**Midterm review**

All topics from Module 1 through and including Module 4

Review assignments 1, 2 and 3.

In general, be able to write algorithms discussed, determine/explain runtime, prove or justify correctness, explain how algorithms work, give an example of the algorithm.

**Write algorithms that may be a variation of those reviewed in class and**

- Explain the main concept of your algorithm

- Give pseudo-code

- Present an example of running your algorithm

- Prove/justify its correctness and its running time

**Given input, provide the output and/or other information relating to the algorithm.**

- e.g. for graph algorithms, given a representation of a graph, provide info on the DFS tree, BFS tree, Directed DFS or BFS, Top Sort order and other information that may be derived from the algorithm

**Proof techniques**

- Proof by Induction

- Proof by contradition

- Proof by counter example

- Proof by explaning a loop invariant

show a recursion tree for runtime analysis

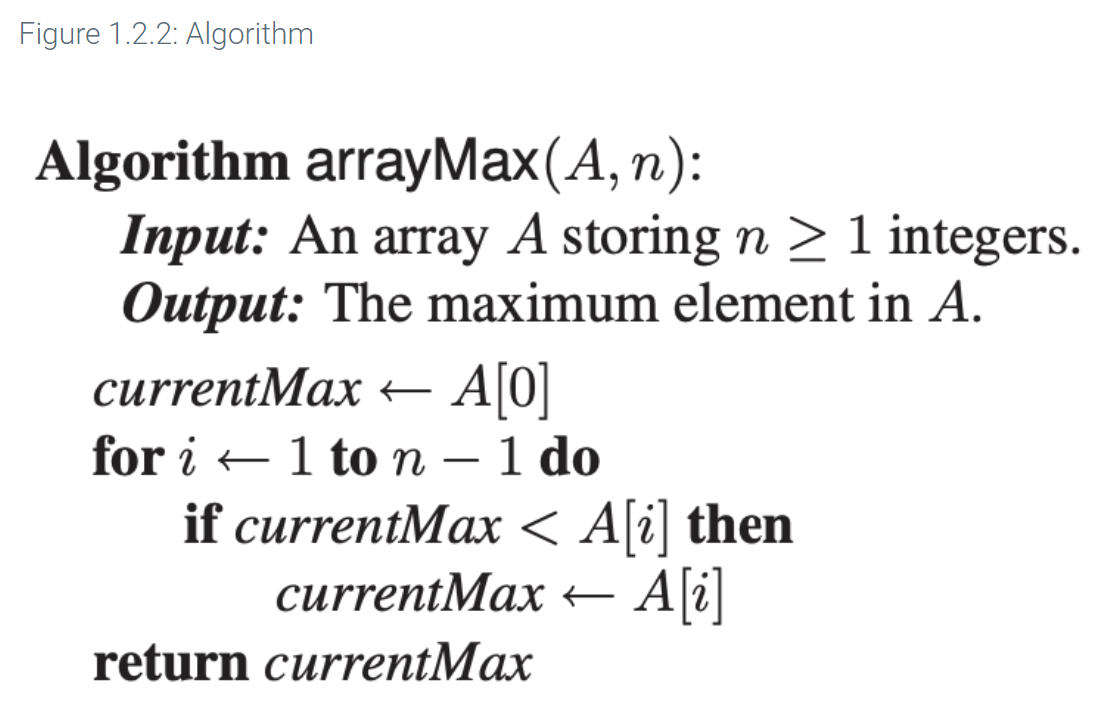
**What does Big-Oh, Big-Theta, Big-Omega, little-oh, little-theta and little-omega mean?**

- Given function function f(n) use limit rule or defintion of Big-Oh, Big-Theta, ... to show that it is in g(n)

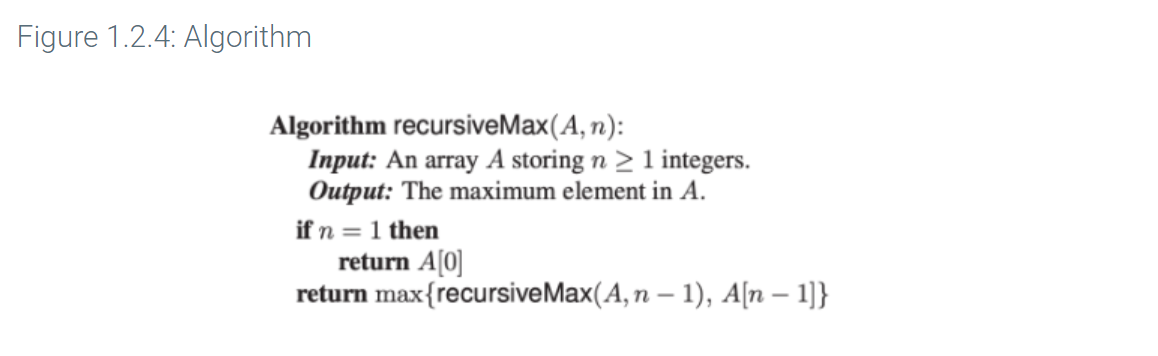
- Given some code, determine the runtime

**Algorithms Discussed**

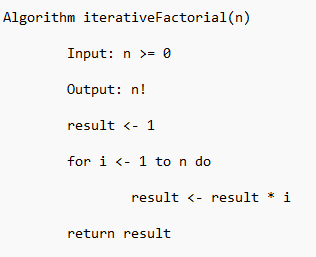
- arrayMax



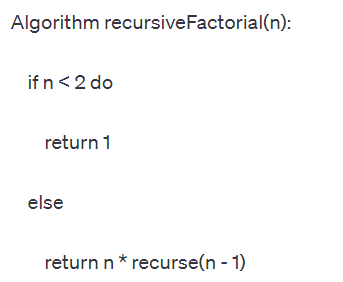
- recursiveArrayMax



- iterativeFactorial



- recursiveFactorial



**Sorting Algorithms**

- Merge Sort

- Quick Sort

- Heap Sort

- Bucket Sort

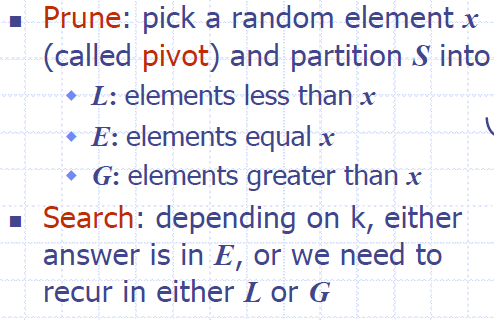
- Radix Sort

**What is the selection problem and how to solve,** **runtime analysis.**

**1. Definition**

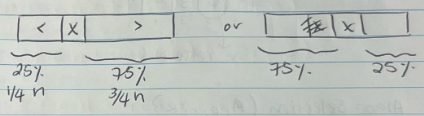
Given an integer k and n elements x1, x2, …, xn, taken from a total order, find the k-th smallest element in this set.

**2. how to solve**



**3. runtime analysis**

Scenario：pick a pivot that leaves a subarray of 75% of elements that we use for the next invocation.



Base case: n = 1:

T(1) = c

T(n) = cn + T(3/4 n)

Claim: Tn = cn + T(3/4 n) <= 4\*cn

Proof by induction:

Base case n = 1:

T1 = c <= 4 c true

Inductive hypothesis: n = k - 1

T(k-1) <= 4(k-1)

Inductive steps: n = k

Tk <= 4\*ck

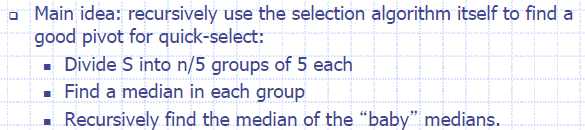
According to the recursion, T(k) = ck + T(3/4 k) <= ck + T(3/4 k)\*4c = 4\*ck

So, T(n) <= T(3/4 n) + cn (expected # of calls before a good call)

**What is Median-of-Medians, when used,** **why important, runtime analysis.**

**Median-of-Medians, when used**

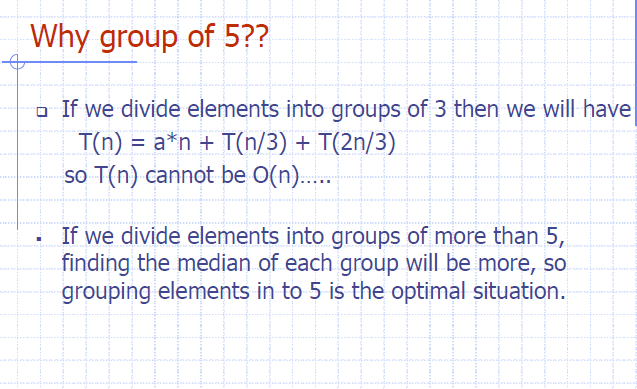
Median-of-Medians is a selection algorithm used to find the kth smallest (or largest) element in an unsorted list or array.



**why important**

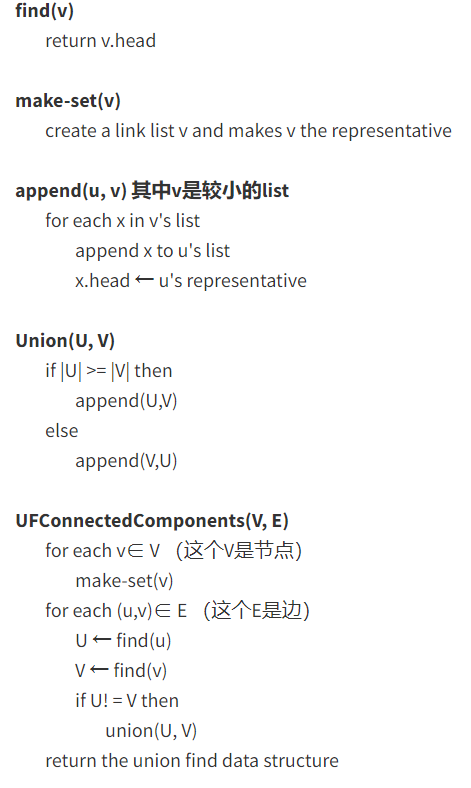
Uses Divide and Conquer strategy. Uses elimination in order to cut down the

running time substantially.



**Union Find Connected Components Algorithm and the Union Find Data Structure made with linked-lists**

**- operations (a.k.a. functions)**



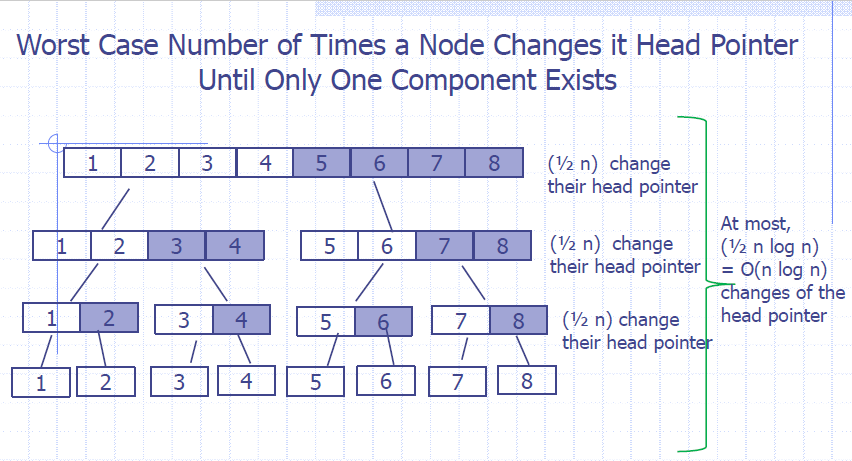
**- runtime analysis for different operations**

find(e) O(1)

makeset(x) O(1)

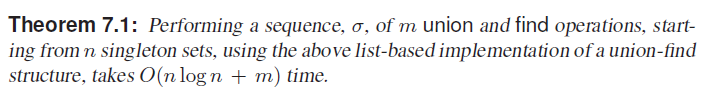
Union(A,B) O(min(|A|, |B|)), in worst-case is O(n), because **in worst-case,** |A| = |B| = n/2,O(n/2) is O(n)

**- runtime for creating the data structure**



UFCC(V,E)





**Graphs**

- terminology for directed, undirected, acyclic graphs

- how to represent internally (adjacency lists via vertex or edges, adjacency matrix)

-- explain/give runtime of operations on graphs that are represented by an adjacency list or adjacency matrix

- Graph Algorithms

-- Depth First Search (DFS)

-- Breadth First Search (BFS)

-- Directed DFS

-- Directed BFS

-- Topological Ordering