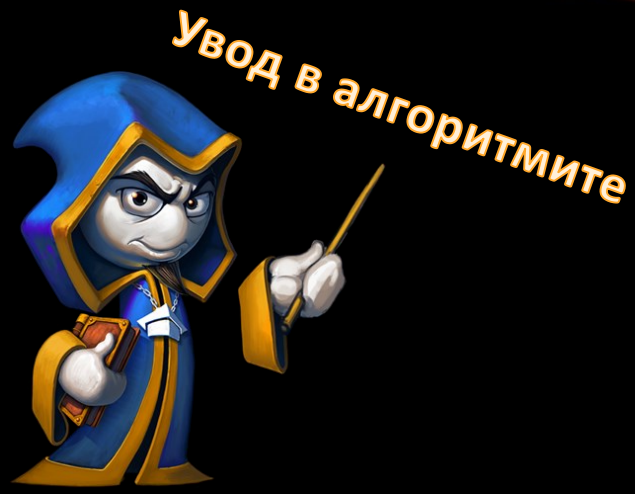
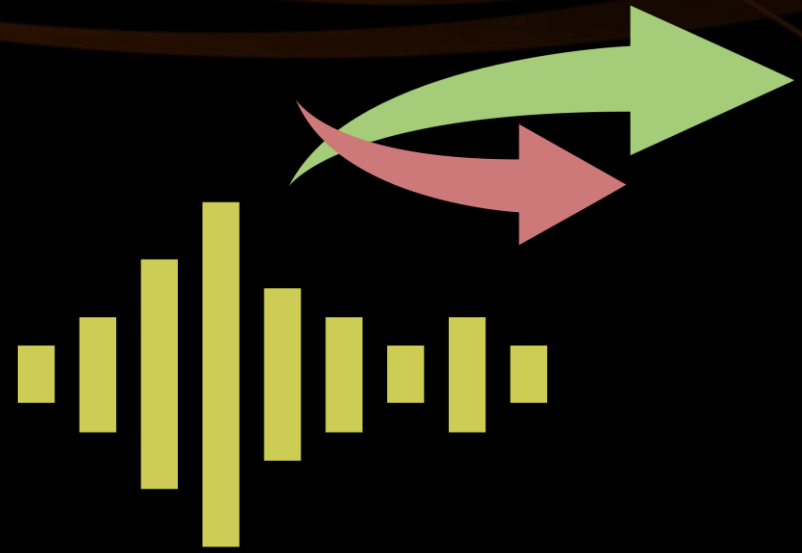
Други алгоритми за сортиране





Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/

Съдържание

1. Разбъркване
2. Сортиране чрез сливане
3. Бързо сортиране
4. Сортиране чрез броене
5. Bucket сортиране
6. Сравнение на алгоритмите

Разбъркване

- Разбъркване (shuffling) == постигане на случаен ред на елементите в колекция
  - Пораждане на случайни пермутации
- *Генерирането на случайни числа е прекалено важно, за да бъде оставено на шанса.* —Robert R. Coveyou

Алгоритъм за разбъркване на Fisher–Yates

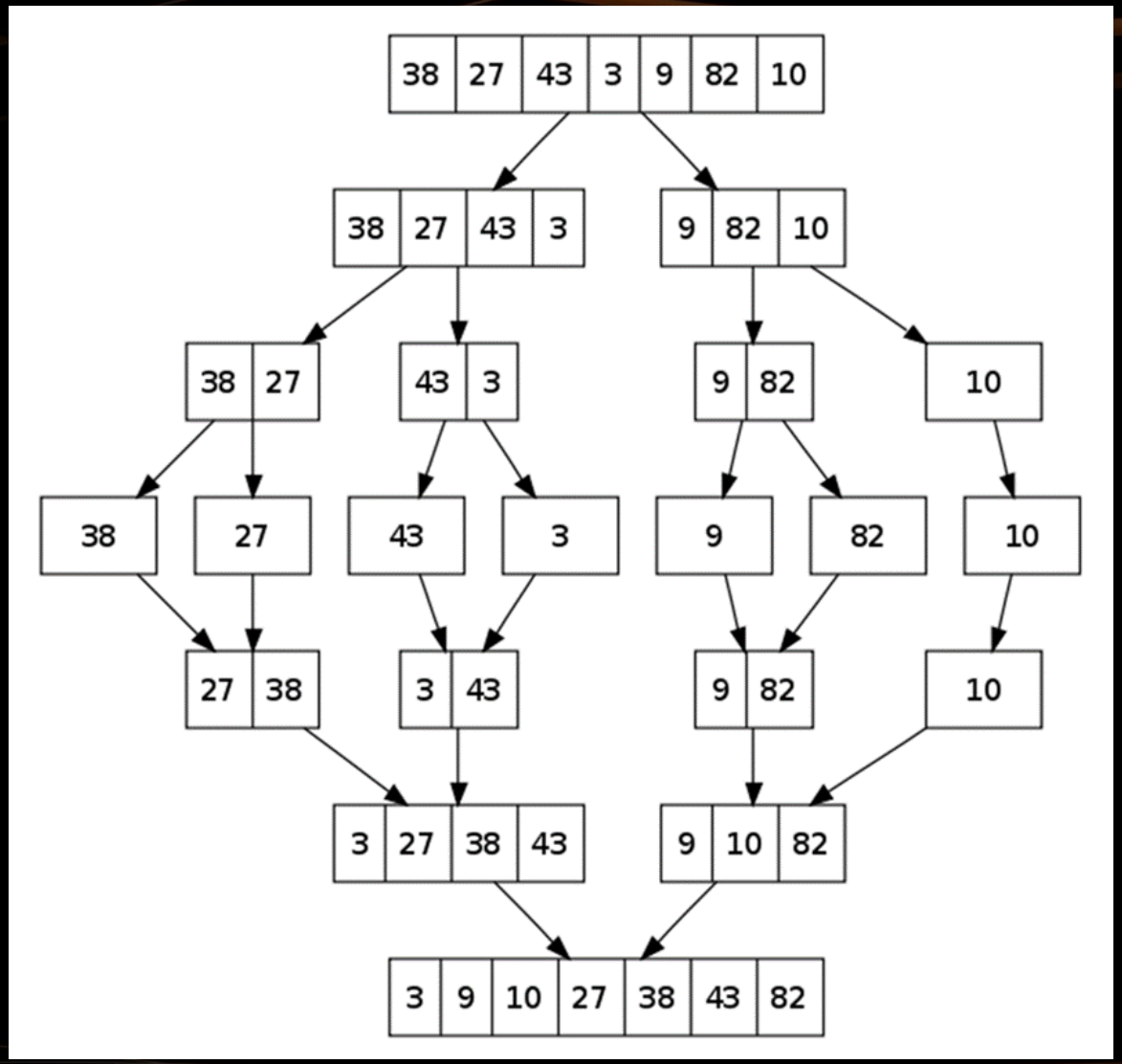
```
public static void Shuffle<T>(T[] source)
{
  Random rnd = new Random();

  for (int i = 0; i < source.Length; i++)
  {
    // Exchange array[i] with random element in array[i … n-1]

    int r = i + rnd.Next(0, source.Length - i);

    T temp = source[i];
    source[i] = source[r];
    source[r] = temp;
  }
}
```

Shuffle algorithms: visualization

Сортиране чрез сливане

- Сортиране чрез сливане (merge sort) е ефективен алгоритъм за сортиране (онагледяване)
  1. Разделя списъка на подсписъци (обикновено 2 подсписъка)
  2. Сортира всеки подсписък (рекурсивно извиквайки merge-sort)
  3. Слива сортираните подсписъци в един списък
- Най-добър, обичаен и най-лош случай: O(n*log(n))
- Памет:
  - Обикновено O(n)
  - Със сливане на място стига до O(1)
- Стабилен: Да
- Метод: Сливане
- Приложим за паралелно изпълнение на множество ядра / машини → до O(log(n))

Сортиране чрез сливане: Как работи?

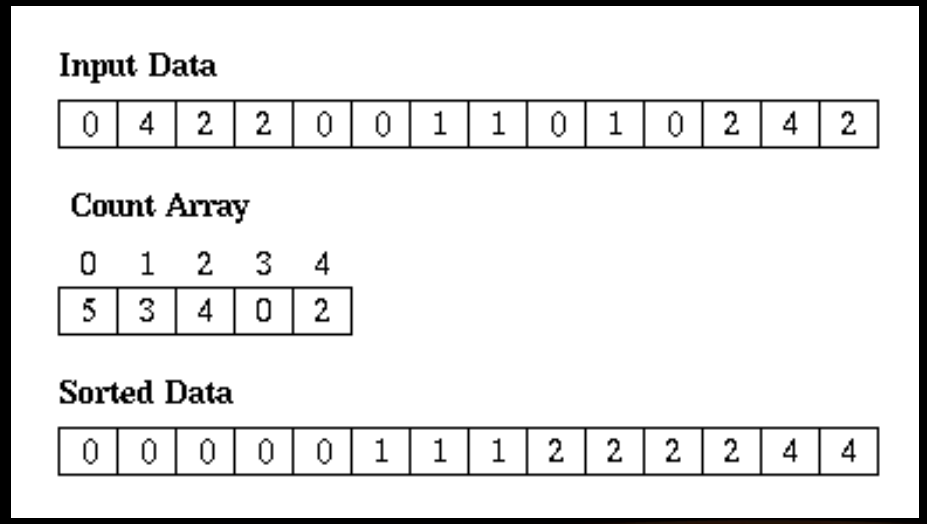


Бързо сортиране

- Бързо сортиране (QuickSort) – ефективен алгоритъм за сортиране (онагледяване)
  - Избира се „опорен“ елемент; премества по-малките елементи вляво от него, а по-големите - вдясно; сортира лявата и дясната част
  - Най-добър и обичаен случай: $O(n*\log(n))$; Най-лош: $O(n^2)$
  - Памет: $O(\log(n))$ място в стека (за рекурсия)
  - Стабилен: Зависи
  - Метод: Разделяне

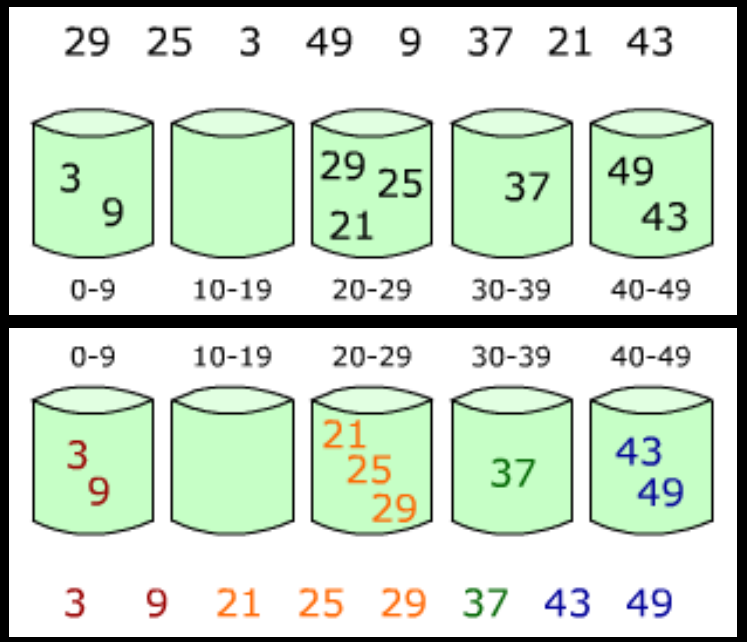
Сортиране чрез броене

- Сортиране чрез броене (counting sort) е много ефективен алгоритъм за сортиране (онагледяване)
  - Сортира малки числа чрез броене на техните срещания
  - Не е базиран на сравнение
- Най-добър, обичаен и най-лош случай: O(n + k)
  - k е диапазонът на сортираните числа
  - Например [-1000 ... 1000] → k = 2001
- Памет: O(n + k) Място: O(k)
- Стабилен: Да
- Метод: Броене



Bucket сортиране

- Bucket sort разделя масива на много „ведра“
  - Всяко „ведро“ се сортира с различен алгоритъм
  - Не е сортиране, базирано на сравнение
- Обичайните случаи: O(n + k)
  - k = броя на „ведрата“
- Най-лошия случай: O(n * log n)
- Стабилен: Да (зависи от алгоритъма)
- Памет: O(n) – k „ведра“ съдържащи общо n елемента



Сравнение на сортиращи алгоритми

онагледяване

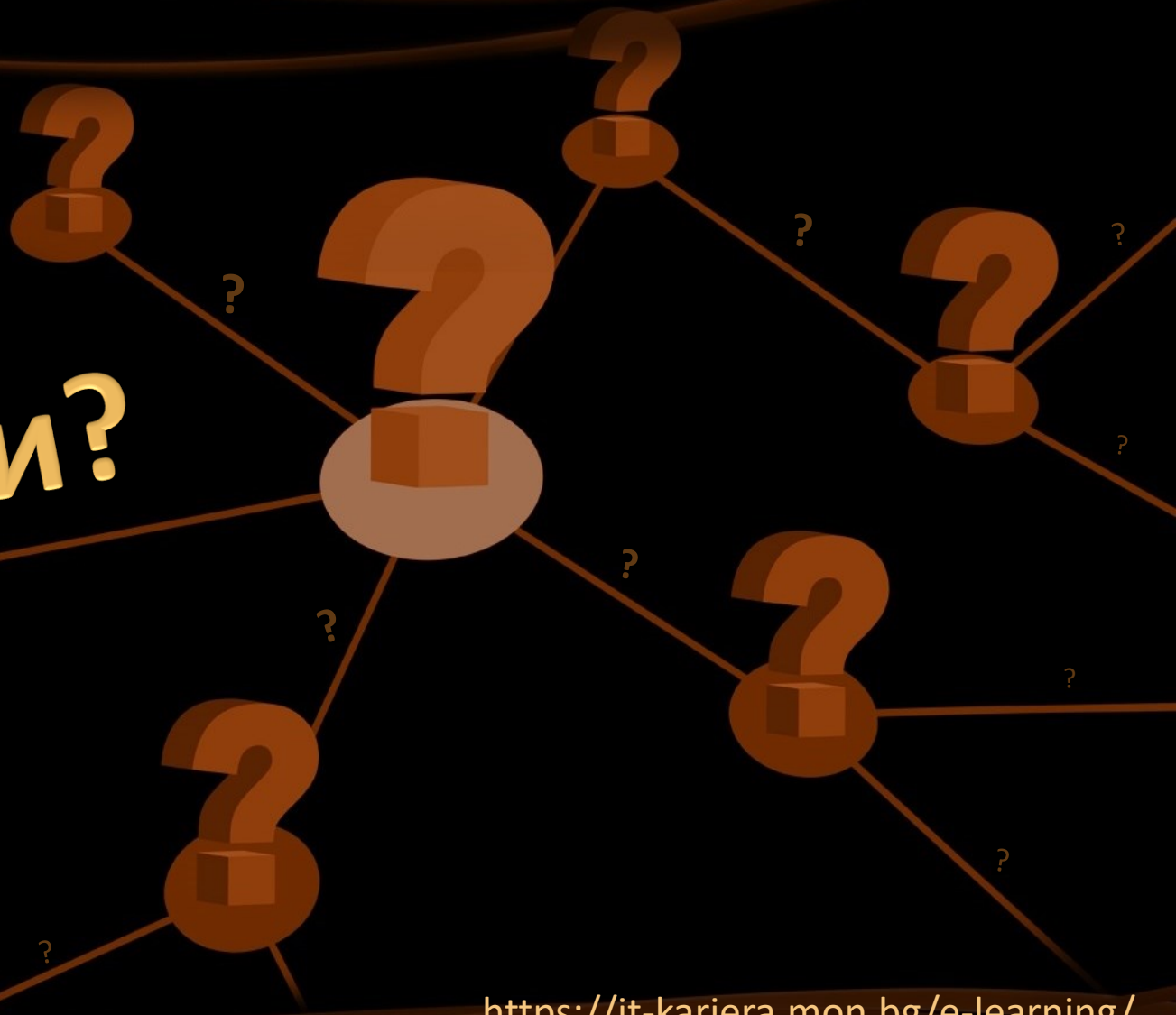
Име	Най-добре	Средно	Най-зле	Памет	Стабилен	Метод
SelectionSort	n^2	n^2	n^2	1	Не	Селекция
BubbleSort	n	n^2	n^2	1	Да	Размяна
InsertionSort	n	n^2	n^2	1	Да	Вмъкване
QuickSort	n*log(n)	n*log(n)	n^2	log(n)	Зависи	Разделяне
MergeSort	n*log(n)	n*log(n)	n*log(n)	1 (or n)	Да	Сливане
HeapSort	n*log(n)	n*log(n)	n*log(n)	1	Не	Селекция
BogoSort	n	n*n!	n*n!	1	Не	Късмет

Обобщение

- Бавни алгоритми за сортиране:
  - Чрез пряка селекция, метод на мехурчето, сортиране чрез вмъкване
- Бързи алгоритми за сортиране:
  - Бързо сортиране, сортиране чрез сливане и т.н.
- Как да изберем най-удачния метод за сортиране?
  - http://stackoverflow.com/a/1934004

Други алгоритми за сортиране

Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**