Алгоритмы поиска подстроки в строке

Хайрулин Сергей Сергеевич
s.khayrulin@gmail.com

Overview

- Алгоритмы точного поиска подстроки в строке
- Префиксные деревья
- Алгоритм Ахо-Корасика → на следующей лекции

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. ….

Алгоритмы точного поиска подстроки в строке

Алгоритм	**Время на пред обработку**	**Среднее время поиска**	**Худшее время поиска**	**Затраты по памяти**
Грубой силы	Нет	2*n	O(n*m)	Нет
Бойера-Мура	O(m+s)	O(m+n)	O(n*m)	O(m+s)
Бойера-Мура-Хорспула	O(m+s)	O(m+n)	O(n*m)	O(m+s)
Турбо-БМ	O(m+s)	O(m+n)	2*n	O(m+s)
Быстрый поиск	O(m+s)	O(m+n)	O(n*m)	O(m+s)
Оптимальное несовпадение	O(m+s)	O(m+n)	O(n*m)	O(m+s)

Алгоритмы точного поиска подстроки в строке

Турбо-обращение сегмента	O(m)	O(n·(log_s (m))/m)	O(n*m)	O(m+s)
Максимальный сдвиг	O(m+s)	O(m+n)	O(n*m)	O(m+s)
Сдвиг-Или	O(m+s)	O(n)	O(n)	-

Алгоритм Грубой Силы

Как по вашему мнению он мог бы выглядеть?

Алгоритм Бойера-Мура

1. Сканирование слева направо, сравнение справа налево.

Алгоритм Бойера-Мура

2. Функция плохого символа

Строка: * * * * * * к * * * * * *

Шаблон: к о л о к о л

Следующий шаг: к о л о к о л

Если стоп-символа в шаблоне вообще нет, шаблон смещается за этот стоп-символ.

Строка: * * * * * * а л * * * * * * * *

Шаблон: к о л о к о л

Следующий шаг: к о л о к о л

Алгоритм Бойера-Мура

3. Функция хорошего суффикса.

Строка: * * * т о к о л * * * * *

Шаблон: к о л о к о л

Следующий шаг: к о л о к о л

Алгоритм Бойера-Мура

Таблица стоп-символов. Для каждой символа алфавита указывается последняя позиция этого символа в строке. Для всех остальных символов указывается 0 если нумерация идет с 1 и -1 если 0.

Пример для строки s=abcdadcd.

Символ	a	b	c	d	[все остальные]
Последняя позиция	4	1	6	5	-1

Алгоритм Бойера-Мура

Таблица суффиксов

Для каждого возможного суффикса t данного шаблона s указываем наименьшую величину, на которую нужно сдвинуть вправо шаблон, чтобы он снова совпал с t и при этом символ, предшествующий этому вхождению t, не совпадал бы с символом, предшествующим суффиксу t. Если такой сдвиг невозможен, ставится |s|=m.

Пример для строки s=колокол

Суффикс	[пустой]	л	ол	кол	...	олокол	колокол
Сдвиг	1	7	7	4	...	4	4
Иллюстрация							
было	?	?л	?ол	?кол	...	?олокол	колокол
стало	колокол	колокол	колокол	колокол	...	колокол	колокол

Алгоритм Бойера-Мура-Хорспула

Этот алгоритм - некоторое упрощение стандартного Бойера - Мура. В 1980 году Хорспул (Horspool) предложил использовать только сдвиг по самому правому символу для вычисления сдвига в алгоритме Бойера – Мура

Алгоритм Карпа-Рабина

1. Легко вычисляться.
2. Как можно лучше различать несовпадающие строки.
3. hash(y[i+1 , i+m]) должна легко вычисляться по hash(y[i , i+m-1]): hash(y[i+1 , i+m]) = rehash(y[i], y[i+m], hash(y[i , i+m-1]).

hash(w[0 , m-1]) = (w[0]· 2^(m-1) + w[1] ·2^(m-2) + … + w[m-1]) mod q,

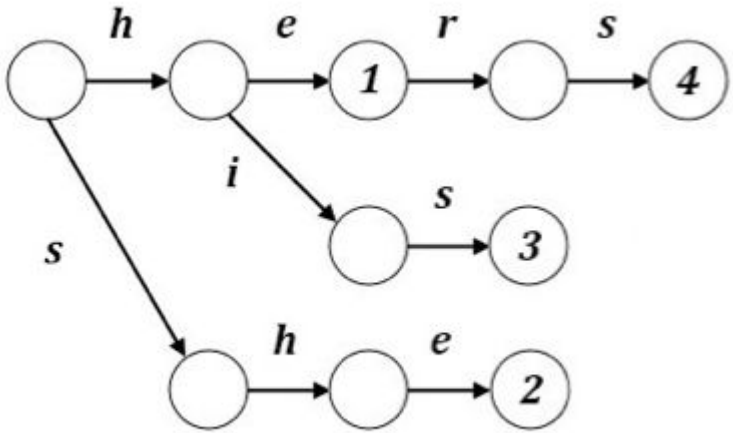
rehash(a, b, h) = ((h - a ·2^(m-1)) · 2 + b) mod q.

Бор (Префиксное дерево, луч, англ. trie)

Бор - структура данных для хранения набора строк, представляющая из себя подвешенное дерево с символами на рёбрах. Строки получаются последовательной записью всех символов, хранящихся на рёбрах между корнем бора и терминальной вершиной. Размер бора линейно зависит от суммы длин всех строк, а поиск в бору занимает время, пропорциональное длине образца.

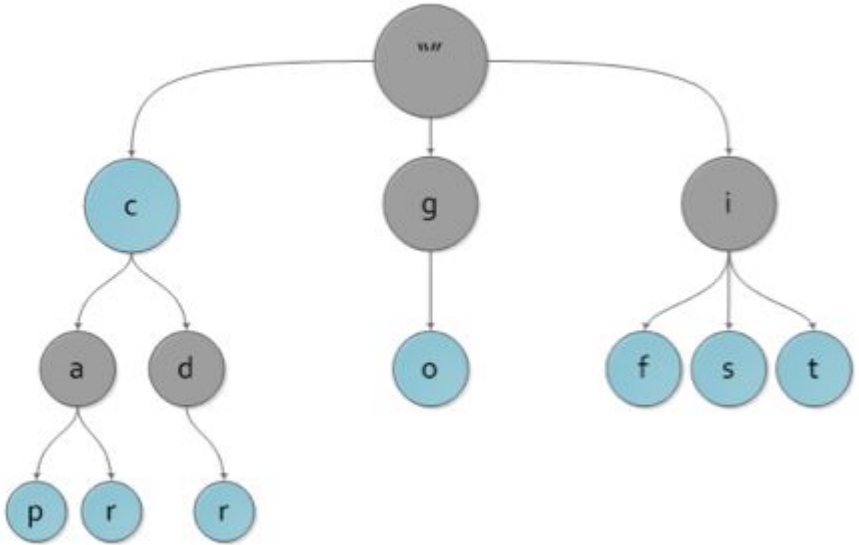
Бор (Префиксное дерево, луч, англ. trie)

Бор для набора образцов **{ *he, she, his, hers* }**

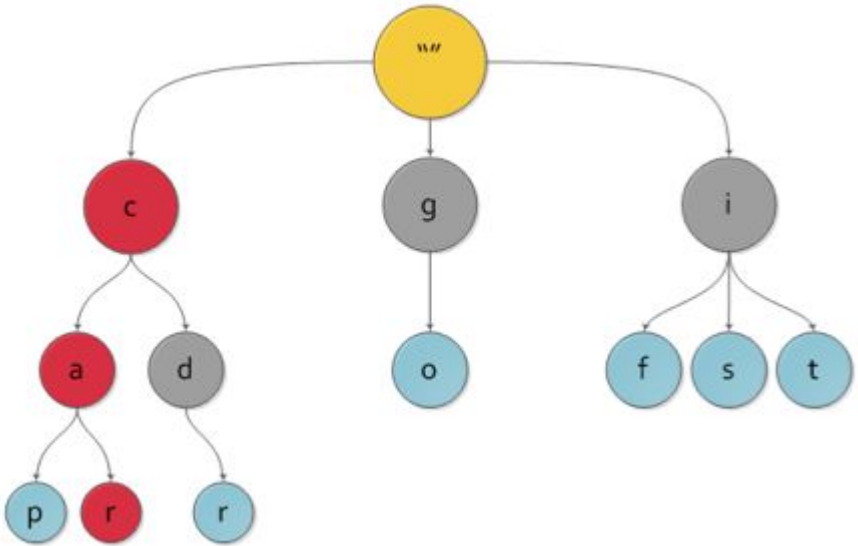


Бор (Префиксное дерево, луч, англ. trie)

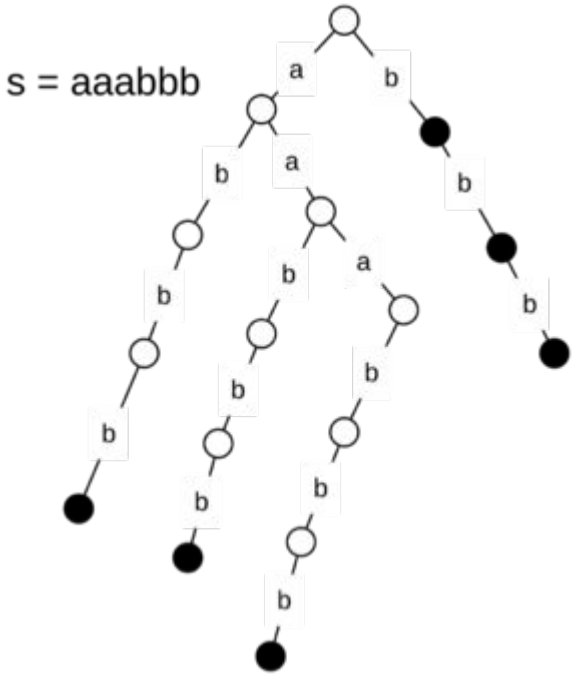
Нагруженного дерева с ключами ***c, cap, car, cdr, go, if, is, it.***



Бор (Префиксное дерево, луч, англ. trie)



Бор (Суфиксное дерево, луч, англ. trie)



Указания для задачи

Для оценки сложности вашего алгоритма можно использовать функции находящиеся в стандартной библиотеки языка питон модуль datetime функция now()

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
```

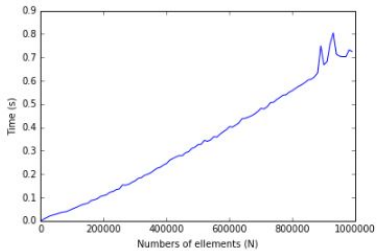
In [7]:
```
times = []
for N in xrange(1000, 1000000, 10000):
    array = np.random.rand(N)
    t1 = datetime.datetime.now()
    sorted(array)
    t_res = datetime.datetime.now() - t1
    times.append([N,t_res])
y = [t[1].total_seconds() for t in times]
x = [t[0] for t in times]
```

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!