

COS 226 Data Structures and Algorithms

Spring 2016 - Flipped Lecture Handout

Week 3 Flipped Activities

1. Sorting Invariants

The column on the left is the original input of strings to be sorted or shuffled; the column on the right are the strings in sorted order; the other columns are the contents at some intermediate step during one of the algorithms listed below. Match up each algorithm. You may not be able to classify some of them. Leave them as cannot determine.

COS	ARC	ARC	ARC	COS	ART	CHM	CHE	REL	ARC
PHY	CHE	CHE	ART	COS	CEE	ART	COS	PHY	ART
ELE	COS	COS	CEE	ELE	CHM	ARC	CHM	PHY	CEE
COS	COS	COS	CHE	PHY	ARC	CEE	COS	ELE	CHE
MAT	ECO	ECO	CHM	ARC	COS	CHE	COS	PHI	CHM
MOL	ELE	EEB	COS	LIN	CHE	COS	ART	ORF	COS
LIN	GEO	ELE	COS	MAT	EEB	COS	CEE	ORF	COS
ARC	LIN	ELE	COS	MOL	COS	COS	ARC	COS	COS
ECO	MAE	ENG	COS	CHE	COS	COS	COS	ELE	COS
CHE	MAT	GEO	COS	ECO	COS	COS	COS	EEB	COS
MAE	MOL	LIN	COS	GEO	EEB	COS	MAE	MUS	COS
GEO	PHY	MAE	ECO	MAE	COS	ORF	GEO	GEO	ECO
ORF	ORF	MAT	ORF	EEB	COS	EEB	ORF	ORF	EEB
EEB	EEB	MOL	EEB	ELE	ELE	ENG	EEB	MAT	EEB
ENG	ENG	ORF	ENG	ENG	MAE	ELE	ENG	LIN	ELE
ELE	ELE	PHY	ELE	ORF	ELE	GEO	ELE	COS	ELE
COS	COS	ART	MOL	CEE	ECO	ELE	ECO	COS	ELE
ELE	ELE	CEE	ELE	COS	ENG	MAE	ELE	ECO	ENG
CEE	CEE	COS	ELE	EEB	MAT	EEB	LIN	CEE	GEO
EEB	EEB	EEB	EEB	ELE	LIN	ECO	EEB	CHE	LIN
ART	ART	ELE	PHY	ART	ELE	MUS	MOL	ART	MAE
MUS	MUS	MUS	MUS	MUS	MUS	PHI	MUS	MAT	MAT
PHI	PHI	ORF	PHI	ORF	MAT	ORF	PHI	MAE	MAT
ORF	ORF	PHI	ORF	PHI	ORF	LIN	ORF	ELE	MOL
COS	COS	COS	GEO	COS	GEO	PHY	MAT	COS	MUS
PHY	PHY	PHY	PHY	COS	ORF	MOL	PHY	MOL	ORF
COS	COS	COS	LIN	MAT	MOL	MAT	COS	COS	ORF
MAT	MAT	MAT	MAT	PHY	PHY	MAT	MAT	EEB	ORF
CHM	CHM	CHM	MAT	CHM	ORF	ORF	ELE	CHM	PHI
ORF	ORF	ORF	ORF	COS	PHY	ELE	ORF	ENG	PHY
COS	COS	COS	MAE	ORF	PHI	REL	PHY	COS	PHY
REL	REL	REL	REL	REL	REL	PHY	REL	ARC	REL
---	---	---	---	---	---	---	---	---	---
0									1

(0) Original input (1) Sorted (2) Selection sort (3) Insertion sort (4) Mergesort (top-down) (5) Mergesort (bottom-up) (6) Quicksort (standard, no shuffle) (7) Quicksort (3-way, no shuffle) (8) Cannot determine

2. Parsimonious Sorting Algorithms

A sorting algorithm is parsimonious if no pair of items is compared more than once. Circle the following sorting algorithms (as implemented in lecture and the textbook) if they are parsimonious; cross them out if they are not parsimonious.

- (a) insertion sort
- (b) selection sort
- (c) top-down mergesort

3. Sorting Equal Keys

Suppose that you are sorting an array containing the following 7 equal keys (the subscript is not part of the key; its purpose is to uniquely identify each of the equal keys).

$A_0 A_1 A_2 A_3 A_4 A_5 A_6$

What is the result of running the standard version (from the Sedgewick textbook) of each of the following sorting algorithms?

- (a) Insertion Sort
- (b) Selection Sort
- (c) 2-way Quick Sort
- (d) 3-way Quick Sort
- (e) top-down merge sort
- (f) bottom-up merge sort

4. More Sorting

- (a) Modern computers have memory caches, which speed up reads and writes if they are to locations near recently-accessed memory. This makes sequential access to memory faster, in general, than random access. Circle the sorting algorithm below that you would expect to benefit least from caching?

insertion sort mergesort quicksort

- (b) Modern computers also have multiple processes and can take advantage of parallel computing. Circle the sorting algorithm below that you would expect to benefit from parallelism?

insertion sort mergesort quicksort

- (c) You are managing the accounts for BigIBankCo, and have an array of customers together with their balances. You would like to rearrange the array such that the richest customers (those with balances greater than 1 million) are grouped at the beginning, with everyone else at the end.

Describe an algorithm for performing this task in linear time, and using only constant extra memory. Adhering to the spirit of code reuse, adapt an algorithm from class and describe only the changes you would make.

5. Identifying Sorting Algorithms

Awaking drenched in sweat one night, you clearly see your path to fame and fortune. You will build a robotic rhinoceros and tour the country singing songs about nature to children, who will be allowed to play and interact with the rhinoceros. While a real rhinoceros would be too dangerous, you believe a rhinoceros can be kept in check. In each of the situations below, which sort would you use? In all cases, assume memory is not an issue, and that the goal is to minimize run time so that the rhino can react as quickly as possible to any potential trouble. Choose from Mergesort, Insertion sort, Selection sort, Knuth shuffle. Answers may be used many times.

- (a) The rhinoceros is outfitted with a large number of sensors, each of which generates objects of type `Observation`. Observations include many instance variables, taken at fixed time intervals, including importance, timestamp, pressure, temperature, light intensity, etc. These are placed in an unsorted array, and every time 1,000,000 `Observations` are generated, they are delivered to a central processing unit that sorts the `Observations` by the importance field, which is of type `double`. What sort should you use to minimize the run time required to sort all `Observations` by importance?
- (b) Due to some close calls, you're going to refactor the sorting process to deal with a rare but dangerous situation where some `Observations` are generated with an incorrect importance value. For engineering reasons not described here, you can detect these by sorting by the timestamp and importance of each `Observation`. Instead of importance, you first want to sort by the timestamp of each `Observation`. The timestamp is of a comparable type called `DateTime`. What sort should you use to minimize the run time required to sort all 1000000 `Observations` by timestamp?
- (c) After sorting by time stamp, you want to sort by importance such that all the objects of the same timestamp stay clustered. What sort should you use to minimize the run time while maintaining this clustering?
- (d) You iterate through the array, update the importance of the very rare bad `Observations` with a new value, and sort once more. What sort do you use to put items in order of importance while minimizing run time?