Анализ Алгоритмов

Лабораторная работа №8

По теме “*Эффективный поиск по словарю*”

Студент: Юмаев Артур

Группа: ИУ7-55

Оглавление

[Введение 3](#_Toc28098588)

[Цель и задачи 3](#_Toc28098589)

[1. Аналитическая часть 4](#_Toc28098590)

[1.1 Обзор алгоритма оптимизированного поиска по словарю 4](#_Toc28098591)

[Вывод 4](#_Toc28098592)

[2. Технологическая часть 5](#_Toc28098593)

[Вывод 5](#_Toc28098594)

[3. Исследовательская часть 6](#_Toc28098595)

[Вывод 7](#_Toc28098596)

[Заключение 8](#_Toc28098597)

[Литература 9](#_Toc28098598)

# Введение

Поиск по словарю явялется задачей, которая стоит во многих сферах программирования. В словарях хранятся базы данных библиотек, и телефонные данные абонентов в телекомуникационных компаниях. Поиск по словарю является задачей, которую требуется решать быстро, поэтому необходимы методы для оптимизации данной задачи.

## Цель и задачи

Целью данной работы является изучение способа оптимизированного поиска по словарю. Для достижения цели требуется решить следующие задачи.

1. Изучить применение частотного анализа в задаче оптимизированного поиска по словарю.
2. Алгоритмически описать решение задачи оптимизированного поиска по словарю на основе разработанного метода.
3. Реализовать алгоритм оптимизированного поиска по словарю.
4. Сравнить алгоритмы оптимизированного и базового поиска по словарю по времени работы.
5. Сделать выводы о применимости алгоритма оптимизированного поиска к решению задачи поиска по словарю.

# 1. Аналитическая часть

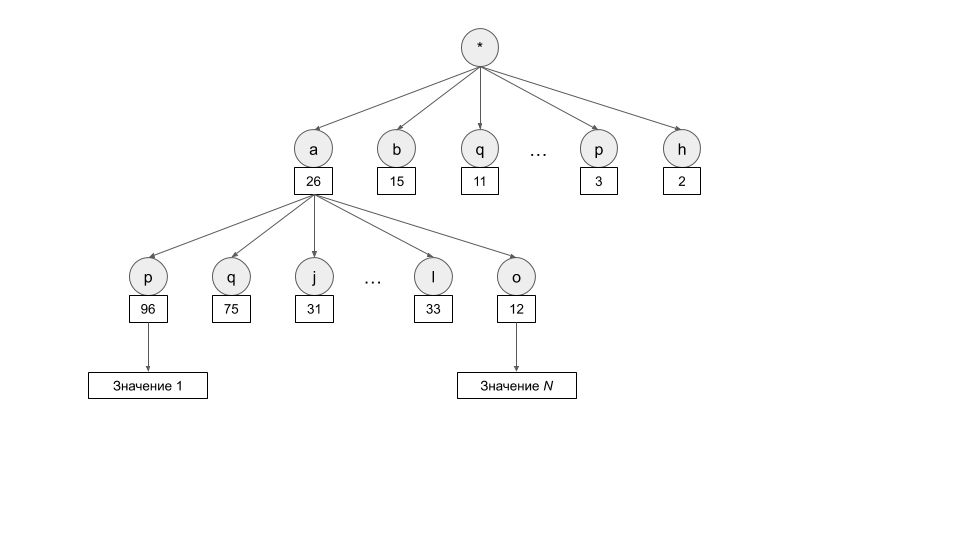
В данном разделе будeт дано полное описание алгоритма оптимизированного поиска по словарю и его алгоритмическое описание.

## 1.1 Обзор алгоритма оптимизированного поиска по словарю

Для начала над словами нужно провести частотный анализ. Для этого нужно взять первый элемент каждого значения в словаре по ключу и подсчитать частотную характеристику, т.е. сколько раз этот элемент встречается в качестве первого элеметна. Таким образом мы повторяем алгоритм для ­*i-*го элемента каждого значения, вычисляя для каждого *i*-го набора частотную характеристику. В качестве примера данного пунка приведена частотная характеристика в формуле 1.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

В качестве следующего пункта словарь сортируется по значению и вставляется в качестве значения в словарь глубины *i -* 1. В конце мы получается похожую структуру словаря.



Далее для большей оптимизации можно применить метод бинарного поиска по дереву, который гарантирует лучшую производительность как .

## Вывод

В данном разделе было приведено общее и алгоритмическое описание алгоритма оптимизированного поиска по словарю и приведен пример.

# 2. Технологическая часть

В качестве языка программирования был выбран Python [3], так как имеется большой опыт работы с ним. На листинге 1 будет приведен алгоритм оптимизированного поиска по словарю.

Листинг 1. Алгоритм формирования оптимизированного словаря

|  |  |
| --- | --- |
|  | names = sorted(names, reverse=True, key=lambda x: len(x))  biggest\_word = names[0]  f\_level = [name[:1] for name in names]  f\_level\_freq = collections.Counter(f\_level)  for k in f\_level\_freq:  f\_level\_freq[k] = [f\_level\_freq[k], []]  for name in names:  f\_level\_freq[name[:1]][1].append(name[1:2])  for k in f\_level\_freq:  f\_level\_freq[k][1] = collections.Counter(f\_level\_freq[k][1])  print(f\_level\_freq[k][1])  f\_level\_freq = {k: v for k, v in  sorted(f\_level\_freq.items(),  key=lambda item: item[1][0],  reverse=True)}  for i in range(len(names)):  if names[i][2:] == '':  f\_level\_freq[names[i][:1]][1][names[i][1:2]] =  [f\_level\_freq[names[i][:1]][1][names[i][1:2]],  values[i]]  else:  if type(f\_level\_freq[names[i][:1]][1][names[i][1:2]]) ==  int:  f\_level\_freq[names[i][:1]][1][names[i][1:2]] =  (f\_level\_freq[names[i][:1]][1][names[i][1:2]], {}) |

## Вывод

В данном разделе был приведен листинг алгоритма оптимизированного поиска по словарю.

# 3. Исследовательская часть

Замеры времени проводились на 64-битной операционной системе Windows 10 и на x64 процессоре Inter Core i7 с 4 гб оперативной памяти для случайно сгенерированных 100,000 слов размером 8. Тестировался метод улучшенного поиска по словарю и метод поиска простым перебором. Результаты эксперимента показали, что метод эффективного поиска ищет слова более чем в 55.93 раза быстрее по сравнению с обычным поиском по словарю. На рисунке 2 приведен график замера скорости работы алгоритма оптимизированного поиска по словарю.

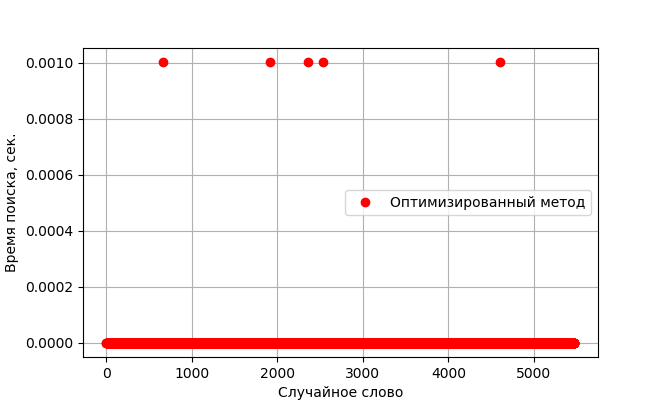


Рисунок 2. Замеры времени алгоритма оптимизированного поиска по словарю

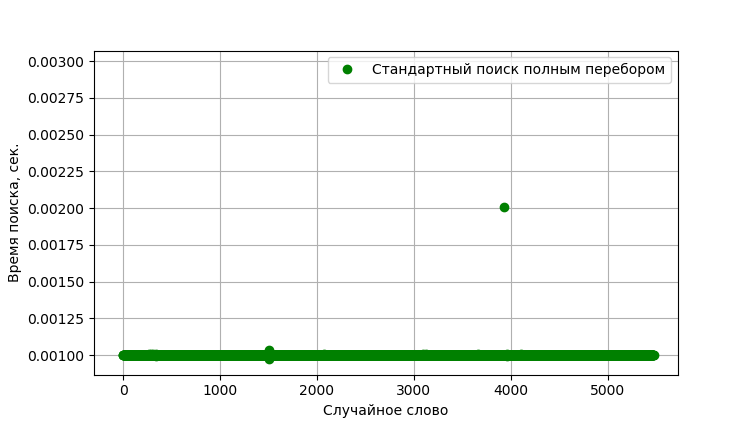


Рисунок 3. Замеры времени алгоритма базового поиска по словарю

По итогам эксперимента было выявлено, что оптимизированный алгоритм поиска работает в среднем лучше \_\_\_ лучше на тех данных, которые были выбраны для исследования. Также алгоритм показал, что метод бинарного поиска невозможно осуществить в условиях данного эксперимента, так как на количестве слов < 5000 он ухудшает производительно в сравнении с полным сравнением. Сделано предположение, что он может быть эффективнее для количества слов > 5000, но в условиях эксперимента не представлялось возможным провести этот эксперимент из-за ограниченности ресурсов вычислительно машины.

## Вывод

В данном разделе было проведено экспериментальное сравнение алгоритмов оптимизированного поиска по словарю и метод полного перебора ключей для поиска по словарю. Эксперимент показал, что на обширной выборке слов размерность 100,000 элементов метод оптимизированного поиска в среднем в 55 раз лучше.

# Заключение

В результате выполнения данной работы были рассмотрены способы решения задачи поиска по словарю с помощью алгоритма оптимизированного поиска поиска и алгоритма поиска полным перебором ключей. Опыт показал, что рекомендуется использовать метод оптимизированного поиска. Было изучено применение частотного анализа в задаче оптимизированного поиска по словарю. Алгоритмически описано решение задачи оптимизированного поиска по словарю на основе разработанного метода. Реализован алгоритм оптимизированного поиска по словарю. Сравнены алгоритмы оптимизированного и базово поиска по словарю по скорости работы. Сделаны выводы о применимости алгоритма оптимизированного поиска к решению задачи поиска по словарю.

# Литература

[1] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134—142, 1991.

[2] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992.