EC504 Course Organization

- Why Algorithm == Data Strutures?
- Use GitHub (EC405), Slack and CCS
- On GitHub see “class policy” and “outline”
- HW’s pencil and paper handed in class
- Coding with fixed I/O — auto-grading
- In class help with coding style in C/C++
- Basic Unix environment — useful for computer engineers to know!

Course Outline

- Algorithms Analysis CRSL 1-4 (5) HW1
  - Definition of Problem Class of Size N
  - Math for large N Asymptotics:
- I. 1-D Data Structures CRSL 6,7,8,9 HW2
  - Arrays, Lists, Stacks, Queues CRSL 10
  - Searching, Sorting, String Matching, Scheduling
- II. 1.5 D Trees CRLS 12 -`14 HW3
  - BST, AVL,
  - Coding, Union/Join CRLS 18-21, midterm HW4
- III. 2D Graphs CRLS 22,23,24,25, HW 5
  - Traversal, Min Spanning Tree, Shortest Path, Capacity, Min Flow CRLS 26, HW6
- IV Selected Advanced Topics & Projects
  - Spatial Data Structures, FFT's, Complexity, Approx. Solutions, Quantum Computing etc

What is an algorithm? An unambiguous list of steps (program) to transform some input into some output.

- Pick a Problem (set)
- Find method to solve
  1. Correct for all cases (elements of set)
  2. Each step is finite ($\Delta\, t_{step} <$ max time)
  - Next step is unambiguous
  - Terminate in finite number of steps

◆ You know many examples:

GCD, Multiply 2 N bit integers, …

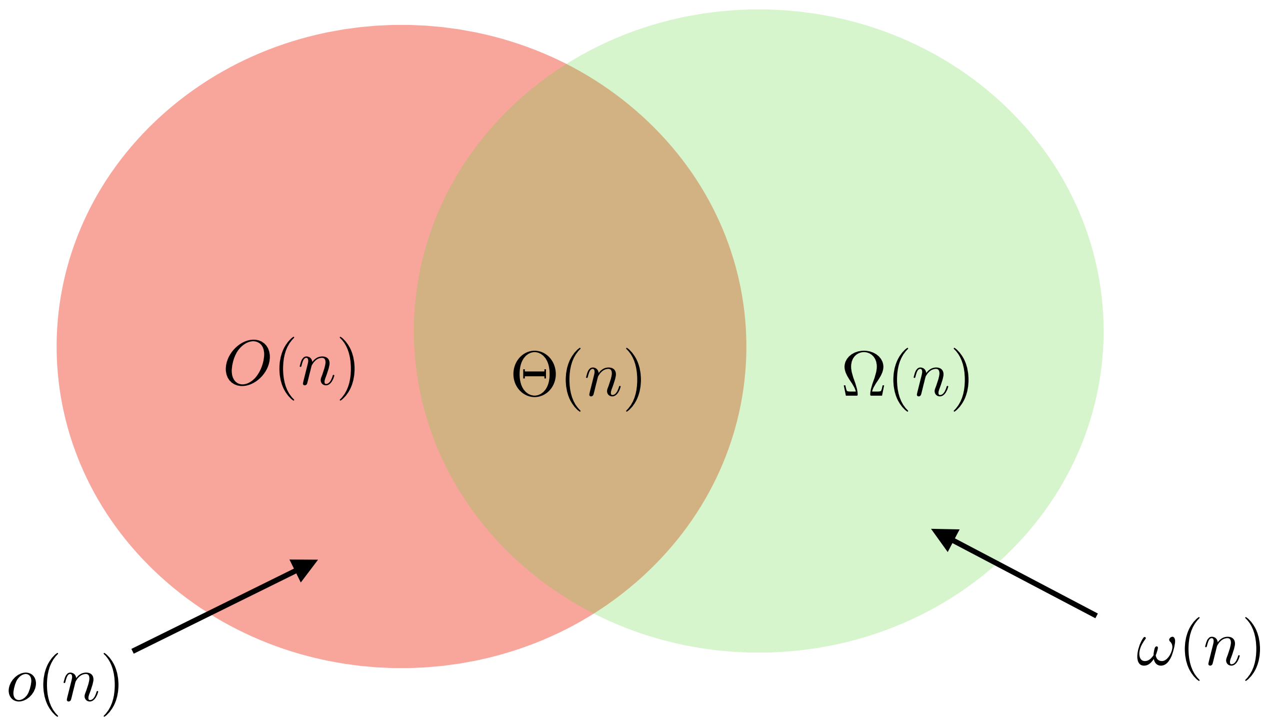
Abu Ja'far Muhammid ibn Musa Al-Khwarizmi
Bagdad (Iraq) 780-850

Al-Kwarimi-Wiki

Growth of Algorithm with Size n

$$T(n) = O(g(n)) \quad \text{or} \quad T(n) \in O(g(n))$$

- T(n) in set O(g(n))
  - like T(n) <= g(n) for large
  - e.g n^a log(n) exp[n] etc.



Rules of thumb

- For polynomials, only the largest term matters.

$$a_0 + a_1 N + a_2 N^2 + \cdots + a_k N^k \in O(N^k)$$

- log N is in o(N)

Proof: As N ➔ 1 the ratio log(N)/N ➔ 0

- Some common functions in increasing order:

1 log N √N N NlogN N^2 N^3 N^{100} 2^N 3^N N! N^N

Insertion Sort --- Deck of Cards

- Insertion Sort(a[0:N-1]):

```
for (i=1;  i < n;  i ++)
      for (j = i;  (j>0) && (a[j]<a[j-1]);  j--)
            swap a[j] and a[j-1] ;
```

Worst case $\Theta(N^2)$ number of "swaps" (i.e. time)

Outer loop trace for Insertion Sort: O(n^2)

- a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] (Swaps)
- 6 | 5 2 8 3 4 7 1 (1)

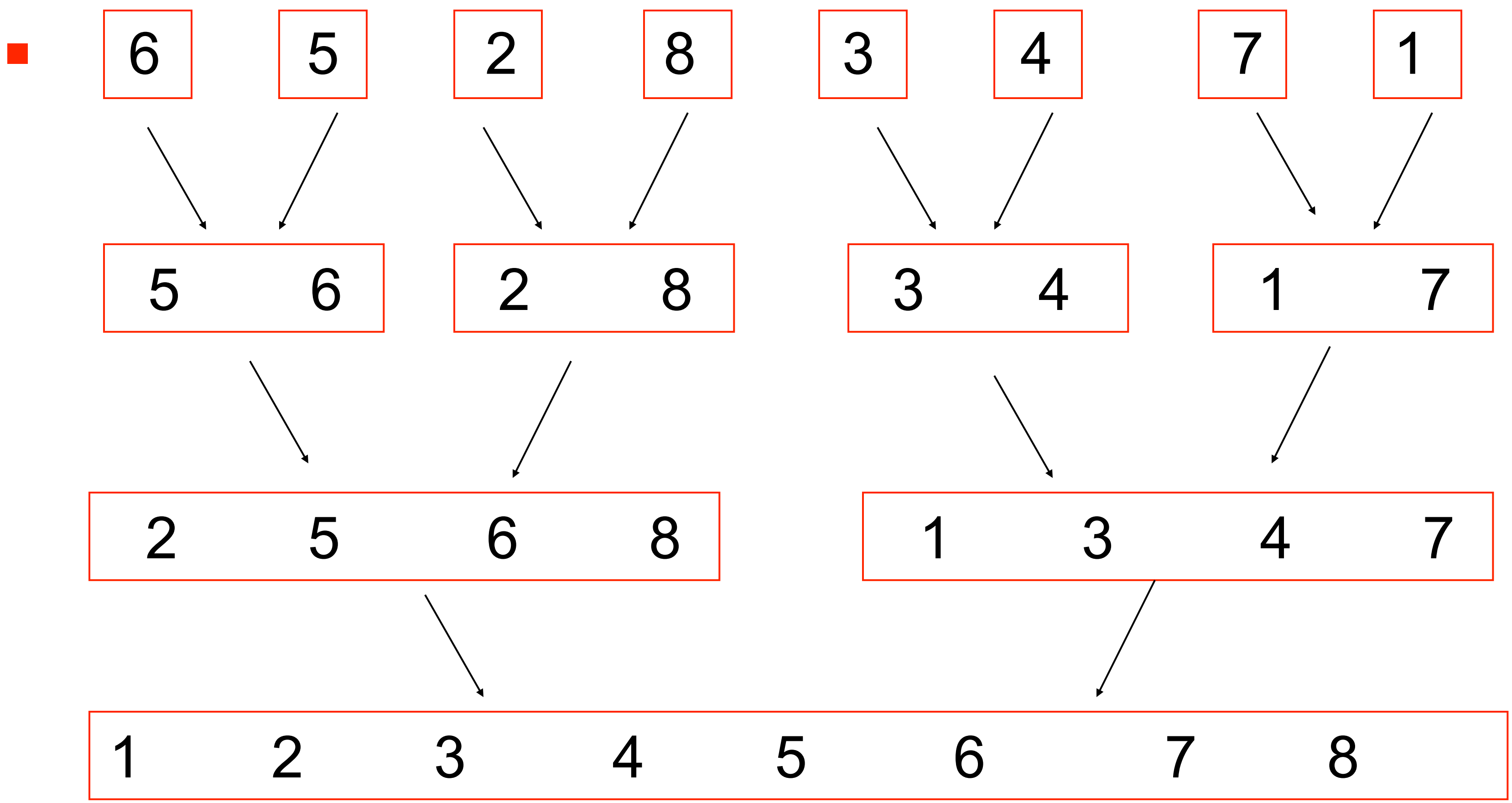
  5 ← → 6
- 5 6 | 2 8 3 4 7 1 (2)

  2 ← → 6

  2 ← → 5
- 2 5 6 | 8 3 4 7 1 (0)
- 2 5 6 8 | 3 4 7 1 (3)
- 2 3 5 6 8 | 4 7 1 (3)
- 2 3 4 5 6 8 | 7 1 (1)
- 2 3 4 5 6 7 8 | 1 (7)

Merge Sort - Recursive O(n log(n))

- a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]



How do we find T(n)? What is big Oh ?

- Count the number of steps:
  - What is a step? RAM serial model.

  - Iterative loops: Sum series like

$$\sum_{i=0}^{N} i^k = 1 + 2^k + 3^k + \cdots + N^k \sim O(N^{k+1})$$

but k= -1 ➔ O(log(n))

  - Solve Recursive Relations:

T(n) = a T(n/b) + O(f(n))

Sums

- Cases: $$\sum_{i=1}^{N} 1 = N \approx \frac{1}{1} N$$

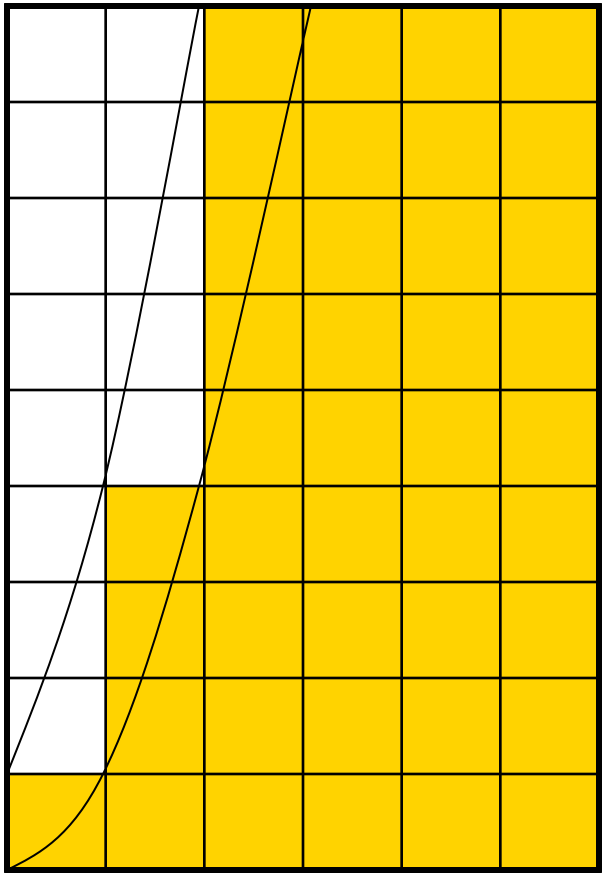
$$\sum_{i=1}^{N} i = \frac{1}{2} N(N+1) \approx \frac{1}{2} N^2$$

$$\sum_{i=1}^{N} i^2 = \frac{1}{6} N(N+1)(2N+1) \approx \frac{1}{3} N^3$$

$$\sum_{i=1}^{N} i^k \approx \frac{1}{k+1} N^{k+1}$$

Prove this by Integration:

Estimating Sums



- Integral Bounds:

$S_k = \sum_{i=1}^{N} i^k$

Estimate by integrating $S_k(x) = x^k$

$$\int_0^N x^k dx \leq S_k = \sum_{i=1}^{N} i^k \leq \int_0^N (x+1)^k dx$$

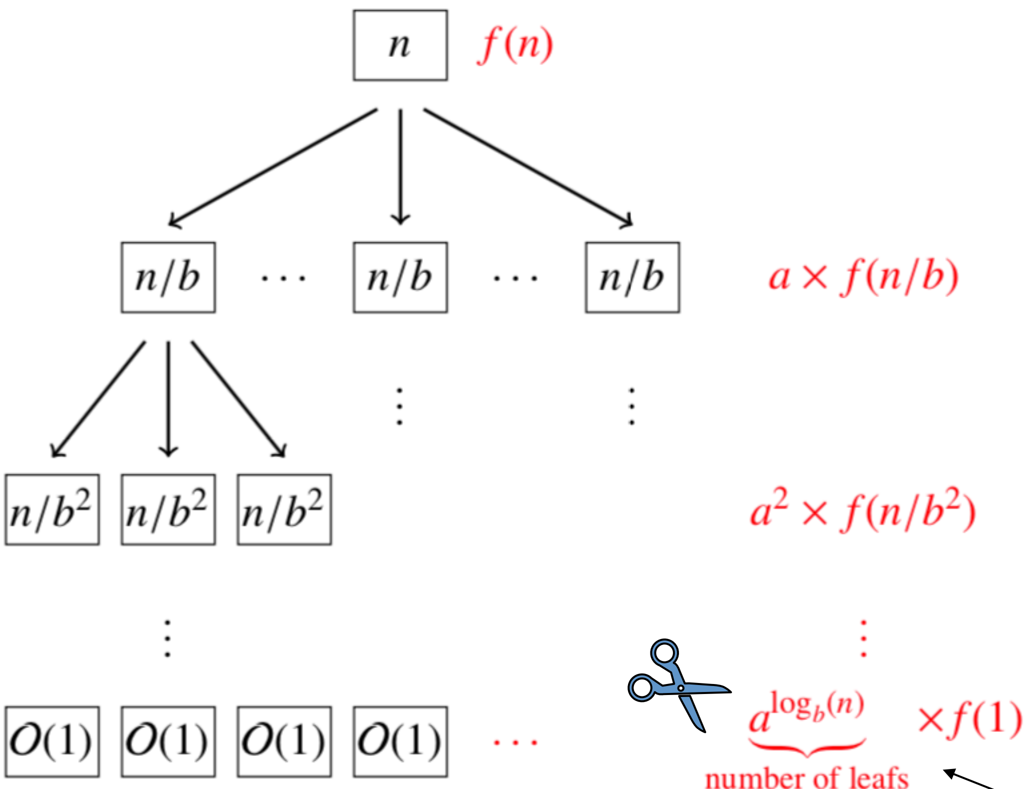
$$\frac{1}{k+1} N^{k+1} \leq S_k \leq \frac{1}{k+1}((N+1)^{k+1} - 1)$$

Build Tree to Solve

$$T(n) = aT(n/b) + f(n)$$

$$\boxed{n/b^h = 1 \implies h = \log_b(n)}$$

Height



Breadth

$$\boxed{T(n) = f(n) + af(n/b) + \cdots + a^{log_b(n)-1} f(b^2) + a^h T(1)}$$

Master Equation (brute force): T(n) = aT(n/b) + f(n)

$$
\begin{aligned}
T(n) &= aT(n/b) + f(n) \\
aT(n/b) &= a^2T(n/b^2) + af(n/b) \\
a^2T(n/b^2) &= a^3T(n/b^3) + a^2f(n/b^2) \\
\cdots & \quad \cdots \\
a^{h-2}T(b^2) &= a^{h-1}T(b) + a^{h-2}f(b^2) \\
a^{h-1}T(b) &= a^hT(1) + a^{h-1}f(b)
\end{aligned}
$$

$$T(n) = a^hT(1) + f(n) + af(n/b) + a^2f(n/b^2) + \cdots + a^{h-1}f(b)$$

$a^h = n^\gamma$

using: $n/b^h = 1 \implies h = \log_b(n)$

Let's be very careful for $f(n) = cn^k$

$$T(n) = aT(n/b) + c\,n^k$$

$$aT(n/b) = a^2T(n/b^2) + c\,an^k/b^k$$

$$a^2T(n/b^2) = a^3T(n/b^3) + c\,a^2n^k/b^{2k}$$

$$\cdots \qquad \cdots$$

$$a^{h-2}T(b^2) = a^{h-1}T(b) + c\,a^{h-2}n^k/b^{(h-2)k}$$

$$a^{h-1}T(b) = a^hT(1) + c\,a^{h-1}n^k/b^{(h-1)k}$$

Therefore

$$T(n) = a^hT(1) + c\,n^k\frac{(a/b^k)^h - 1}{a/b^k - 1}$$

$$a^h = n^\gamma \longrightarrow \quad = n^\gamma T(1) + c\,\frac{n^\gamma - n^k}{a/b^k - 1}$$

$$\boxed{\text{since} \quad 1 + a/b^k + (a/b^k)^2 + (a/b^k)^3 + \cdots + (a/b^k)^{h-1} = \frac{(a/b^k)^h - 1}{a/b^k - 1}}$$

Master Equation:
T(N) = a T(N/b) + Θ (g(N))

Theorem: The asymptotic Solution is:

- $T(N) \in \Theta(N^\gamma) \quad \text{if } g(N) \in O(N^{\gamma-\epsilon}) \ \forall \epsilon > 0$
- $T(N) \in \Theta(g(N)) \quad \text{if } g(N) \in \Omega(N^{\gamma+\epsilon}) \ \forall \epsilon > 0$
- $T(N) \in \Theta(N^\gamma \log(N)) \quad \text{if } g(N) \in \Theta(N^\gamma)$

where $a = b^\gamma$ or $\gamma = \log(a)/\log(b)$