Programming in R and Python

Lecture 9: Data structures in Python

Adam Gudyś
Silesian University of Technology
2020

Agenda of the Python lectures

- Data structures
- Linear algebra
- Parallel computing
- Object-oriented programming
- Data analysis and visualization
- Image processing and computer vision

Types of data structures

Immutable – content cannot be altered:

- simple types (`int, float, bool`),
- string (`str`),
- tuple.

M*utable* – content can be altered:

- list, set, dictionary.

Code...

List implementation

Python list is implemented as...

dynamic array of pointers.

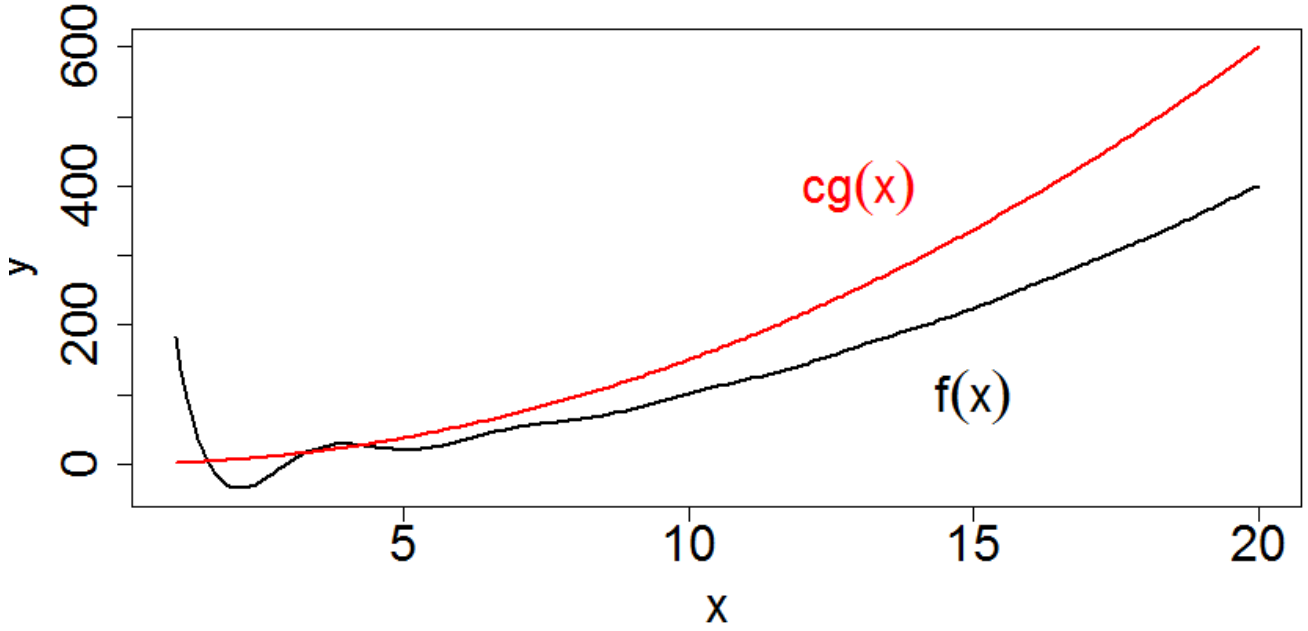
Worst-case time complexity of operations:

- access: $O(1)$,
- search: $O(n)$,
- insertion: $O(n)$ (possible reallocation).

Big O notation

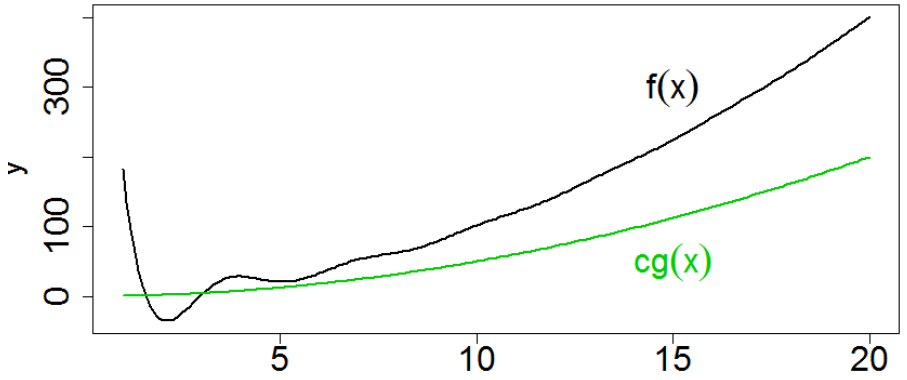
Specifies how fast the function grows with growing argument.

$f(x) \in O\big(g(x)\big)$: there exists $c > 0$ and x_0 such that $f(x) \leq cg(x)$ for $x \geq x_0$, i.e. , $f(x)$ is upper-bounded by $g(x)$.

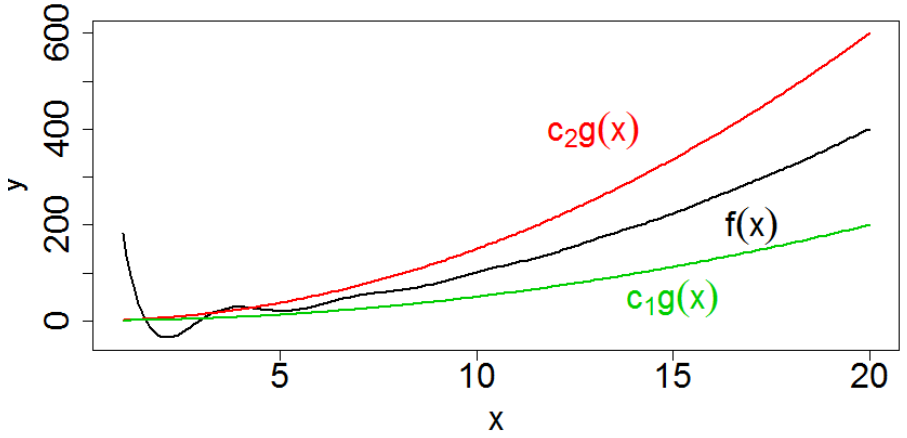


Big O notation

$f(x) \in \Omega\big(g(x)\big)$: $f(x)$ is lower-bounded by $g(x)$.



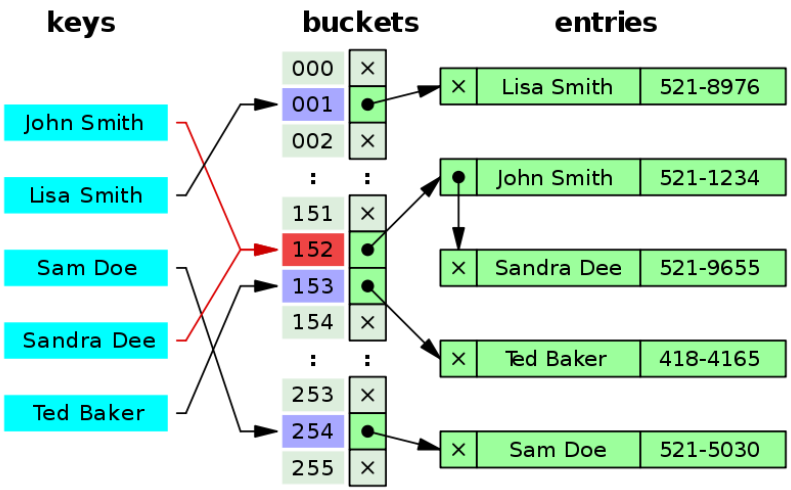
$f(x) \in \Theta\big(g(x)\big)$: $f(x)$ is upper- and lower-bounded by $g(x)$.



Example complexities

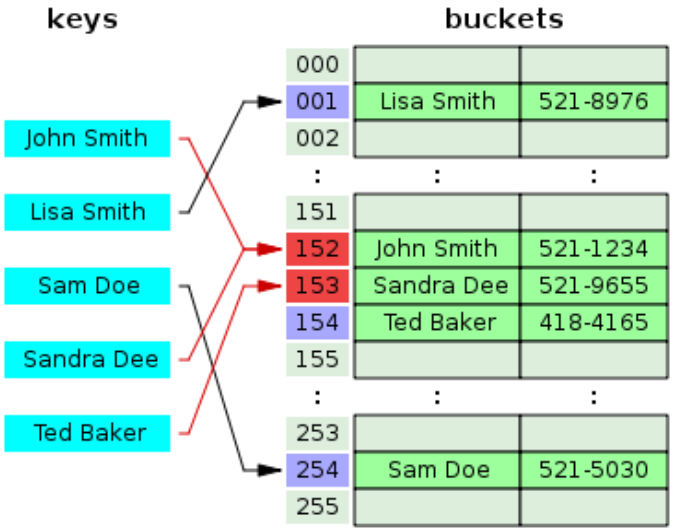
Algorithm	Time complexity
finding smallest element in a sorted array	$O(1)$
binary search	$O(\log n)$
linear search	$O(n)$
sorting: bubble, insertion, selection	$O(n^2)$
sorting: merge, heap, quick	$O(n \log n)$
sorting: radix	$O(n)$
matrix multiplication: naive	$O(n^3)$
matrix multiplcation: Strassen's	$O(n^{2.8074})$
breaking a binary password: exhaustive	$O(2^n)$
traveling salesman problem: exhaustive	$O(n!)$

Hashtable



Chaining:

- conflicts solved by lists,
- hash function $h(x)$,
- easy elements removal.



Open addressing:

- conflicts solved by probing,
- hash function $h(x, i)$,
- special value for removed elements,

Code again...

Thank you for your attention!