Алгоритмы поиска, Хеш функция, Хеш таблица.

Хайрулин Сергей Сергеевич
s.khayrulin@gmail.com

Overview

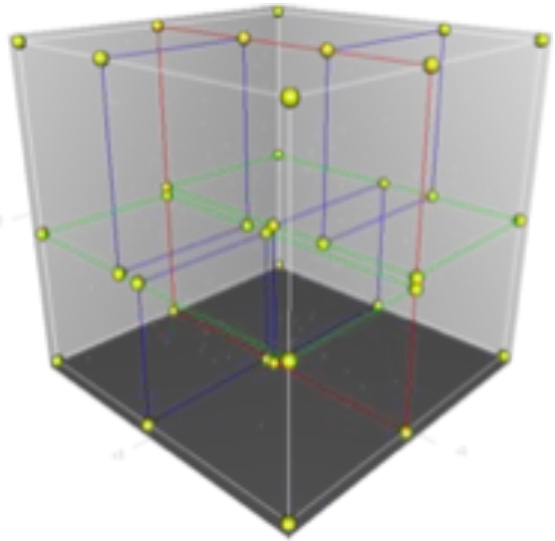
1. Линейный поиск.
2. Бинарный поиск.
    - Рекурсивный
    - нерекурсивный алгоритмы.
3. Интерполяционный поиск элемента в массиве
4.

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. ….

k-d деревья

k-d дерево (k-d tree, сокращение от k-мерное дерево) — это структура данных с разбиением пространства для упорядочивания точек в k-мерном пространстве. k-d деревья — особый вид двоичных деревьев поиска.



Линейный поиск

Простой перебор элементов массива

```
def linear_search(array, val):
    for i in range(len(array)):
        if element == val:
            return i
    return -1
```

Линейный поиск

Сложность?

Бинарный поиск

Двоичный (бинарный) поиск (также известен как метод деления пополам и **дихотомия**) — классический алгоритм поиска элемента в **отсортированном** массиве (векторе), использующий дробление массива на половины.

Бинарный поиск

- Рекурсивный вариант
- Нерекурсивный

Бинарный поиск

Сложность?

Интерполяционный поиск

Есть **a** - отсортированный массив. Задача найти элемент **x** в этом массиве. Если известно верхняя граница a_l и нижняя граница a_h где l, h индексы в массиве **a**.

$$\frac{x - a_l}{a_h - a_l} \cdot (h - l)$$

Хеш-таблица

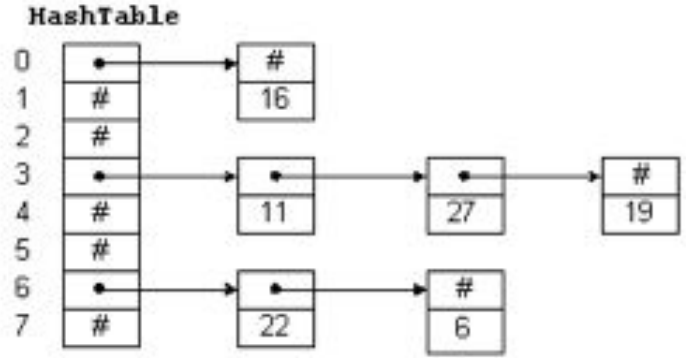
U - множество объектов (универсум).

$h : U \to S = \{0, .., m-1\}$ - хеш-функция отображение из множества U в множество S.

Если $x \in U$, значит $h(x) \in U$

Хеш-таблица

Хеш-таблица - это обычный массив с необычной адресацией, задаваемой хеш-функцией.



Хеш-таблица

Виды хеширования

По способу хранения:

- Статическое — фиксированное количество элементов. Один раз заполняем хеш-таблицу и осуществляем только проверку на наличие в ней нужных элементов,
- Динамическое — добавляем, удаляем и смотрим на наличие нужных элементов.

Хеш-таблица

По виду хеш-функции:

- Детерминированная хеш-функция,
- Случайная хеш-функция.

Хеш-функция

Деление (размер таблицы **hashTableSize** - простое число). Хеширующее значение hashValue, изменяющееся от 0 до (**hashTableSize** - 1), равно остатку от деления ключа на размер хеш-таблицы.

Хеш-функция

Мультипликативный метод (размер таблицы **hashTableSize** есть степень 2^n). Значение key умножается на константу, затем от результата берется **n** бит. В качестве такой константы Кнут рекомендует золотое сечение:

$$\frac{(\sqrt{(5)}-1)}{2} = 0.618033988749 .$$

Хеш-функция

Пусть, например, мы работаем с таблицей из **hashTableSize** = 32 = (2^5) элементов, хеширование производится байтами (8 бит, unsigned char). Тогда необходимый множитель.

$$2^8 \cdot \frac{(\sqrt{5}-1)}{2} = 158$$

Хеш-функция

Аддитивный метод для строк переменной длины (размер таблицы равен 256). Для строк переменной длины вполне разумные результаты дает сложение по модулю 256. В этом случае результат **hashValue** заключен между 0 и 255.

Хеш-функция

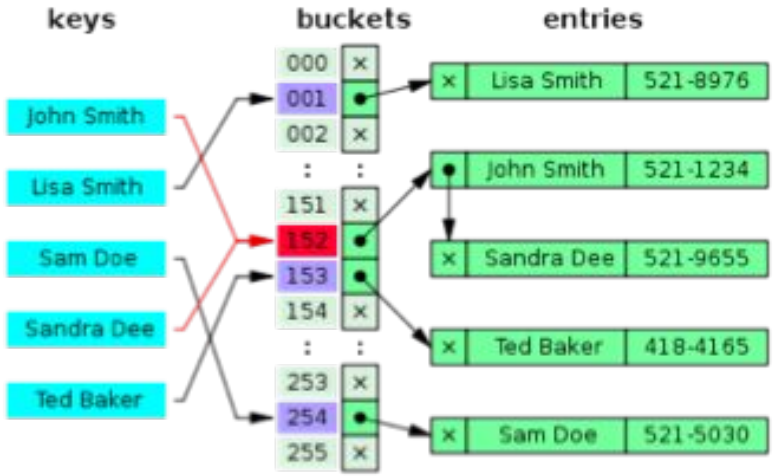
Исключающее ИЛИ для строк переменной длины (размер таблицы равен 256). Этот метод аналогичен аддитивному, но успешно различает схожие слова и анаграммы (аддитивный метод даст одно значение для XY и YX). Метод, как легко догадаться, заключается в том, что к элементам строки последовательно применяется операция "исключающее или".

Коллизия хеша

Коллизия: $\exists x \neq y, h(x) = h(y)$

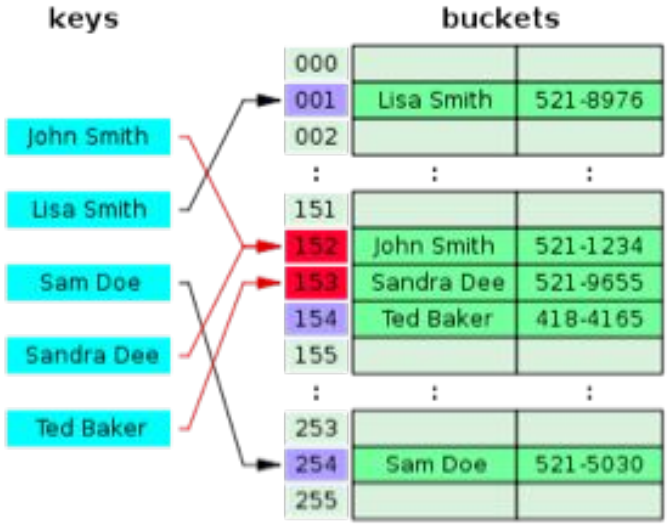
Коллизия хеша

Метод цепочек



Коллизия хеша

Открытая адресация



Сложность

	В среднем	**В худшем случае**
Расход памяти	O(n)	O(n)
Поиск	O(1)	O(n)
Вставка	O(1)	O(n)
Удаление	O(1)	O(n)

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
```

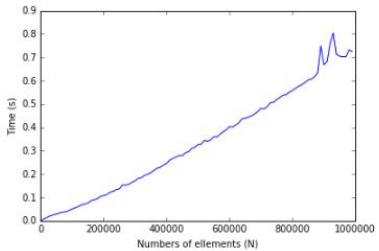
In [7]:
```
times = []
for N in xrange(1000, 1000000, 10000):
    array = np.random.rand(N)
    t1 = datetime.datetime.now()
    sorted(array)
    t_res = datetime.datetime.now() - t1
    times.append([N,t_res])
y = [t[1].total_seconds() for t in times]
x = [t[0] for t in times]
```

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!