

Sorts and Searches

--	--	--

Sorts and Searches Topics

- ❑ Select Sort and Search
- ❑ qsort() and Binary Search
- ❑ Quick Sort Algorithm

Simple Sort and Search

□ Select Sort

- Loop from $inx = 0 \dots N-1$
 - Select inx as current largest item
 - Loop from $jnx = inx \dots N-1$
 - If $array[jnx]$ smaller than $array[inx]$
 - select jnx as item
 - swap $array[inx]$ and $array[jnx]$, only if item is not inx
- N^2 algorithm
- Characterized by a nested loop

Simple Sort and Search

□ Select Sort

- Advantages
 - Easy to remember
 - Works with various data structures
 - Maximum on N "swaps"
- Pitfalls
 - N^2 algorithm doesn't scale well
- Example
 - <http://faculty.washington.edu/sproedp/advc/csamples/less11-1.c.html>

qsort() and Binary Search

- ▣ void qsort(
 - void * array,
 - size_t count,
 - size_t size,
 - int (*cmp)(const void *, const void *);
- array – the array to be sorted
- count – number of objects in array
- size – size of objects in array
- cmp – user defined function which compares objects

qsort() and Binary Search

- ▣ void qsort()
 - Implements Quick Sort Algorithm
 - ▣ More on this later
 - ▣ N Log N algorithm
 - See <stdlib.h>

qsort() and Binary Search

▣ qsort() comparison function

- The qsort comparison function takes two void pointers. These pointers are 'converted' to the appropriate type
- The function returns a
 - ▣ zero if they're equal
 - ▣ -1 if left < right
 - ▣ 1 if left > right
 - ▣ (H&S sect. 20.5)

qsort() and Binary Search

▣ qsort() comparison function

- This sense (> or <) is relative the ordering of the array. As stated, the ordering of the array is smallest to largest. To achieve largest to smallest, then flip the sense of the return.
- In practice the following is usually supported but is not strictly ANSI:
 - ▣ return a zero if they're equal
 - ▣ less then zero if left < right
 - ▣ greater then zero if left > right.

qsort() and Binary Search

- ▣ bsearch() comparison function return value and sense are similar

▣ Example

- <http://faculty.washington.edu/sproedp/advc/csamples/less11-2.c.html>

Quick Sort Algorithm

▣ Quick Sort Algorithm

- also known as divide and conquer
- please refer to the [Data Structures and Program Design in C](#), Section 7.8 for details
- N Log N Algorithm

▣ Example

- <http://faculty.washington.edu/sproedp/advc/csamples/less11-3.c.html>

Quick Sort Algorithm

▣ Case Study

# Objects	Selection Sort	Quick Sort	N^2	$N \log N$
10	45 Loops 10 Swaps	24 Loops 18 Swaps	10^2	$10 \log 10$
100	4,950 Loops 100 Swaps	716 Loops 413 Swaps	100^2	$100 \log 100$
1,000	499,500 Loops 1000 Swaps	10,359 Loops 5250 Swaps	1000^2	$1000 \log 1000$
10,000	49,995,000 Loops 10,000 Swaps	179,923 Loops 74,499 Swaps	10000^2	$10000 \log 10000$
100,000	4,999,950,000 Loops 100,000 Swaps	2,136,929 Loops 1,039,597 Swaps	100000^2	$100000 \log 100000$