Data Structures and Algorithms with Python

Heikki Peura

h.peura@imperial.ac.uk

Merge sort

Merge sort

Divide-and-conquer:

- Identify smallest possible “base case” subproblems that are easy to solve
- Divide large problem and solve smaller subproblems
- Find smart way to combine subproblem solutions to solve larger problems

Applying to sorting:

- Base case: if list length $n < 2$, the list is sorted
- Dividing: if list length $n \geq 2$, split into two lists and recursively sort each, then combine results
- Combining (merging) two lists:
  - Start with empty result list
  - Look at first element of each list, add smaller to end of result, repeat while both lists have elements
  - When one list becomes empty, copy the rest of the other list

Merging

Dividing

56 24 99 32 9 61 57 79

56 24 99 32   9 61 57 79

56 24   99 32   9 61   57 79

56   24   99   32   9   61   57   79

Merging

24 56   32 99   9 61   57 79

24 32 56 99   9 57 61 79

9 24 32 56 57 61 79 99

Merge sort complexity

What is the complexity of merge? Two lists of lengths n_1, n_2:

- Two lists of lengths n_1 and n_2: $O(n_1 + n_2)$ copy operations / comparisons (need to copy each item and each copying follows a single comparison)
- $O(n)$ per level of recursion

Mergesort complexity = merging * # levels of recursion

- Number of recursion levels $O(\log n)$ (like binary search)
- Log-linear: $O(n \log n)$
- Big improvement over selection sort!
- Does need some more space due to copying lists

Complexity classes

Fast algorithm: worst-case running time grows slowly with input size

- $O(1)$: constant running time — primitive operations
- $O(\log n)$: logarithmic running time — binary search
- $O(n)$: linear running time — linear search
- $O(n \log n)$: log-linear time — merge sort
- $O(n^c)$: polynomial running time — selection sort
- $O(c^n)$: exponential running time — ??

Sorting is a canonical algorithms problem

Many algorithms exist: bubble sort, insertion sort, quick sort, radix sort, heap sort, ...

- Useful for developing algorithmic thinking – eg randomized algorithms

Theoretical bound for worst-case performance is $O(n \log n)$ – **we can't do better than merge sort**

But other algorithms are **better on average**

- Python uses **timsort** (In 2002, a Dutch guy called Tim got frustrated with existing algorithms)
- Exploit the fact that lists tend to be partly sorted already