ID1020 2018-09-12 #5

Sorting (record, key)

Target: Sorting any type of data

We need to be able to compare objects
The sorting method should be the same for all types

Total order:
- antisymmetric: $v \leq w, w \leq v \rightarrow v = w$
- transitive: $v \leq w, w \leq x \rightarrow v \leq x$
- total: $v \leq w$ or $w \leq v$ or both

Non-transitive
Stone-scissors-paper

Callback = a reference to executable code
- The client passes an array of object to sort()
- sort() calls compareTo() on the objects that should be sorted

implements comparable, compareTo() returns pos/neg/zero int.
negative if "less" than compared object
positive if "larger" than compared object
zero if "equal" to compared object

Double.compareTo() is non-transitive ($-0.0 \neq 0.0$; NaN)

Selection sort: Find "smallest", put first, repeat (N^2)
(swap)

Insertion sort: Swap if "left" element is "larger" than "right" element
Worst case N^2 Best case N, Better if almost sorted

Partially sorted if #inversions $\leq cN$, c is a constant
(almost)

Binary insertion sort, binary search to find "insertion point" in sorted part of array.

Shellsort

h-sorting (h-sorted array is h interleaved sorted subsequences) $h \rightarrow 1$

Worst case N^2 ($N\log^2 N$ for best gap sequence) Best case $N\log N$
No complete mathematical analysis has been found.
Not efficient for big arrays because of caching

(Radix sort)

Shuffling

Randomize sorting order

Shuffle sort Generate one random number for each element

Pseudo Random Number Generators
seed - Same seed → same sequence
large prime seed is often best.
quality of number generators vary.

(Microsoft - shuffle browsers) (IE to right 50% of time)

Knuth shuffle swap a[i] with random a[r]
Not completely random

"The generation of random numbers is to important to be left to chance!"
- Robert R. Coveyou
Hardware solutions are generally better.