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 $M = \max(C1, C2 \ldots Cn)$
  - 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 $\text{suff}_i = s[1...i]

- Want an array so that $a[j] = i$ means that $\text{suff}_i$ is the $j^{th}$ in the sorted order

<table>
<thead>
<tr>
<th>Suffixes</th>
<th>Sorted</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark</td>
<td>aardvark</td>
</tr>
<tr>
<td>ardvark</td>
<td>ardvark</td>
</tr>
<tr>
<td>rdvark</td>
<td>ark</td>
</tr>
<tr>
<td>dvark</td>
<td>dvark</td>
</tr>
<tr>
<td>vark</td>
<td>k</td>
</tr>
<tr>
<td>ark</td>
<td>rdvark</td>
</tr>
<tr>
<td>rk</td>
<td>rk</td>
</tr>
<tr>
<td>k</td>
<td>vark</td>
</tr>
</tbody>
</table>

$a = [1, 2, 6, 4, 8, 3, 7, 5]$
PI: 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 \( O(N\log(N)) \)
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
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.

<table>
<thead>
<tr>
<th># of types in last 3 items</th>
<th>Widgets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>M</th>
<th>F</th>
<th>F</th>
<th>M</th>
<th>B</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Production achieved**

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>F</th>
<th>M</th>
<th>F</th>
<th>M</th>
<th>F</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+1</td>
<td>+2</td>
<td>+3</td>
<td>+2</td>
<td>+1</td>
<td>+2</td>
<td>+3</td>
<td>+2</td>
</tr>
</tbody>
</table>

**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 4\textsuperscript{th} 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
  - \( \text{DP}[\text{node}][l][s] = \min \text{ number of switches to light up everything under “node”; node is light up if } l = \text{true}, \text{ and the switch at “node” is used is on if } s = \text{true} \)
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**: $\sum a[i...j]$