

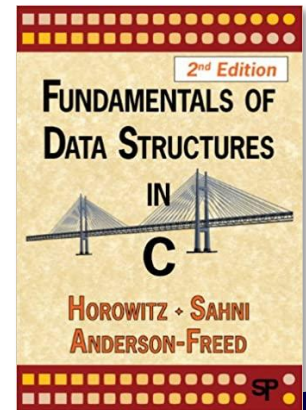
Data Structure

Sorting

Shin Hong

May 19, 2020

Ch. 7 Sorting



Motivation – Search

2

- checking whether or not an item exists in a list of n items
 - sequential search:
check each item by iterating over the list
 - binary search:
assume that the list is sorted ;
compare the middle point of the list with the target item,
and continue the binary search on a half of the list where
the item may be found

Sorting Problem

3

- Given a list of n items (a_1, a_2, \dots, a_n) , find a permutation σ
 $(a_{\sigma 1}, a_{\sigma 2}, \dots, a_{\sigma n})$ s.t. $\text{key}(a_{\sigma i}) \leq \text{key}(a_{\sigma i+1})$ for $1 \leq i < n$
 - assume that the equivalence and the ordering in items are well defined
- A sorting is stable if the permutation satisfies the following condition:
if $\text{key}(a_{\sigma i}) = \text{key}(a_{\sigma j})$ and $i < j$, $\sigma_i < \sigma_j$

Insetion Sort - Idea

- Given an unsorted list U of items, repeat the following two steps until the list becomes empty:
 - (1) remove a minimum/maximum item in the U
 - (2) insert the removed item to the sorted list
- We can use a prefix $U[0 .. i]$ as a sorted list for i -th turn to make sorting happen within the given list

Insetion Sort - Algorithm

5

- Algorithm

Input: a given a list of N items, $L[0 .. N-1]$

Output: a sorted list of item L

Procedure

for i in $[0 .. N-2]$

 find j such that $L[j]$ is minimum among $L[i .. N-1]$

 swap $L[i]$ and $L[j]$

end for

- Example

(7, b)	(3, d)	(2, e)	(9, a)	(5, g)	(3, a)	(2, h)
--------	--------	--------	--------	--------	--------	--------

--	--	--	--	--	--	--

Sorting

Data Structure

2020-05-26

Insertion Sort - Analysis

6

- Algorithm

Input: a given a list of N items, $L[0 .. N-1]$

Output: a sorted list of item L

Procedure

for i in $[0 .. N-2]$

 find j such that $L[j]$ is minimum among $L[i .. N-1]$

 swap $L[i]$ and $L[j]$

end for

- Time complexity

$$O\left(\sum_{i=0}^{N-2} (i + 1)\right) = O(n^2)$$

Quick Sort

7

- Given a list of items $L[0 .. N-1]$,
reorder L such that all elements in $L[0 .. p]$ are less than
equal to all elements in $L[p+1 .. N-1]$, and
sort two sublists $L[0.. p]$ and $L[p+1 .. N-1]$ independently

- Algorithm

Sort ($L[0 .. N-1]$, $left$, $right$)

if $left == right$ **then return**

$p = left$; $l = left + 1$; $r = right$

while $l < r$ **begin**

while $L[l] < L[p] \wedge l \leq right$ **begin** $l++$ **end while**

while $L[p] < L[r] \wedge left \leq r$ **begin** $r--$ **end while**

if $l < r$ **then** swap $L[l]$ and $L[r]$; $l++$; $r--$

end while

swap $L[p]$ and $L[r]$

Sort (L , $left$, $r - 1$)

Sort (L , r , $right$)

Sorting

Data Structure

2020-05-26

Quick Sort - Example

8

5	3	1	9	7	3	2	2
---	---	---	---	---	---	---	---

--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--

Sorting

Data Structure

2020-05-26

Quick Sort - Analysis

9

Sort ($L[0 \dots N-1]$, $left$, $right$)

$$T(N) \leq cN + 2T(N/2)$$

if $left == right$ **then return**

$p = left$; $l = left + 1$; $r = right$

while $l < r$ **begin**

while $L[l] < L[p] \wedge l \leq right$ **begin**

$l++$ **end while**

while $L[p] < L[r] \wedge left \leq r$ **begin**

$r--$ **end while**

if $l < r$ **then**

 swap $L[l]$ and $L[r]$

$l++$; $r--$

end if

end while

swap $L[p]$ and $L[r]$

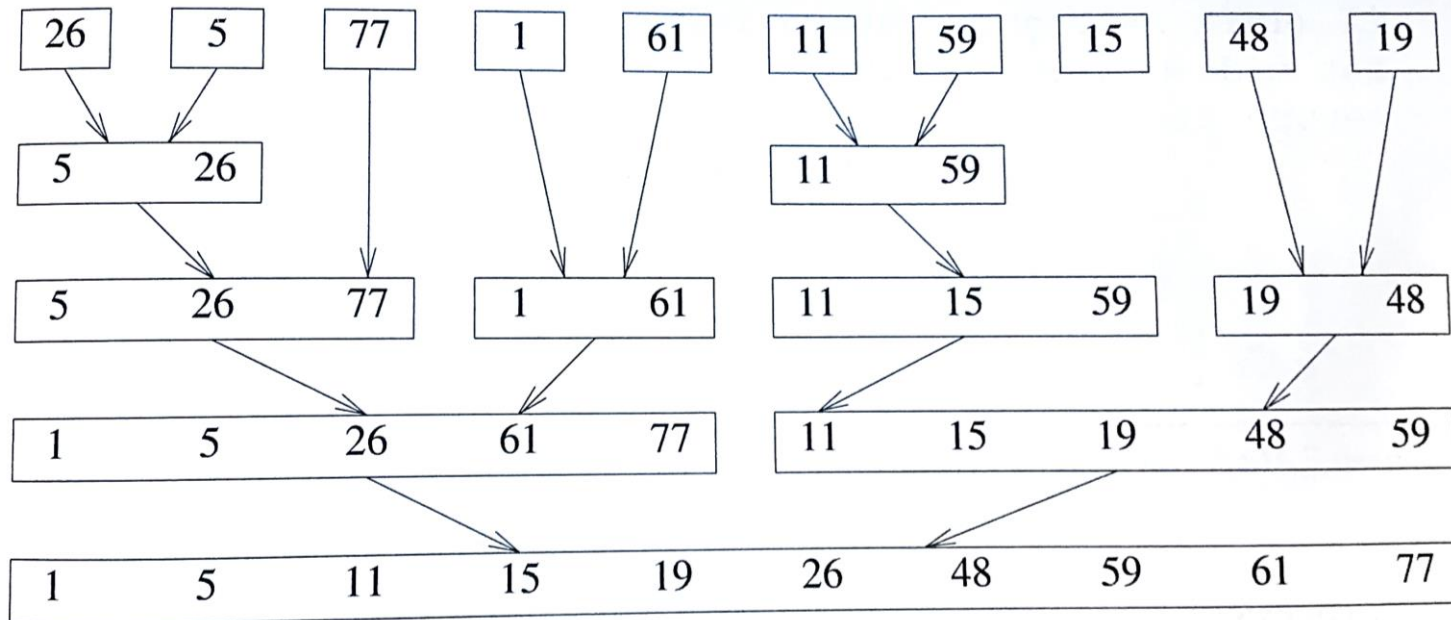
Sort (L , $left$, $r - 1$)

Sort (L , r , $right$)

Merge Sort

10

- Example



Sorting

Data Structure

2020-05-26

Merge Sort

11

Sorting

Data Structure

2020-05-26