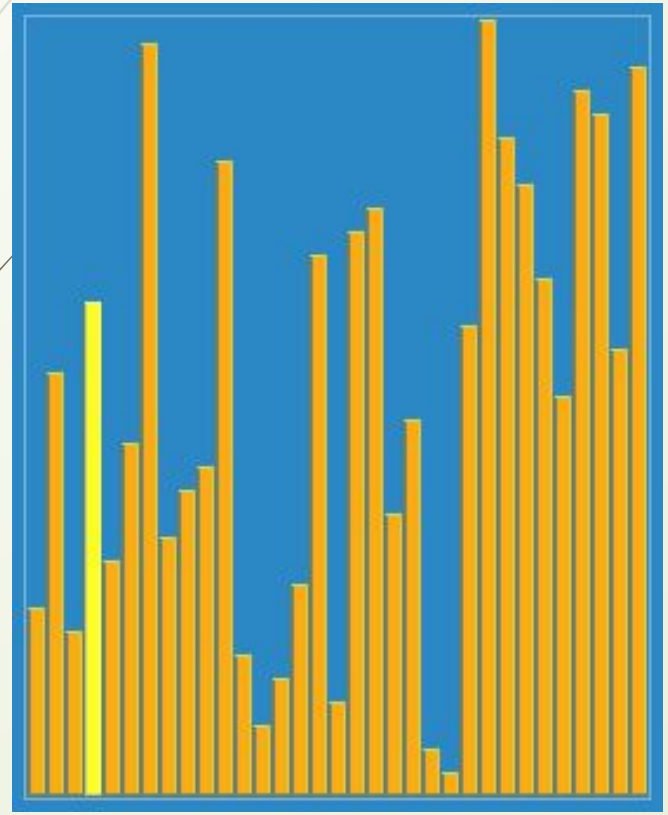


Гномья сортировка (англ. *Gnome sort*) — алгоритм сортировки, похожий на сортировку вставками, но в отличие от последней перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком.

Название происходит от предполагаемого поведения садовых гномов при сортировке линии садовых горшков.

По существу он смотрит на текущий и предыдущий садовые горшки: если они в правильном порядке, он шагает на один горшок вперёд, иначе он меняет их местами и шагает на один горшок назад. Граничные условия: если нет предыдущего горшка, он шагает вперёд; если нет следующего горшка, он закончил.

3. Гномья сортировка



Алгоритм концептуально простой, не требует вложенных циклов. Время работы $O(n^2)$

На практике алгоритм может работать так же быстро, как и сортировка вставками.

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 переставленных элементов.

Задание 3.1:

- Проанализировав алгоритм гномьей сортировки , составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 3.2:

- Составьте программу, реализующую метод гномьей сортировки.

  Массив, конечно, снова тот же

  Сколько итераций теперь потребовалось вам для сортировки вашего массива?

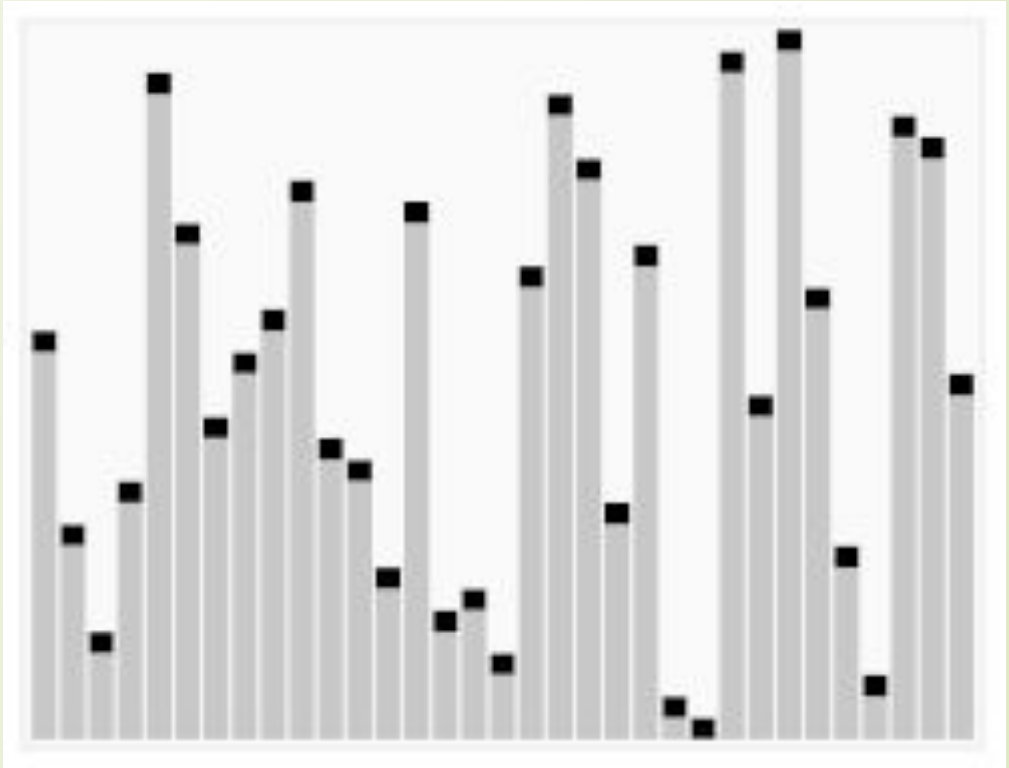
  Ответ на вопрос, листинг программы и результат расчета занесите в отчет по лабораторной работе

4. Быстрая сортировка Хоара

Наиболее популярный и применяемый алгоритм на практике

Этапы :

1. Выбирается опорный элемент.
2. Все значения меньше опорного перебрасываются влево, все остальные- вправо.
3. Алгоритм повторяется для каждой полученной части, пока возможно разделение массива



Задание 4.1:

- Проанализировав быстрой сортировки Хаара, составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 4.2 :

- Составьте программу, реализующую метод быстрой сортировки Хаара.

  Да, массив опять тот же

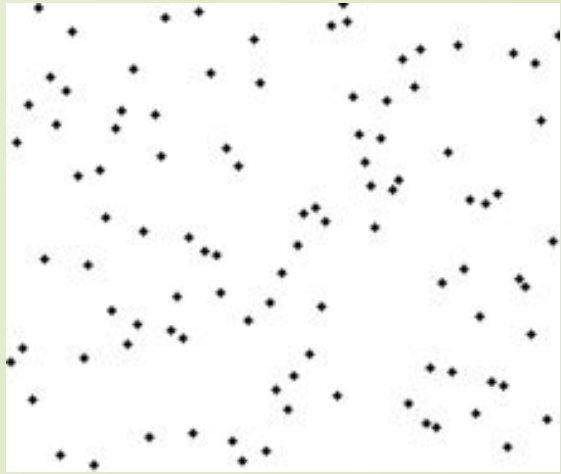
  Сколько итераций теперь потребовалось вам для сортировки вашего массива?

  Ответ на вопрос, листинг программы и результат расчета занесите в отчет по лабораторной работе

5. Сортировка слиянием

6 5 3 1 8 7 2 4

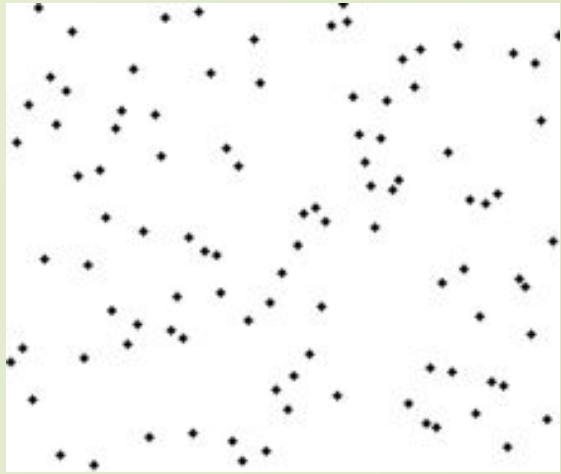
- Сортировка слиянием (англ. merge sort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке.
- Эта сортировка — хороший пример использования принципа «разделяй и 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.



5. Сортировка слияние

6 5 3 1 8 7 2 4

- **Этапы решения:**
- Сортируемый массив разбивается на две части примерно одинакового размера;
- Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;
- Два упорядоченных массива половинного размера соединяются в один.



5. Сортировка слиянием

6 5 3 1 8 7 2 4

- **Этапы решения:**

1.Сортируемый массив разбивается на две части примерно одинакового размера;

- Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

2**. Каждая из получившихся частей сортируется отдельно**, например — тем же самым алгоритмом;

3. Два упорядоченных массива половинного размера соединяются в один:

- 3.1. Соединение двух упорядоченных массивов в один.
  Основную идею слияния двух отсортированных массивов можно объяснить на следующем примере. Пусть мы имеем два уже отсортированных по возрастанию подмассива. Тогда:
- 3.2. Слияние двух подмассивов в третий результирующий массив.
  На каждом шаге мы берём меньший из двух первых элементов подмассивов и записываем его в результирующий массив. Счётчики номеров элементов результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1.
- 3.3. «Прицепление» остатка.
  Когда один из подмассивов закончился, мы добавляем все оставшиеся элементы второго подмассива в результирующий массив.

Задание 5.1:

- Проанализировав алгоритм сортировки слиянием, составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 5.2:

- Составьте программу, реализующую метод сортировки слиянием.

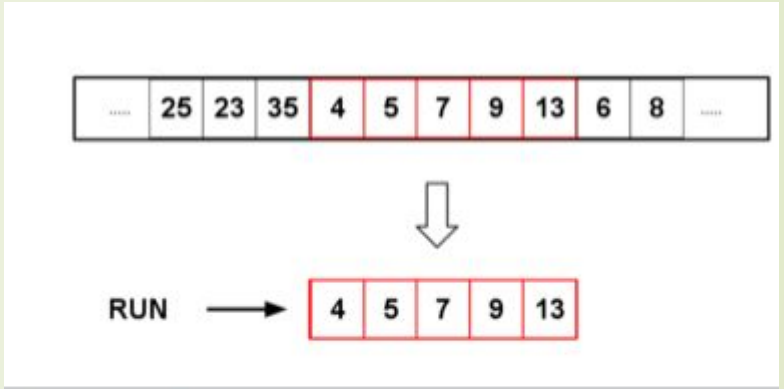
  И у вас массив тоже тот же

  Сколько итераций теперь потребовалось вам для сортировки вашего массива?

  Ответ на вопрос, листинг программы и результат расчета занесите в отчет по лабораторной работе

6. Timsort

Timsort — гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием, опубликованный в 2002 году **Тимом Петерсом**.



Алгоритм Timsort ищет в массиве упорядоченные последовательности, называемые run, для ускорения поиска.

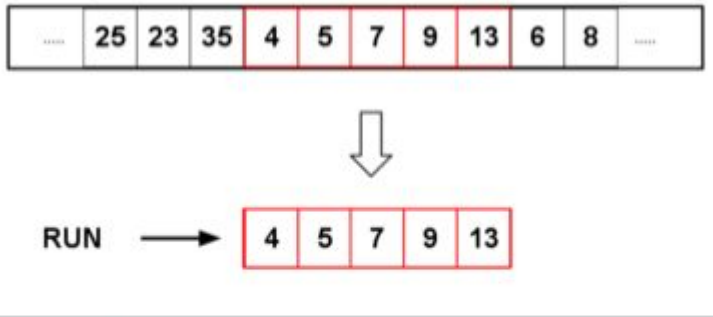
В настоящее время Timsort является стандартным алгоритмом сортировки в Python, OpenJDK 7 и реализован в Android JDK 1.5.

Основная идея алгоритма в том, что в реальном мире сортируемые массивы данных часто содержат в себе упорядоченные подмассивы. На таких данных Timsort существенно быстрее многих алгоритмов сортировки

Принцип:

- По специальному алгоритму входной массив разделяется на подмассивы.
- Каждый подмассив сортируется сортировкой вставками.
- Отсортированные подмассивы собираются в единый массив с помощью модифицированной сортировки слиянием.

Timsort



Алгоритм Timsort ищет в массиве упорядоченные последовательности, называемые run, для ускорения поиска.

Шаг 0. Вычисление minrun.

(1) Число minrun (минимальный размер упорядоченной последовательности) определяется на основе N исходя из следующих принципов: оно не должно быть слишком большим, поскольку к подмассиву размера minrun будет в дальнейшем применена сортировка вставками, а она эффективна только на небольших массивах.

(2) Оно не должно быть слишком маленьким, поскольку чем меньше подмассив — тем больше итераций слияния подмассивов придётся выполнить на последнем шаге алгоритма.

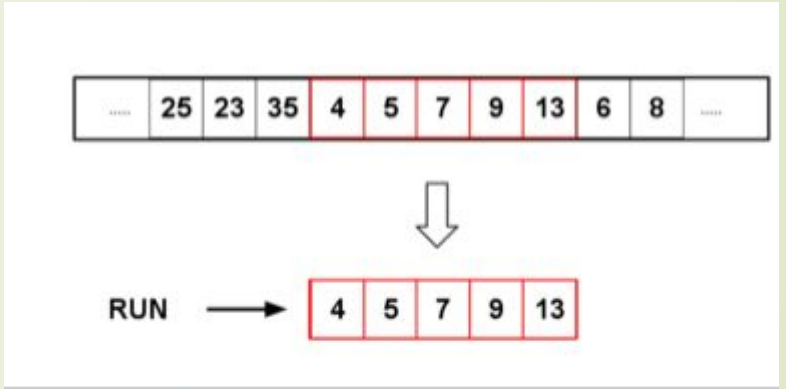
Оптимальная величина для N / minrun это степень числа 2 (или близким к нему).

При minrun > 256 нарушается пункт (1),

при minrun < 8 — пункт (2) и наиболее эффективно использовать значения из диапазона (32;65).

Исключение — если N < 64, тогда minrun = N и timsort превращается в простую сортировку вставкой.

Timsort



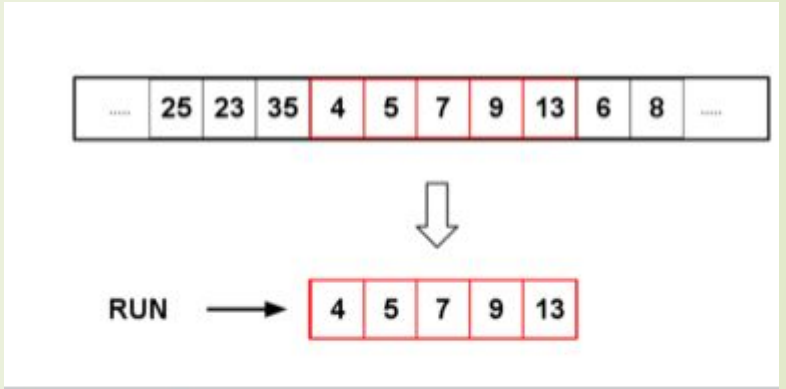
Алгоритм Timsort ищет в массиве упорядоченные последовательности, называемые run, для ускорения поиска.

Шаг 1. Разбиение на подмассивы и их сортировка

Указатель текущего элемента ставится в начало входного массива.

- Начиная с текущего элемента, в этом массиве идёт поиск упорядоченного подмассива run. По определению, в run однозначно войдет текущий элемент и следующий за ним. Если получившийся подмассив упорядочен по убыванию — элементы переставляются так, чтобы они шли по возрастанию.
- Если размер текущего run'а меньше, чем minrun — выбираются следующие за найденным run-ом элементы в количестве minrun-size(run). Таким образом, на выходе будет получен подмассив размером minrun или больше, часть которого (а в идеале — он весь) упорядочена.
- К данному подмассиву применяется сортировка вставками. Так как размер подмассива невелик и часть его уже упорядочена — сортировка работает быстро и эффективно.
- Указатель текущего элемента ставится на следующий за подмассивом элемент.
- Если конец входного массива не достигнут — переход к пункту 2, иначе — конец данного шага.

Timsort



Алгоритм Timsort ищет в массиве упорядоченные последовательности, называемые run, для ускорения поиска.

Шаг 2. Слияние

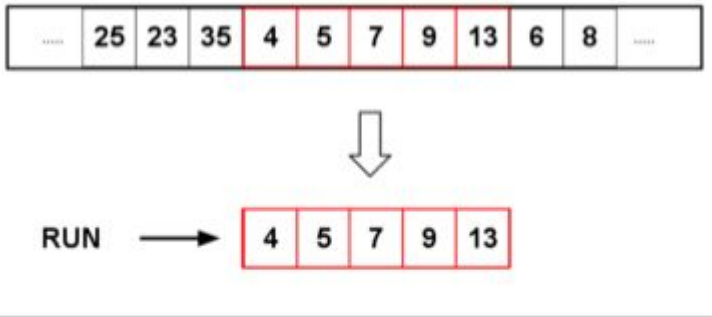
Если данные входного массива были близки к случайным — размер упорядоченных подмассивов близок к minrun, если в данных были упорядоченные диапазоны — упорядоченные подмассивы имеют размер, превышающий minrun.

Нужно объединить эти подмассивы для получения результирующего, полностью упорядоченного массива. Для достижения эффективности объединение должно удовлетворять двум **требованиям:**

- Объединять подмассивы примерно равного размера
- Сохранить стабильность алгоритма — то есть не делать бессмысленных перестановок.

Timsort

Шаг 2. Слияние



Алгоритм Timsort ищет в массиве упорядоченные последовательности, называемые run, для ускорения поиска.

Алгоритм:

Создается пустой стек пар <индекс начала подмассива>-<размер подмассива>. Берётся первый упорядоченный подмассив.

В стек добавляется пара данных <индекс начала>-<размер> для текущего подмассива.

Определяется, нужно ли выполнять процедуру слияния текущего подмассива с предыдущими. Для этого проверяется выполнение двух правил (пусть X, Y и Z — размеры трёх верхних в стеке подмассивов):

$$Z > Y + X$$
$$Y > X$$

Если одно из пра[illegible]тся — массив Y сливается с меньшим из массивов X и Z. Повторяется до выполнения обоих правил или полного упорядочивания данных.

Если еще остались не рассмотренные подмассивы — берётся следующий и переходим к пункту 2. Иначе — конец.

Цель этой процедуры — сохранение баланса. размеры подмассивов в стеке эффективны для дальнейшей сортировки слиянием.
В идеальном случае: есть подмассивы размера 128, 64, 32, 16, 8, 4, 2, 2.

В этом случае никакие слияния не выполнятся, пока не встретятся 2 последних подмассива, после чего будут выполнены 7 идеально сбалансированных слияний.

Timsort

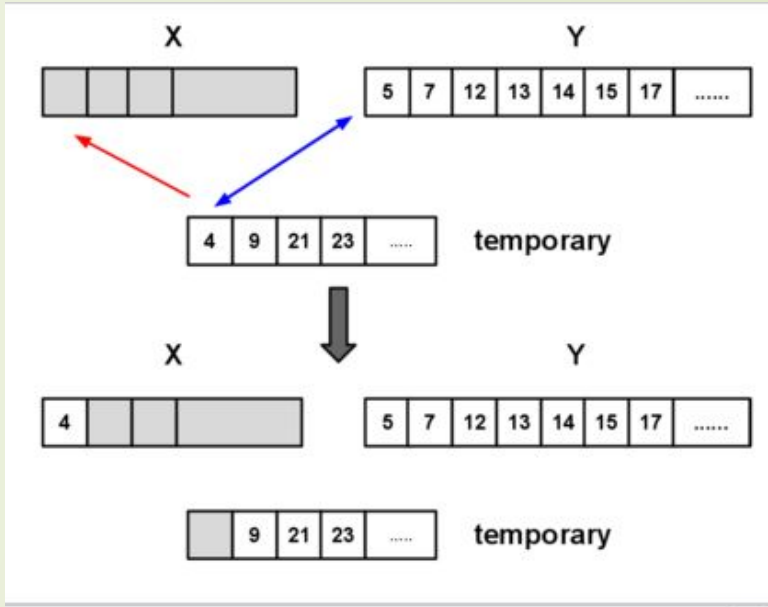
Шаг 2. Слияние

Процедура слияния подмассивов

1. Создаётся временный массив в размере меньшего из соединяемых подмассивов.
2. Меньший из подмассивов копируется во временный массив
3. Указатели текущей позиции ставятся на первые (последние) элементы большего и временного массива.
4. На каждом следующем шаге рассматривается значение текущих элементов в большем и временном массивах, берётся меньший (больший) из них и копируется в новый отсортированный массив. Указатель текущего элемента перемещается в массиве, из которого был взят элемент.

Пункт 4 повторяется, пока один из массивов не закончится.

Все элементы оставшегося массива добавляются в конец нового массива..



Элемент временного массива(выделенный голубой стрелкой) сравнивается с наименьшим элементом большого массива и меньший из них перемещается в новый отсортированный массив (как показано красной стрелкой).

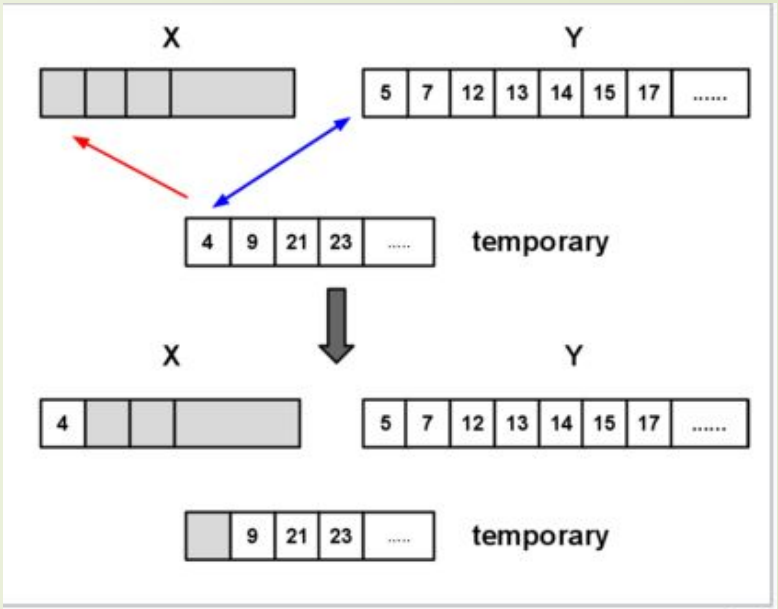
Процедура слияния совершает обход и сравнение в зависимости от того, как расположены подмассивы,
Для реализации алгоритма необходимы процедуры обхода слева направо (если меньший массив находится слева) и справа налево (если меньший массив находится справа).
На практике обычно ограничиваются первой процедурой и выбирают левый массив независимо от его размера— это почти не влияет на скорость сортировки.

Timsort

Шаг 2. Слияние

Процедура слияния подмассивов

1. Создаётся временный массив в размере меньшего из соединяемых подмассивов.
2. Меньший из подмассивов копируется во временный массив
3. Указатели текущей позиции ставятся на первые (последние) элементы большего и временного массива.
4. На каждом следующем шаге рассматривается значение текущих элементов в большем и временном массивах, берётся меньший (больший) из них и копируется в новый отсортированный массив. Указатель текущего элемента перемещается в массиве, из которого был взят элемент.
5. Пункт 4 повторяется, пока один из массивов не закончится.
6. Все элементы оставшегося массива добавляются в конец нового массива..



Элемент временного массива(выделенный голубой стрелкой) сравнивается с наименьшим элементом большого массива и меньший из них перемещается в новый отсортированный массив (как показано красной стрелкой).

Процедура слияния совершает обход и сравнение в зависимости от того, как расположены подмассивы,
Для реализации алгоритма необходимы процедуры обхода слева направо (если меньший массив находится слева) и справа налево (если меньший массив находится справа).
На практике обычно ограничиваются первой процедурой и выбирают левый массив независимо от его размера— это почти не влияет на скорость сортировки.

Задание 6.1

- Проанализировав алгоритм сортировки Timsort, составьте блок-схему работы данного метода

Результат занесите в отчет по лабораторной работе

Задание 6.2 :

- Составьте программу, реализующую метод сортировки Timsort.

Сколько итераций теперь потребовалось вам для сортировки вашего массива?

Массив.. Ну, вы поняли

Ответ на вопрос, листинг программы и результат расчета занесите в отчет по лабораторной работе

Теперь составляем сводную таблицу с учетом результатов прошлой работы

Задание	Метод	Количество итераций
1		
2		
….		

И проводим сравнительный анализ результатов

Да, письменно и в отчете.

Этот анализ мы записываемы в вывод по лабораторной работе

Собираем сводную таблицу эффективности всех известных вам методов (используем дополнительные источники)

метод	скорость	примечания

Дополните отчет

- Титульным листом
- Целью и вариантом задания

…и можно отправлять на проверку