Algorithm 1: Searching

Name: Lucas Jacobs
Pcn: 490692
Class: S4 Academic preparation
Teachers: Simona Orzan
Date: 26-02-2023
Word count: 816

Contents

Code

Read a file code

```python
#in this function a file path will be put in
#it will read the file as a string with file.read and with the split function it will seperate the string in a list of words
#returns a list of words
def readFile(filename):
    with open(filename, 'r') as file:
        words = file.read().split()
    return words
```

Linear search code

```python
#for each word in the list it will check if the argument given in the function matches the word
#it will return the amount of how much the given argument is in the textfile
def LinearCount(search_word):
    amount_of_words = 0
    words = readFile('ordered_words100mil.txt')
    for word in words:
        if(word == search_word):
            amount_of_words +=1
    return amount_of_words
```

Binary search code

```python
#it uses binary search on a ordered list of words
#if the given search_word is found it will return the amount of how much this word occured
#otherwise it returns 0
def BinaryCount(search_word):
    words = readFile('ordered_words100mil.txt')
    words.sort()

    low = 0
    high = len(words) - 1
    amount_of_words = 0

    while low <= high:
        mid = (low + high)//2
        if words[mid] == search_word:
            left = mid -1
            right = mid + 1
            amount_of_words = 1
            while left >= 0 and words[left] ==search_word:
                amount_of_words +=1
                left -= 1
            while right < len(words) and words[right] == search_word:
                amount_of_words += 1
                right += 1
            break
        elif words[mid] < search_word:
            low = mid + 1
        else:
            high = mid - 1
    return amount_of_words
```

Search performance algorithms

Unordered text file

In this test, we want to see if a linear or binary search is faster. To do this test I generated unordered text files with hundred thousand words, one million, and ten million. Also not to forget, every test will be done with the same word, both with linear and binary search.

Linear search times

The following times when performing the different files with the linear search:

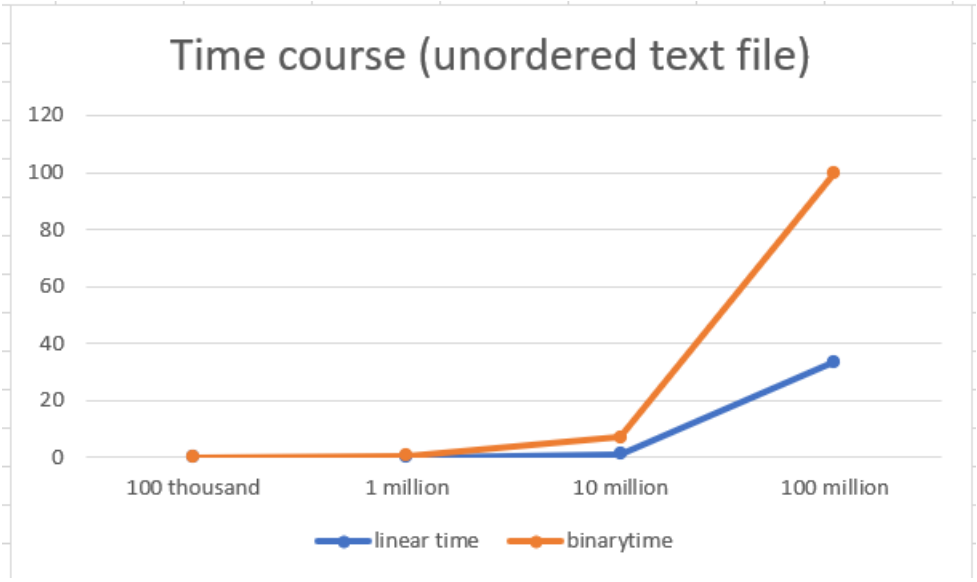
- 100 thousand words: 0.0 seconds
- One million words: 0.1 seconds
- Ten million words: 1.2 seconds
- 100 million words: 33.4 seconds

Binary search times

The following times when performing the different files with the binary search:

- 100 thousand words: 0.0 seconds
- One million words: 0.5 seconds
- Ten million words: 7.1 seconds
- 100 million words: 99.9 seconds

Time plot



Note the y-axis is time in seconds and the x-axis is the words in the file.

Ordered text file

In this test, we again want to see whether the linear or binary search is faster. This time it will be done with an ordered text file. Tests will be done with files with 100 thousand, one million, ten million, and 100 million words.

Linear search times

The following times when performing the different files with the linear search:

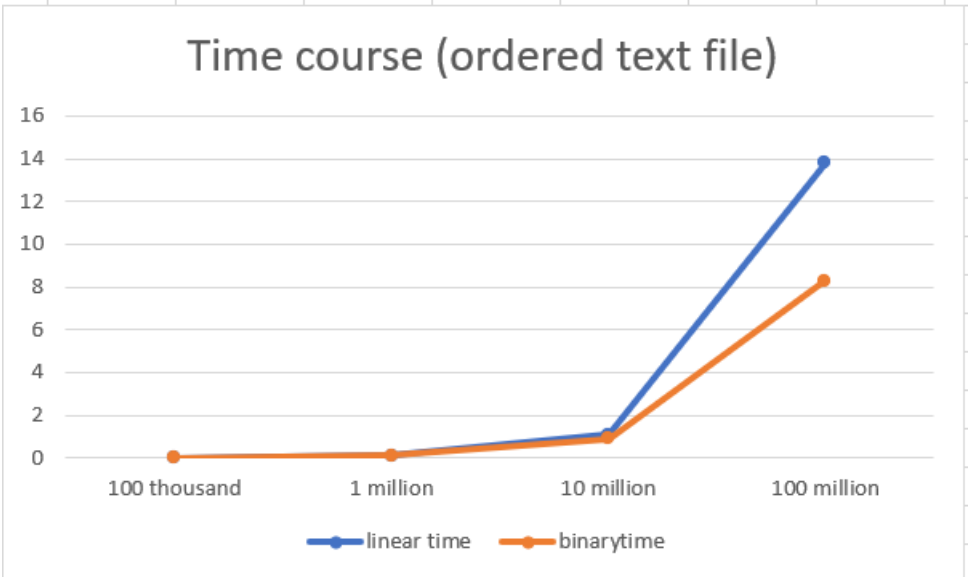
- 100 thousand words: 0.0 seconds
- One million words: 0.1 seconds
- Ten million words: 1.1 seconds
- 100 million words: 13.8 seconds

Binary search times

The following times when performing the different files with the binary search:

- 100 thousand words: 0.0 seconds
- One million words: 0.1 seconds
- Ten million words: 0.9 seconds
- 100 million words: 8.3 seconds

Time plot



Note the y-axis is time in seconds and the x-axis is the words in the file.

Conclusion

In the first test, we can clearly see that linear searching is much faster compared to binary searching when using an unordered text file. At first, you maybe think binary is much faster because it has a time complexity of O(log(n)) compared to a time complexity of O(n) with a linear search.
However, to do a binary search you first need to sort it in alphabetical order when looking at a text file. This sorting has a time complexity of O(nlog(n)) which slower than O(n). Therefore when using an unordered text file it will be better to use a linear search.
To continue, in the second test the text file is already ordered, which means that binary search is a lot quicker. This is because Binary search now has a time complexity of O(log(n)) and linear search still has an O(n) time complexity.
To conclude, when you know the text file is ordered it will be better to use a binary search to find a certain word. But when using an unordered text file (normal text files) linear search is much faster.

Maximum size handling

The test that I did with both linear and binary search where handling verily well. But when testing on bigger files it would be a problem due to the size of the text file, it would be too big to have on my laptop.
But to answer this properly, there are a few factors that come to play. First of all let's look at binary search, how much memory do you have and what capacity do you have on the device? This is because the binary search will be loaded on the memory, when it can handle it the binary search will be efficient. But when the file exceeds the memory, the search can become slow and very intense because it has to swap data between memory and the storage of your device.
Secondly, when you have an unordered text file, for the binary search you first have to sort it which can take quite some time.
For linear searching, it can be suitable for using it when you have a small file when you have an unsorted list. Furthermore, when your file is for example stored on a slow disk, it can be really slow when having a big file.

Diversity of vocabulary

When you have a text file with large words and a very different vocabulary it will be much slower and it uses a lot more memory, compared to when you have a small and not that complex vocabulary.