Алгоритмы сортировки

Хайрулин Сергей Сергеевич
s.khayrulin@gmail.com

Overview

1. Сортировка выбором.
2. Сортировка пузырьком.
3. Сортировка перемешиванием
4. Сортировка вставками.
5. Быстрая сортировка.
6. Пирамидальная сортировка.
7. Алгоритм поразрядной сортировка

Литература и др. источники

1. Дональд Эрвин Кнут. Искусство программирования (Том 1, 2, 3) // Вильямс 2015.
2. Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы // Вильямс 2000.
3. Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов // М.: Наука, 1990.
4. Харари Ф. Теория графов // М.: Мир, 1973.
5. Косточка А. В. Дискретная математика. Часть 2 //Новосибирск: НГУ, 2001.
6. Котов В. Е., Сабельфельд В. К. Теория схем программ // Наука 1991.
7. http://algolist.manual.ru
8. ….

Постановка задачи

Пусть есть последовательность a_0, a_1... a_n и функция сравнения, которая на любых двух элементах последовательности принимает одно из трех значений: меньше, больше или равно. Задача сортировки состоит в перестановке членов последовательности таким образом, чтобы выполнялось условие: $a_i <= a_{i+1}$, для всех i от 0 до n.

Постановка задачи

Время сортировки - основной параметр, характеризующий быстродействие алгоритма.

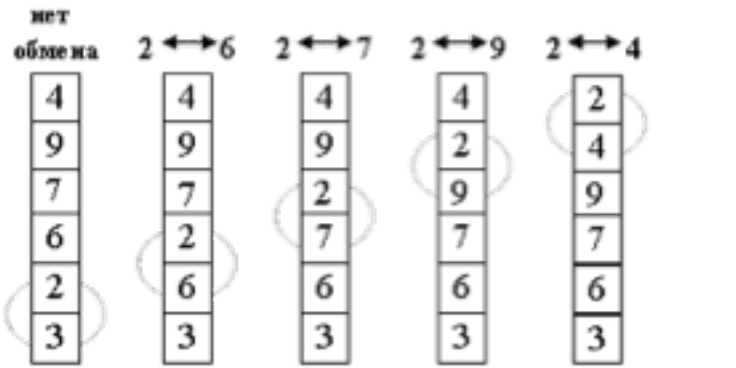
Постановка задачи

Память - ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. При оценке используемой памяти не будет учитываться место, которое занимает исходный массив и независящие от входной последовательности затраты, например, на хранение кода программы.

Постановка задачи

Устойчивость - устойчивая сортировка не меняет взаимного расположения равных элементов. Такое свойство может быть очень полезным, если они состоят из нескольких полей, а сортировка происходит по одному из них, например, по x.

Сортировка пузырьком



Нулевой проход, сравниваемые пары выделены

Сортировка пузырьком

Реализация?

Недостатки?

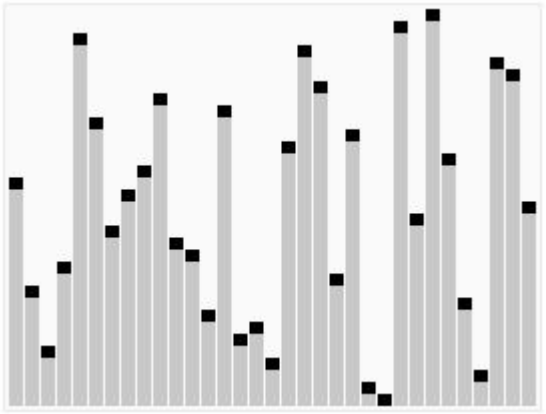
Сложность?

Сортировка перемешиванием (шейкер)

```
def shaker_sort(l):
    k = len(l) - 1
    ub = len(l) - 1
    db = 0
    while ub > db:
        for i in range(ub,0,-1):
            if l[i] < l[i-1]:
                l[i], l[i-1] = l[i-1], l[i]
                k = i
        db = k + 1
        for i in range(1,ub+1):
            if l[i] < l[i-1]:
                l[i], l[i-1] = l[i-1], l[i]
                k = i
        ub = k - 1
```

Сортировка простыми вставками

Быстрая сортировка (Qsort)

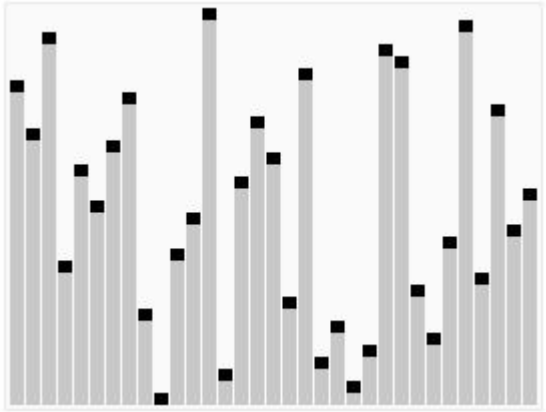


Быстрая сортировка (Qsort)

Реализация?

Сложность?

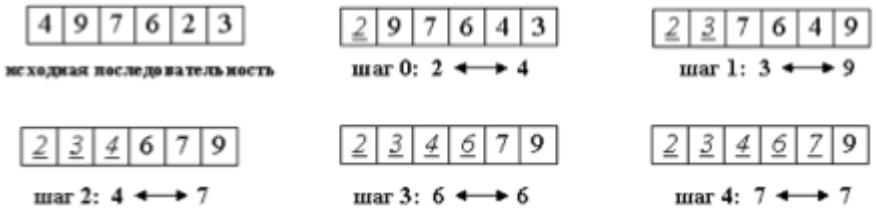
Heap Sort (пирамид)



Radix - sort

```
def radix_sort(l):
    k = calc_k(max(l))
    for j in range(k):
        l_i = [[] for x in range(10)]
        for i in l:
            d = int(i / 10**(j - 1)) % 10
            l_i[d].append(i)
        l = [item for sublist in l_i for item in sublist]
    print(l)
```

Сортировка Выбором




```
def select_sort(l):
    for i in range(len(l)):
        for j in range(i+1,len(l)):
            if l[i] > l[j]:
                l[i], l[j] = l[j], l[i]
```

Указания для задач

Для замера работы функции нужно использовать метод now() класса datetime модуля datetime

Указания для задач

```
array = [0] * N
array.insert(N,0)
```

```
import datetime


def main():
    t1 = datetime.datetime.now()
    #You'r code here
    ...
    print(datetime.datetime.now() - t1)

if __name__ == '__main__':
    main()
```

Указания для задач

```
import numpy as np


...
# Generate numpy Array with N random numbers
array = np.random.rand(N)
#Sort Array by quick sort
sorted(array)
...
```

Указания для задач

jupyter MyTests Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

In [6]:
```
import datetime
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```

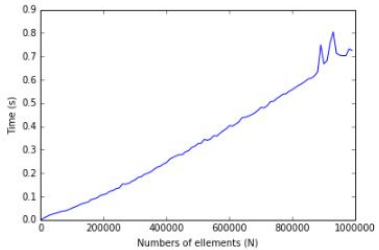
In [7]:
```
times = []
for N in xrange(1000, 1000000, 10000):
    array = np.random.rand(N)
    t1 = datetime.datetime.now()
    sorted(array)
    t_res = datetime.datetime.now() - t1
    times.append([N,t_res])
y = [t[1].total_seconds() for t in times]
x = [t[0] for t in times]
```

In [8]:
```
%matplotlib inline
```

In [9]:
```
plt.plot(x, y)

plt.xlabel('Numbers of ellements (N)')
plt.ylabel('Time (s)')

plt.savefig("test.png")
plt.show()
```



Задачи

- Реализовать алгоритм перемножения квадратных матриц. Матрицы могут задаваться как список списков. Считывать можно из файла потока ввода, или задавать случайным образом (используя функцию np.random.rand(N)). Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Найти все пифагоровы тройки ($c^2 = a^2 + b^2$) для заданного интервала. Интервал задается парой чисел через пробел считанных из входного потока (например: 10 100) помните, что верхняя грань отрезка должна быть больше нижней. Если задано одно число, то считаем, что ограничение снизу равно по умолчанию 1. Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм факторизации числа (разложение числа как произведение двух других чисел). Оценить временную и ассимптотическую сложность алгоритма, построить график.
- Реализовать алгоритм рассчитывающий сочетания и размещения.
- Факториал довольно емкостная функция, при расчете которого для больших значений может случится переполнение (т.е. полученное число будет больше чем максимально возможное число в вашей системе). Подумайте как преодолеть эту проблему.

Задачи

Написать оболочку для работы с графами:

- создавать графы
- Выводить граф (в виде таблицы смежности)
- Удалять ребра
- Ищет путь в графе для заданных вершин
  - Флойда-Уоршела
  - Форда-Беллмана
  - Дейкстра

Задачи

1. Скачать файл https://goo.gl/z7H7DU
2. Файл содержит карту препятствия обозначены символом ‘%’ клетки, по которым можно передвигаться обозначены ‘-’, при этом каждая клетка по которой можно двигаться имеет вес 1.
3. Робот начинает движение в клетке обозначенной буквой ‘P’ и движется в клетку обозначенной буквой ‘T’.
4. Нужно рассчитать оптимальную траекторию пути робота с помощью алгоритма A*.
5. Выведите траекторию в отдельный файл.

Задачи

1. Реализуйте функцию DFS
2. С помощью вашей функции реализуйте алгоритм разбиение графа на компоненты связности.
3. Реализуйте алгоритм проверки орграфа на цикличность
4. Реализуйте алгоритм Крускала/Прима для поиска минимального остовного дерева взвешенного графа.

Задачи

1. Как можно объединить два списка. Напишите программу делающую это
2. Реализуйте очередь через два стека.

Спасибо за внимание!