Computational Complexity - Sorting

Week 15

Required Activities

- Check Announcements regularly (every 2-3 days)
- Read FA book: Chapter 7
- Keep working on Assignment 5 (due Feb 15)

Computational Complexity

- Study of all possible algorithms that solve a given problem
- Tries to determine a lower bound on efficiency of all algorithms for a given problem (tells you what's the best time complexity that an algorithm could potentially achieve)
- Problem analysis as opposed to algorithm analysis
- Can attach problem two ways: develop more efficient algorithm or prove that more efficient one is not possible

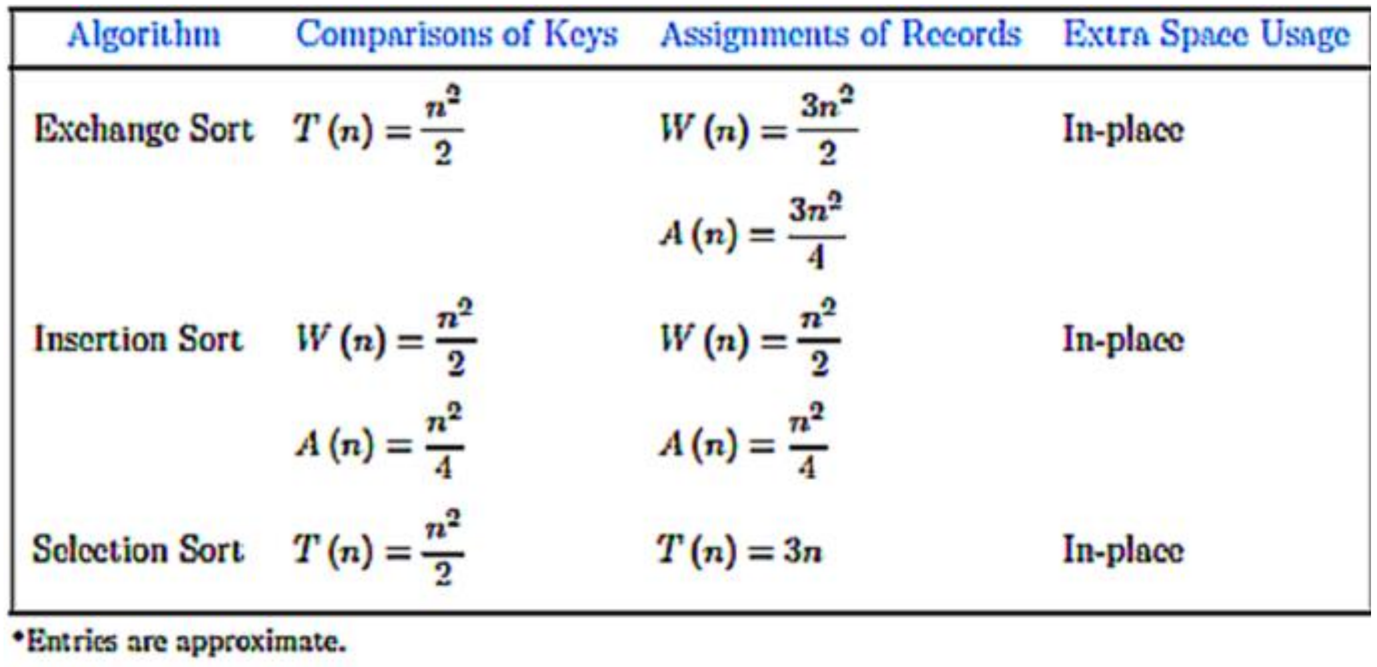
Sorting Problems

- By studying and comparing the algorithms, gain insight how to select algorithm for the same problem and how to improve a different algorithm
- Sorting is one of the few problems for which there are algorithms have time complexities about as good as their lower bound(key comparison algorithms)
- algorithms that sort only by comparison of keys:
  - sort only by comparisons of keys
  - compare two keys to determine which is larger
  - copy keys but no other operations on them
- Analyze the algorithms in terms of the number of comparisons of keys and the number of assignments of records (e.g. exchange takes three assignments)
- Analyze how much extra space the algorithms require besides the space needed to store the input
- When the extra space is a constant, the algorithm is called an in-place sort
- Assume we are always sorting in nondecreasing order

Insertion and Selection Sort

- **Insertion sort:**
  - Sorts by inserting in correct place in sorted array
  - From week 4: complexity: time O(n) best, O(n^2) average, O(n^2) worst; space O(1)
  - Comparison: for each element compares with other until n-1 in worst case and about half of that in average case
  - Assignment: keeps swapping every two elements out of place as inserts so same number as comparisons
- **Selection sort:**
  - Sorts by keeping track of smallest element and swaps
  - From week 4: complexity: time O(n^2) best, O(n^2) average, O(n^2) worst; space O(1)
  - Comparison: for each element compares with other until n-1
  - Assignment: tracks of smallest so only swaps once to put in right place (n-1 times for total sort)

Table 7.1 Analysis Summary



Algorithm	Comparisons of Keys	Assignments of Records	Extra Space Usage
Exchange Sort	$T(n) = \frac{n^2}{2}$	$W(n) = \frac{3n^2}{2}$	In-place
		$A(n) = \frac{3n^2}{4}$	
Insertion Sort	$W(n) = \frac{n^2}{2}$	$W(n) = \frac{n^2}{2}$	In-place
	$A(n) = \frac{n^2}{4}$	$A(n) = \frac{n^2}{4}$	
Selection Sort	$T(n) = \frac{n^2}{2}$	$T(n) = 3n$	In-place

•Entries are approximate.

Mergesort and Quicksort

- Mergesort
  - Breaks list into two halves of about equal size and break each one into two halves until we get list of one number then we return and merge it with the preceding list
  - From week 4: complexity: O(n log n) for time and O(n) for space
  - Three ways to improve mergesort:
    - Dynamic version (Algorithm 7.3) – start with singletons (no divide needed), merge into groups of two, four, etc. until sorted (avoid overhead of stack operations)
    - Linked list version (Algorithm 7.4) – adjusts links instead of moving records (no extra array of records, less time since not assigning large records)
    - More complex merge algorithm (Huang and Langston 1988) with O(n) merge and constant space
- Quicksort
  - Selects pivot element where list is split in two then sorts elements to be less value on left of pivot and greater on right of pivot. Recursively splits each sub-list and sorts
  - From week 4: complexity: O(n log n) for best/average and O(n^2) for worst time and O(log n) for space but worst O(n)
  - Unlike mergesort does not need extra array
  - To improve :
    - Determine larger subarray and stack that one while other is sorted (reduces space because usually smaller stack)
    - Use faster version for partition (p. 333 ex 23) which cuts number of assignments
    - Decrease number of pushes and pops on the stack by only saving values that are needed and not always low, high, and pivot point
    - Determine threshold value to use iterative instead of diving instance further
    - If array already closed to sorted (worst case for this algorithm) not chose first item as pivot

Heapsort

- Type of selection sort where sorts by selecting records in order and placing in their proper sorted position
- From week 4: time complexity O(n log n) and space O(1)
- When implemented as complete binary tree it is really in place (Algorithm 7.5) and heap can be constructed in linear time

Table 7.2 Analysis Summary

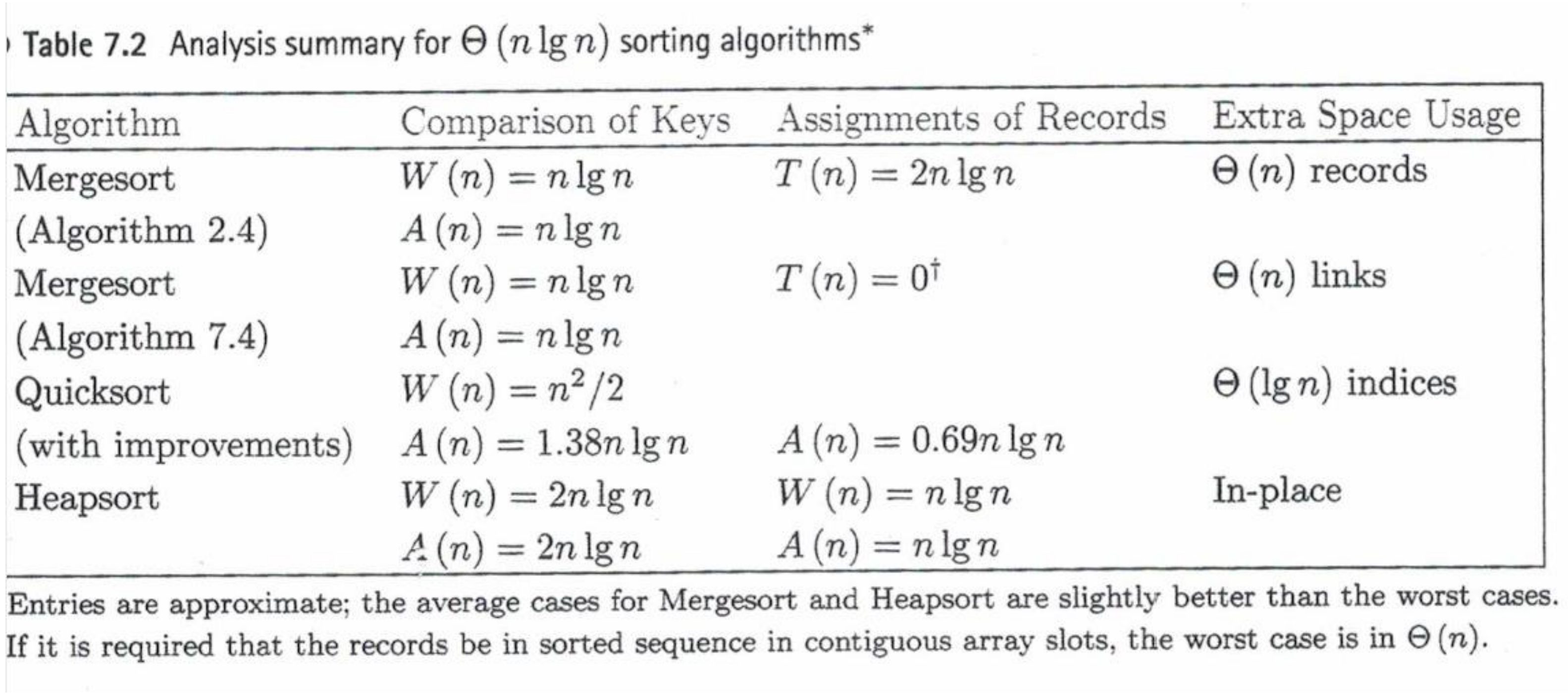


Table 7.2 Analysis summary for $\Theta(n \lg n)$ sorting algorithms*

Algorithm	Comparison of Keys	Assignments of Records	Extra Space Usage
Mergesort (Algorithm 2.4)	$W(n) = n \lg n$ $A(n) = n \lg n$	$T(n) = 2n \lg n$	$\Theta(n)$ records
Mergesort (Algorithm 7.4)	$W(n) = n \lg n$ $A(n) = n \lg n$	$T(n) = 0^{\dagger}$	$\Theta(n)$ links
Quicksort (with improvements)	$W(n) = n^2/2$ $A(n) = 1.38n \lg n$	$A(n) = 0.69n \lg n$	$\Theta(\lg n)$ indices
Heapsort	$W(n) = 2n \lg n$ $A(n) = 2n \lg n$	$W(n) = n \lg n$ $A(n) = n \lg n$	In-place

Entries are approximate; the average cases for Mergesort and Heapsort are slightly better than the worst cases.
If it is required that the records be in sorted sequence in contiguous array slots, the worst case is in $\Theta(n)$.

Radix Sort

- Remember that any algorithm that sorts only by comparisons of keys can be no better than O (n log n)
- If have additional data about keys (e.g. always 3 digits) we can improve
- From week 4: time complexity O(dn) where d is number of digits in key
- Uses linked list
- Distributes keys into piles (distributions) based on digit
- Fig 7.14 p. 327 inspect digits left to right so sorted by first digit, then second digit, etc.
- Fig 7.15 p 328 inspect digits right to left so first sorted by last digit, then next, etc.
- Algorithm 7.6:
  - Distributes into piles based on digit
  - In loop remove keys from pile and merge into master list ordered based on pile
  - Best case time complexity O(n log n) and often than Quicksort
  - Only extra space needed is O(n) links

Questions ?

- Post in the discussions
- Send email to RMcFadden@HarrisburgU.edu
- Respond usually within 48hours