Algorithm

An **algorithm** is a step-by-step method of solving a problem.

- Each step should be basic
- the procedure must terminate

> Aside: Algs have much in common with proofs.
> - each step of your proof should be easily seen to follow from axioms (theorems) and earlier statements you have already proved.

How do we come up with algorithms to solve problems?

- There are a handful of Algorithmic Paradigms that are good to know; the majority of algorithms use these approaches in one form or another.

E.g. Problem "Sorting"

Input: An array A of orderable items, like int

Output: array A is now in non-decreasing order

0	1	2	3	4	5	6	7	8	9	10	11 = n-1
6	14	3	20	19	17	5	16	23	1	8	13

Alg Approach #1: "More of the input"

Insertion Sort

- pretend A ends at index $i = 0$.
  Sort it. [Done!]
- If A is actually bigger (let $i = i+1$)
  - Know that $A[0..i]$ is already sorted.
  - Swap element $A[i]$ leftwards until $A[0..i]$ is sorted.
  - if $i = n$, Done!
    else

Alg Approach "More of the Output"

Selection Sort

- "output" A one element at a time. $i = 0$
- What should be the first element $A[0]$?
  - Scan $A[i..n-1]$, find smallest element.
  - Swap $A[i]$ with that smallest element.
  - now $A[0..i]$ is "the" A prefix you will eventually output
    $i = i+1$
    if $i == n$ .. Done!
    else

Analyze Insertion Sort

for $i = 1 .. n-1$ do

  $j = i - 1$

  while $j >= 0$ and $A[j] < A[j+1]$

    swap $A[j], A[j+1]$

$O(1)$ $O(i)$ n times

$i = 1..n-1$

Worst case: $$O\left(\sum_{i=1}^{n-1} i\right) = O(n^2)$$

[Recall: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$]

Analyze Selection Sort

for $loc = 0 ... n-1$ do

  $min\_i = loc$

  for $i = min\_i + 1 ... n-1$ do

    if $A[i] < A[loc]$

      swap $A[i], A[loc]$

Worst case: $$O\left(\sum_{i=n-1}^{1} i\right) = O(n^2)$$

Algorithmic Approach: Divide & Conquer

MergeSort (A, i, j) // A is an array, i, j are indices in the range $A[1 \ldots n]$.

if $j > i$

$mid = \left\lfloor \frac{i+j}{2} \right\rfloor$

MergeSort (A, i, mid)

MergeSort (A, mid+1, j)

Merge (A, i, mid, j)

Merge (A, i, m, j)

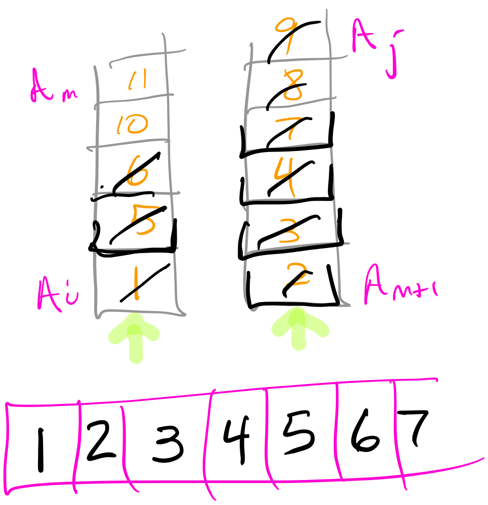
$k = 1;\ x = i;\ y = m+1;$

while $x \le m$ and $y \le j$

if $A[x] \le A[y]$

$B[k{+}{+}] = A[x{+}{+}]$

else $B[k{+}{+}] = A[y{+}{+}]$



if $x \le m$ copy $A[x..m]$ into $B[k..j]$ 89

else if $y \le m$ copy $A[y..j]$ into $B[k..j]$.

Copy $B[1..j-i+1]$ into $A[i..j]$

Running time is proportional to the number of assignment op's; each element is assigned twice so it is $O(n)$

Analyze Merge Sort

1. Running time of Merge is $O(1)$ for every element in the array range(s).

   $O(n)$ for an n-element array.

2. Let time for MergeSort on n elements be $T(n)$

   Then $T(n) = 2 \cdot T(\frac{n}{2}) + n$

MergeSort(n) : mergeSort(n/2)
mergeSort(n/4)
mergeSort(n/4)
O(n/2)

mergeSort(n/2)
MS(n/4)
MS(n/4)
O(n/2).

+ O(n) Merge

The Master Theorem

For a function $T(n)$ defined on positive integers, where $T(n) = aT(\frac{n}{b}) + f(n)$ and $f(n)$ is pos^ve^ valued function, and $a \geq 1$ and $b > 1$, then:

1. $f(n) \in O(n^{\log_b a - \varepsilon})$ for some const $\varepsilon > 0$

$\Rightarrow T(n) \in \Theta(n^{\log_b a})$

2. $f(n) \in \Theta(n^{\log_b a})$

$\Rightarrow T(n) \in \Theta(n^{\log_b a} \lg n)$

3. $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ for some const $\varepsilon > 0$,

regularity $\left\{ \begin{array}{l} \text{AND } \exists \text{ constant } c \text{ s.t. } 0 < c < 1 \text{ and} \\ a f(n/b) < c f(n) \text{ when } n \text{ sufficiently large,} \end{array} \right.$

$\Rightarrow T(n) \in \Theta(f(n))$.

Eg: $T(n) = 2T(\frac{n}{2}) + n^2$

$a =$ $\quad b$ $\quad f(n)$

$n^2 \in O(n^{\lg 2 - \varepsilon})$?

$\in \Theta(n)$?

$\in \Omega(n^{1+\varepsilon})$

Hence $T(n) \in \Theta(n^2)$

$\exists ? \; 0 < c < 1$ where

$2\left(\left(\frac{n}{2}\right)^2\right) < c\, n^2.$

$\frac{n^2}{2} < c\, n^2$

$c = \frac{1}{2}.$

$T(n) = 2T(\frac{n}{2}) + n$

$f(n).$

$n \in$ $\Theta(n^{1-\varepsilon})$

$\Theta(n)$

$\Omega(n^{1+\varepsilon})$

$\therefore T(n) \in \Theta(n \lg n)$, by MT part 2.

$T(n) = 2T(\frac{n}{2}) + \lg n$ $\therefore$

$\lg n \overset{?}{\in}$ $O(n^{1-\varepsilon})$

$\Theta(n)$

$\Omega(n^{1+\varepsilon})$

$\lg n \in O(n^{1-\varepsilon})$, for $\varepsilon = 0.1$

$\therefore$ By MT, case 1, $T(n) \in \Theta(n)$