6.006 Recitation 10

Agenda
- Lect review
  - lower bound
  - counting sort.

Overhead
- Quiz
  - Friday: review
  - Wednesday: optional
  - problems soon
  - exercises in text
- PSet 3 out: Work in grps!

Lower Bounds on Sorting
- If this hurts your brain: don't worry about it.

Decision Tree
- Each node is a comparison → 2 possibilities → binary tree.
- Each leaf is a possible termination of the algorithm.
- A traversal from root to leaf is ONE execution of the alg.

example: insertion sort 3 elts. (construct for students)

permuation of input array.

permutations:

note: all permutations are ∈ leaf set of decision tree.

Math

\( n! = \# \text{permutations of input} \)
\( h = \text{height of decision tree} = \text{runtime of algorithm} \)
\( \# \text{nodes in tree of height} h \leq 2^h \)
\( \min \# \text{nodes in a correct comparison sort alg} = n! \)
\( n! \leq 2^h \)
\( h \geq \lg(n!) = n \lg(n) \)
so \( n \lg(n) \) is a lower bound on the runtime of a comparison based sort.
Counting Sort

Idea: don't compare them!

Assumption: Integers only.

- Use the values as indices into an auxiliary array.

Input: \[ x \in \text{array then } x \in \mathbb{Z}_k \]

Aux: \[ \text{len} = n \]

\[ \text{Counts # of occurrences of each int.} \]

\[ \text{Location in end array to put each elt.} \]

(= cumulative sum of counts array)

Output: \[ \text{len} = n \]

Example:

(Step-by-step)

\[
\begin{array}{cccc}
2 & 0 & 1 & 1 \\
0 & 6 & 1 & 1 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}
\]

\[
\begin{array}{cccc}
2 & 3 & 2 & 0 & 1 \\
2 & 5 & 7 & 0 & 8
\end{array}
\]

Oops too high! Sub 1.

- Really when going from (input, aux) \to output
  - We traverse the input array in reverse.
  - Why? Stable = if two values are "equal" and \( x \) comes before \( y \) in input, then \( x \) before \( y \) in output

Runtime:

1. Go through input \( A \)
   a. For each elt - incr aux[\( A[i] \)]
2. Compute cumulative sum of aux
3. Go through input \( A \)
   a. For each elt - find loc in output using aux
   b. Copy elt into output
   c. Decr aux[\( A[i] \)]

\[ \Theta(2nk) \]

Fast! (probably)

What if \( k \) is large? \to \text{Radix}!
Quick Probability Review

- probability
  \[ P(\text{dice}=4) = \frac{1}{6} \]

- expected value
  \[ E[\text{dice roll}] = \sum_{i=1}^{6} \frac{i}{6} = \frac{1}{6} (1+2+3+4+5+6) = 3.5 \]

  \[ E[2 \text{ rolls}] = \sum_{i=1}^{12} i P(\text{rolls}=i) \leq 2\text{-hard.} \]

- linearity!
  \[ E[\text{roll}_{1} + \text{roll}_{2}] = E[\text{roll}_{1}] + E[\text{roll}_{2}] = 7 \]

  in general... \[ E[\Sigma c_{i} X_{i}] = \Sigma E[c_{i} X_{i}] \]

- indicator random variables
  \[ I_{A} = \begin{cases} 1 & \text{A occurs} \\ 0 & \text{else} \end{cases} \]

  \[ E[I_{A}] = \Pr(A) \quad \text{(proof: } E[I_{A}] = \text{value of } \Pr(A) = 1 \text{ } \Pr(A) + 0 \text{ } \Pr(\neg A) = \Pr(A) \text{)} \]

ex: expected # of 6s in k rolls.

\[ I = \sum_{i=1}^{k} I_{i} \quad \text{where } I_{i} = \text{got a 6 on the } i^{\text{th}} \text{ roll.} \]

\[ E[I] = E[\sum_{i=1}^{k} I_{i}] = \sum_{i=1}^{k} E[I_{i}] = k \frac{1}{6} \]

Coding

Good Practices
- A write comments first.
- B use small, simple pieces.
- C tests first then code.
- D step away from your code sometimes; fresh eyes are good.

Debugging
- goal: isolate the problem
- B & C help with this A LOT
- use asserts / prints

1. help you
2. help others help you.
3. you will forget what it does no matter how obvious.
4. easier to do
5. easier to understand
6. easier to test.
7. understand corner cases before you run into them.
8. can run as soon as code written to see if it works.
Testing
- **TRY** to break the code
  - use corner cases (0, 1, -1 ... ∞)
  - give invalid inputs ← if robustness matters.
  - sample the problem space well