IN3120/4120 - Search technology

Group 2, Third session
TA: Markus Sverdvik Heiervang

Today’s plan

- Brief coverage of suffix arrays

*** Full digital lab session ***

- Assignment discussion
- Topic discussion

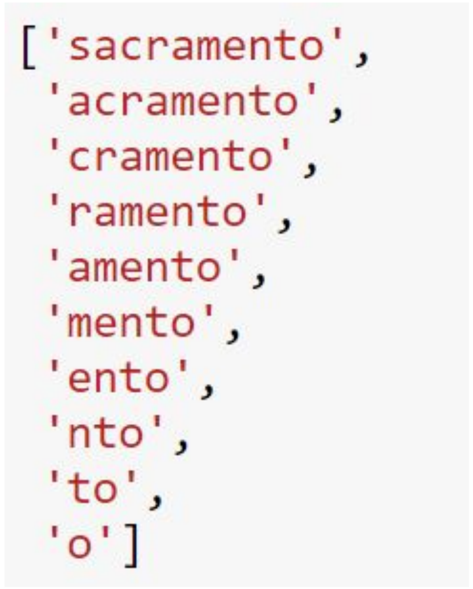
- 1-to-1 help (breakout rooms)
- Queue system

Suffix arrays

- A suffix is any non-empty substring that a string s ends on
  - Example: “apple” ends on “e”, “le“, “ple”,“pple”, “apple” (yes we also consider apple to be a substring of itself)
- A suffix array is an efficient data structure for searching with a prefix:
  - It is a sorted array of strings (suffixes) (construction == O(n log n))
  - It uses binary search to find the matches in O(log n) time
  - E. g. “how do i” can give results
    - “how do i put on my socks”
    - “how do i download ram”
    - “how do i construct a suffix array”
  - In a way, it can finish your query sentence from the documents in the corpus

Suffix arrays

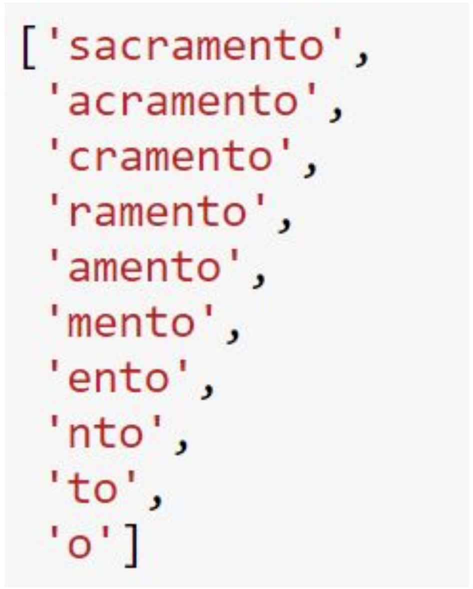
Let’s create a suffix array for the word “sacramento”



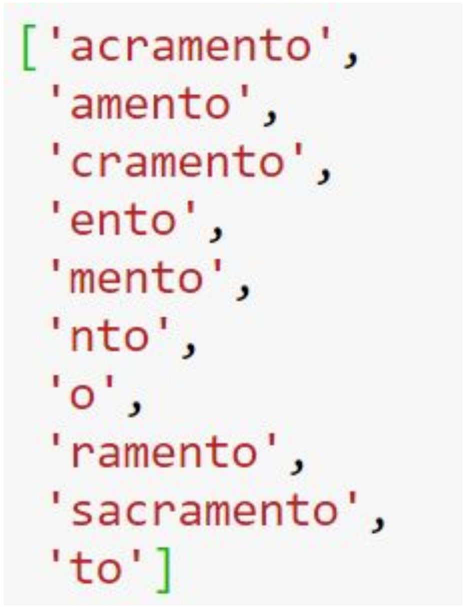
```
['sacramento',
 'acramento',
 'cramento',
 'ramento',
 'amento',
 'mento',
 'ento',
 'nto',
 'to',
 'o']
```

Suffix arrays

Let’s create a suffix array for the word “sacramento”



```
['sacramento',
 'acramento',
 'cramento',
 'ramento',
 'amento',
 'mento',
 'ento',
 'nto',
 'to',
 'o']
```



```
['acramento',
 'amento',
 'cramento',
 'ento',
 'mento',
 'nto',
 'o',
 'ramento',
 'sacramento',
 'to']
```

- We sort it in lexicographical (alphabetical) order

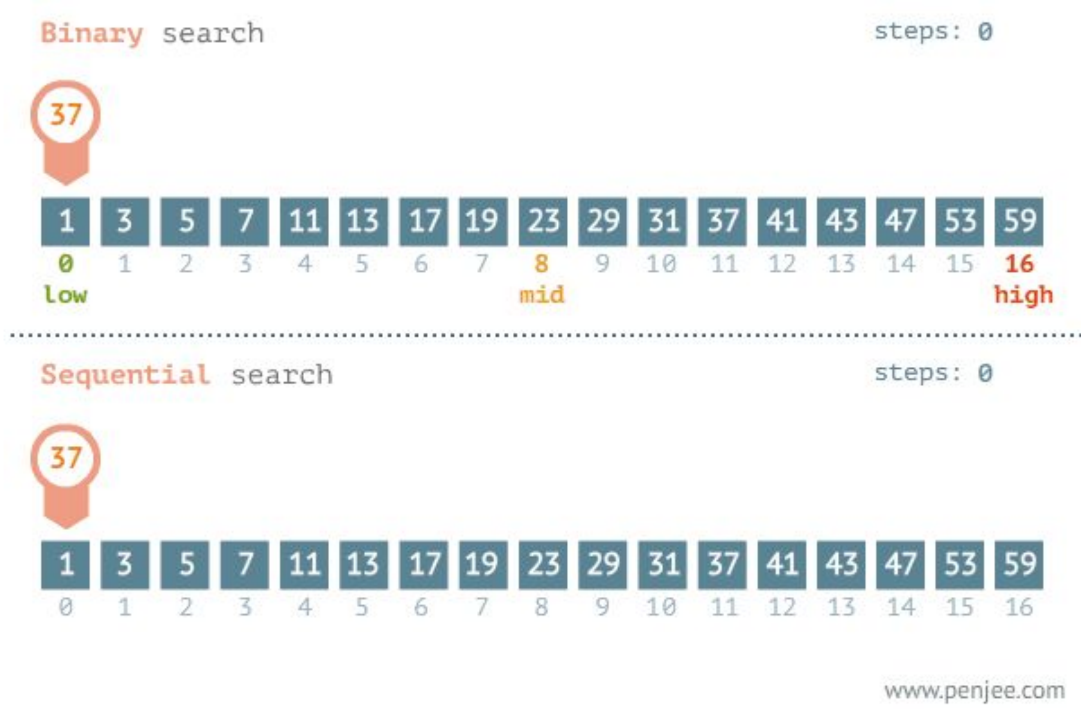
Suffix arrays

- A sorted array lets us do binary search
- Binary search recap

Say i was to think of a number between 0 and 100.

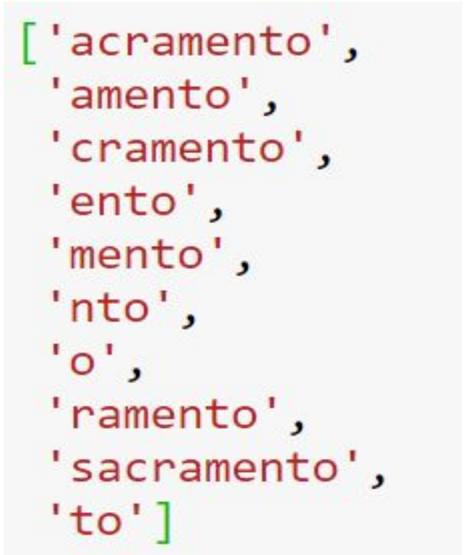
You had to guess the number in as few steps as possible.

- Using binary search, you guess 50.
- I say lower.
- You guess 25.
- I say higher.
- you guess 37...


Binary search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
low
mid
high
Sequential search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
www.penjee.com

Searching in a suffix array

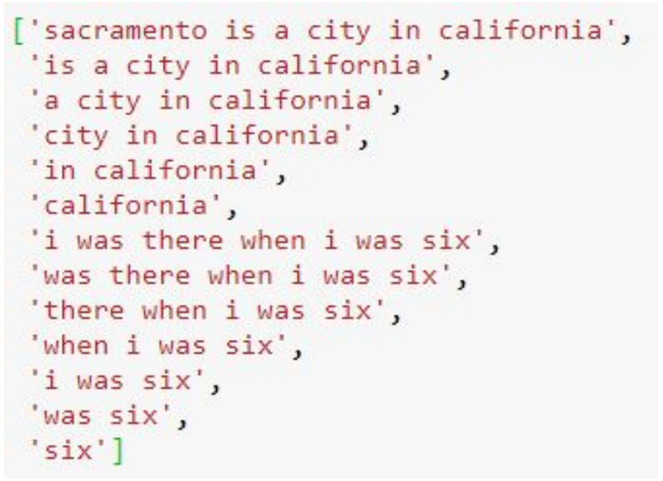
- We do the same thing, but with string comparison instead of numbers (a < ab < ac < b < ba)



```
['acramento',
 'amento',
 'cramento',
 'ento',
 'mento',
 'nto',
 'o',
 'ramento',
 'sacramento',
 'to']
```

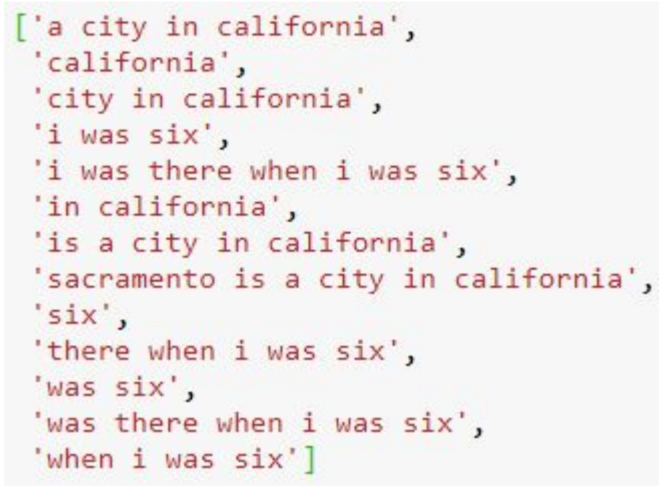
Suffix arrays on term level

“Sacramento is a city in California. I was there when i was six”



```
['sacramento is a city in california',
 'is a city in california',
 'a city in california',
 'city in california',
 'in california',
 'california',
 'i was there when i was six',
 'was there when i was six',
 'there when i was six',
 'when i was six',
 'i was six',
 'was six',
 'six']
```

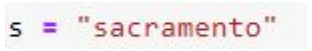
$\longrightarrow$



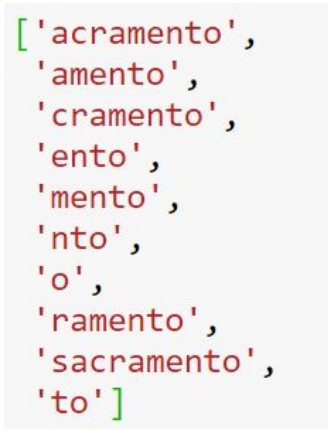
```
['a city in california',
 'california',
 'city in california',
 'i was six',
 'i was there when i was six',
 'in california',
 'is a city in california',
 'sacramento is a city in california',
 'six',
 'there when i was six',
 'was six',
 'was there when i was six',
 'when i was six']
```

Efficient storing of suffixes

- In suffix arrays you don't actually store the suffixes as strings, as they are vastly inefficient. You store the indexes of the start and end of the string in tuples, that can be converted back to the string if needed
- Example:

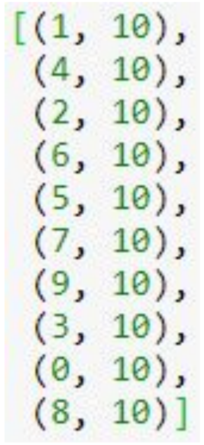


```
s = "sacramento"
```



```
['acramento',
 'amento',
 'cramento',
 'ento',
 'mento',
 'nto',
 'o',
 'ramento',
 'sacramento',
 'to']
```

$\longrightarrow$

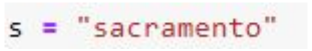


```
[(1, 10),
 (4, 10),
 (2, 10),
 (6, 10),
 (5, 10),
 (7, 10),
 (9, 10),
 (3, 10),
 (0, 10),
 (8, 10)]
```

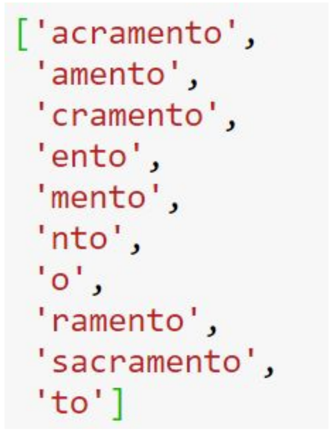
Efficient storing of suffixes

- In suffix arrays you don’t actually store the suffixes as strings, as they are vastly inefficient. You store the indexes of the start and end of the string in tuples, that can be converted back to the string if needed
- Example:

(In this specific example, only the starting index is necessary)

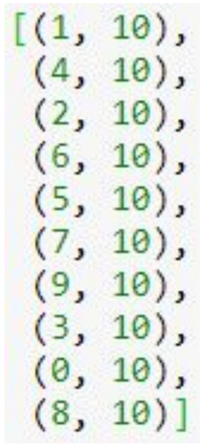


```
s = "sacramento"
```



```
['acramento',
 'amento',
 'cramento',
 'ento',
 'mento',
 'nto',
 'o',
 'ramento',
 'sacramento',
 'to']
```

$\longrightarrow$



```
[(1, 10),
 (4, 10),
 (2, 10),
 (6, 10),
 (5, 10),
 (7, 10),
 (9, 10),
 (3, 10),
 (0, 10),
 (8, 10)]
```