Randomized Selection

Tufts University

Warm-up Question

What are the main steps of SELECTION algorithm?

Previously ...

- Randomized analysis is a fair way of expressing runtimes

  Worst possible input

  Average over random choices
- When analyzing a randomized algorithm:
  1. Define $X(n)$ = runtime for worst case input of size n

     X_i = IRV for each possible random case
  2. Compute $E[X_i]$ and runtime c_i associated to that case
  3. Express X as combination of X_is

     Often $X = \sum_i X_i c_i +$ common operations
  4. Compute $E[X]$ and use cheat of the week

Cheat of the day

$$\sum_{i=\lfloor \frac{n}{2} \rfloor}^{n-1} i \leq \frac{3}{8} n^2$$

Selection review

Selection(A,n,k)

if $n \le 5$, $k = 1$ or $k = n$ solve by brute force

i $\leftarrow$ MoM and recurse to pick an index(A, n)

pos $\leftarrow$ Partition(A, n, i)

if pos $= k$ return $A[pos]$

if pos $> k$ return Selection$(A[1, \text{pos} - 1], \text{pos} - 1, k)$

else return Selection$(A[\text{pos} + 1, n], n - \text{pos}, k - \text{pos})$

Can we simplify the algorithm?

Hint

Remember step 3?

Lemma

Regardless of how we pick index, the algorithm is always correct

Let's pick an index at random

Randomized Selection

RandSelection(A,n,k)

  if $n \leq 5$, $k = 1$ or $k = n$ solve by brute force

  i ← Random Number between 1 and n

  pos ← $\textsc{Partition}(A, n, i)$

  if pos $= k$ return $A[pos]$

  if pos $> k$ return $\textsc{RandSelection}(A[1, \mathrm{pos} - 1], \mathrm{pos} - 1, k)$

  else return $\textsc{RandSelection}(A[\mathrm{pos} + 1, n], n - \mathrm{pos}, k - \mathrm{pos})$

Runtime of RandSelection?

RandSelection

If base case solve by brute force

`i` $\leftarrow$ Random Number between 1 and n

`pos` $\leftarrow$ Partition(A, n, i)

recursively solve in correct portion

Let's analyze:

X_j =1 if $A[i]$ is j-th smallest number in array

If $X_j = 1$, runtime $c_j = \max\{R(j-1), R(n-j)\}$

Overall runtime $R(n) = \sum_j X_j c_j + \Theta(n)$

Math magic

$$
\begin{aligned}
E[R(n)] &= \\
&= E[\Theta(n) + \textstyle\sum_{j=1}^{n} X_j \max\{R(j-1), R(n-j)\}] && \text{definition} \\
&= \Theta(n) + \textstyle\sum_{j=1}^{n} E[X_j \max\{R(j-1), R(n-j)\}] && \text{LoE} \\
&= \Theta(n) + \textstyle\sum_{j=1}^{n} E[X_j] E[\max\{R(j-1), R(n-j)\}] && \text{independent} \\
&= \Theta(n) + \tfrac{1}{n} \textstyle\sum_{j=1}^{n} E[\max\{R(j-1), R(n-j)\}] && \text{algebra} \\
&= \Theta(n) + \tfrac{2}{n} \textstyle\sum_{j=n/2}^{n-1} E[R(j)] && \text{double counting}
\end{aligned}
$$

Sounds familiar?

Math magic

$E[R(n)] \le \Theta(n) + \frac{2}{n}\sum_{j=n/2}^{n-1} E[R(j)]$

Claim: $E[R(n)] \le cn$

Proof by substitution

Base case: $E[R(1)] = d \le c \cdot 1 \rightarrow$ ok as long as $c \ge d$

Induction:

$$\begin{aligned}
E[R(n)] &= \Theta(n) + \frac{2}{n}\sum_{j=n/2}^{n-1} E[R(j)] \\
&\le d'n + \frac{2}{n}\sum_{j=n/2}^{n-1} cj && \text{(induction hypothesis)} \\
&\le d'n + \frac{2c}{n}\sum_{j=n/2}^{n-1} j && \text{(algebra)} \\
&\le d'n + \frac{2c}{n}\left(\frac{3}{8}n^2\right) && \text{(cheat of the day)} \\
&\le d'n + \frac{3c}{4}n && \text{(algebra)} \\
&\le cn && (\text{if } c \ge 4d')
\end{aligned}$$

Discussion

- Randomized analysis is a powerful tool
  - Does not improve runtime (just help analysis)
- Theoretically speaking, worst case is better than expected
  - In practice expected $\approx$ worst-case
  - Randomized algorithms are often easier to code

Additional Practice questions:

- Compare the formulas between QS and Rand Select
  - What is different?
  - One solves to $\Theta(n \log n)$ and other to $\Theta(n)$
  - Can you explain why?
- What other randomized algorithms do you know?
  - Can you analyze them?

Block 1 topics

- $\star$ Asymptotic notation
- $\star$ *Comparison-based* sorting algorithms
- *Other* sorting algorithms
- Other sorting properties (in-place, stable, ...)
- $\star$ Recurrences (identifying and expressing runtime)
- $\star$ Solving recurrences (trees/substitution/master theorem)
- $\star$ SELECTION algorithm
- $\star$ Sorting lower bound
- $\star$ IRVs and Randomized algorithms
- Heaps

Items with $\star$ are of critical importance

Impressive!

Asymptotic notation

- ⋆ Understanding the concepts of big O, Ω, and Θ

  Juggle definitions to prove statements

- ⋆ Given an algorithm, compute (ideally tight) bounds

  Justify why bounds apply

- ⋆ Bounds **always** apply to worst-case runtime

  Capable of comparing bounds

Sorting algorithms

No need to know C++ code, just big picture overview

- ⋆ Given a setting, choose fastest algorithm

Comparison-Based

InsertionSort, MergeSort, BubbleSort, Heapsort, and QuickSort

- ⋆ Analyze new algorithms (i.e. NotSoEfficientSort)

Other algorithms

RadixSort, CountingSort

- ⋆ Given strange setting, give values of k, ℓ and d

Math tools

Recurrences

- Express runtime of recursive algorithm as recurrence
- Solve a recurrence using substitution and recursion tree

  Identify if master theorem applies (and apply if possible)

IRVs

- Model a complex problem as a combination of simple IRVs

  Use linearity of expectation to juggle math

SELECTION algorithm

- Know pseudocode of algorithm
- ⋆ Understand relationship between all pieces
- Argue correctness
- ⋆ Understand recurrence for runtime
- ☺ Advocate for addition into MoMa

Sorting lower bound

- ⋆ Understand it applies to **unknown** strategies

  Recognizing total number of outcomes

- ⋆ Identifying number of branches

  Connect height of tree to runtime

- ⋆ Glueing all pieces together

Randomized algorithms

- ⋆ Given algorithm, identify simple event
- ⋆ Express overall runtime as combination of IRVs
- ⋆ Understand all components of final expression

  Being able to juggle math

  Expected = worst case input, average over random choices

About the exam

Will try to cover all topics

- Tight! Beware of time constraints!
- Spend time proportionate to points
- Light on memorization, focus on understanding

Remember golden rules

- Read how to write proofs
- Justify your answers
- Show that you know
- Make it easy for graders to give you full credit

Relax! It is only one exam

- How many courses at Tufts?
- How many exams before?
- Many more await after!

Exam Tips

Keep a mental map of tools

- Read question and analyze
- Identify: do I need sorting? IRVs?

Prioritize time by points

- Read how to write proofs
- Practice math for fluidity
- Topics will (most likely) not be repeated
- If you get stuck go to next question
- Look at it afterwards

Show that you know

- Say what you will do
- Justify your steps
- Good handwriting

Coming Soon

- Revisit Hashing, BST, AVL

  Can you prove that Hashing takes $O(1)$ time?
- Augmented trees

  Let's make AVL trees even more powerful
- Dynamic Programming

  Recursion on steroids
- Amortized runtime

  Another way to average runtimes

This and much more in block 2 of the course!