6.006 Introduction to Algorithms
Spring 2008

# 6.006 Recitation

Build 2008.last

# 6.006 Proudly Presents

- Life After 6.006: Options
  - Daydream: Theory
  - Pick Skillz: Competitions
  - Go Pro: Master the Art [of Programming]

- Final Review

# After 6.006: Daydream

- This is the best time to do it
  - Web 2.0 → a lot of data sources to play with: Google, eBay, Facebook, Flickr, ...
- Algorithms in 6.006 can be do cool stuff
  - Web 2.0 → you can build an app that makes a real impact quickly

# After 6.006: Pick Skillz

- Warm up with HS competitions
  - www.usaco.org - USA training site
    - google “IOI” - International Olympiad
- College: the ACM ICPC
  - google “acm problems”
- Top Coder - www.topcoder.com

# After 6.006: Pick Skillz

- Pros
  - (almost) Instant gratification
  - Learn to pwn exams
  - Free trips, prizes, rep (ask recruiters)
- Cons
  - Lower level coding: C, *maybe* Java
  - Luck matters a lot

# After 6.006: Go Pro

- Read: "Hackers and Painters - Big Ideas from the Computer Age" by Paul Graham
- Get in the habit of writing beautiful code
  - Take communication classes: code that is hard to understand can't be beautiful
- Learn from the masters: agile programming, pragmatic programmers

# After 6.006: Go Pro

- Have a weapon at every level: n00bs (Java), low (C / C++), high (Python, Ruby, Erlang)
- General knowledge in all aspects of coding: architecture and OSes, networks, security, parallel processing, databases, web
  - MIT classes covering all of the above
- Learn a new language a year
- CODE

# After 6.006: Go Pro

- Pros
  - Every interviewer will love you
  - Can do contract work to make quick $$
  - Build cool stuff
- Cons
  - Results take more time to show
  - Lots of competition

# After 6.006: Wrap-up

- The options above are not disjoint

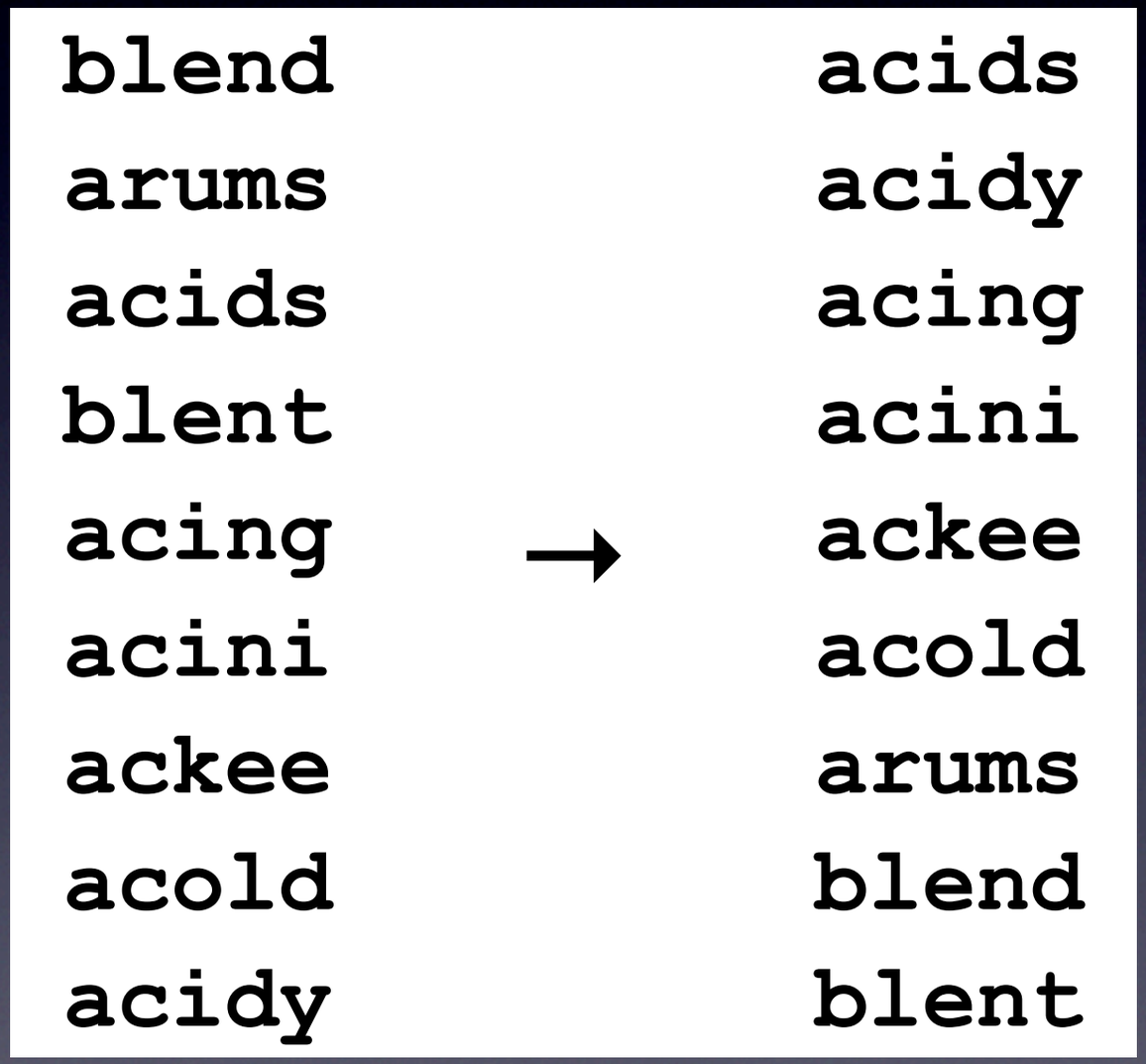
**Thank you!**

# Warm-up: Sort Strings

- N strings, O(1) alphabet size, want to sort them

- Easy: each string has M characters, sort in O(MN)

- Hard: string i has $C_i$ characters, sort in $O(\Sigma C_i)$



# Warm-up: Solutions

- Easy
  - Radix sort, strings are M-digit numbers
- Hard
  - let M = max(C1, C2 ... Cn)
  - use radix sort w/M rounds, 0...M-1
  - add string i at round M - Ci, its smaller than all existing strings

# P1: String Suffixes

- Given a string s of N characters, O(1) alphabet size
- The string's suffixes are $suff_i$=s[1...i]
- Want an array so that a[j] = i means that $suff_i$ is the $j^{th}$ in the sorted order

aardvark

| Suffixes | | Sorted | |
|---|---|---|---|
| 1 | aardvark | 1 | aardvark |
| 2 | ardvark | 2 | ardvark |
| 3 | rdvark | 6 | ark |
| 4 | dvark | 4 | dvark |
| 5 | vark | 8 | k |
| 6 | ark | 3 | rdvark |
| 7 | rk | 7 | rk |
| 8 | k | 5 | vark |

a = [1, 2, 6, 4, 8, 3, 7, 5]

# P1: Solution

- Radix sort, log(N) rounds 0...log(N)-1
- Round k sorts $a[i...i+2^k]$ (suffixes truncated to up to $2^k$ characters)
  - Round 0: simple sorting letters = digits
  - Round i: use the results of round i-1
- Notice $a[i...i+2^k] = a[i...i+2^{k-1}] + a[i+2^{k-1}+1...i+2^k]$
- So can use ranks computed in round i to represent $a[i...i+2^k]$ as 2 base-N digits
- O(N) per round, for a total running time of O(Nlog(N))

# P2: Longest Palindrome

- Given a string of N characters, find the longest palindrome substring
- Substring: s[i...j] (continuous)
- Palindrome: if you read it backwards it's the same

fun**abccba**fun

No **straw warts** here

G**ATTA**CA

3**1**4**1**592**653**8**979**0

want **atoyota**

# P2: Solution

# P3: Feed the Drones

- drones produce widgets when given food
- 3 types of food: (Fish, Meat, Bread)
- drones like variety: remember the last 3 crates they were fed and produce widgets according to variety

| # of types in last 3 items | Widgets |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

Sample production given food

| **F** | B | M | B | B | B | M | F | F | F |
|---|---|---|---|---|---|---|---|---|---|
| **W** | 1 | 2 | 2 | 2 | 1 | 2 | 3 | 2 | 1 |

# P3: Feed the Drones

- Given: 2 work sites , a sequence of N crates of food (of specific types)
- Have to assign each crate to one of the two sites, want to maximize production
- Cannot throw away or reorder the crates

Sample input and answer

| I | B | M | F | F | M | B | F | F |
|---|---|---|---|---|---|---|---|---|
| **A** | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |

Production achieved

| B | F | M | F | M | F | B | F |
|---|---|---|---|---|---|---|---|
| +1 | +2 | +3 | +2 | +1 | +2 | +3 | +2 |

Widgets at both sites: 16

# P3: Solution

- Dynamic Programming
- State
  - the current crate
  - the types of the last 2 crates delivered at each of the 2 work sites
  - adding N as the 4th type, means Nothing
- DP[i][($u_1$,$u_2$)][($v_1$, $v_2$)] = max. production for the first i crates, so that the last 2 crates at site 1 were of types $u_1$, $u_2$, and the last 2 crates at site 2 were of types $v_1$, $v_2$
- Recursion: exercise
- Running time: O(N)

# P4: Light up the House

- House of rooms, and paths between rooms; unique path between any two rooms

- Light switch in room R toggles the light in R and its neighbors

- Start with all lights off, end with all lights on, min. number of switches

# P4: Solutions

- Structural DP (missed it?)
- Strategy: solve subtrees before parents
  - State: light on or off; used switch at node
  - DP[node][l][s] = min. number of switches to light up everything under “node”; node is light up if l = true, and the switch at “node” is used is on if s = true

# P5: Partial Sums v2

- Start out with array of numbers a[1...N]
- Want to answer M operations, an op can be:
  - Update: a[i] = v
  - Query: Σa[i...j]