Advanced Algorithms

Spring 2021

Lecture 1

Agenda for Today

- Welcome back to a new semester !!
  - We are hopefully seeing the last embers of the pandemic.
  - Nevertheless, we have to prepare for a possibly new normal in many ways.
  - We will continue to teach and learn in the online mode for Spring 2021 too.
  - Stay safe, but curious!

Agenda for Today

- Welcome back to a new semester !!
  - We are hopefully seeing the last embers of the pandemic.
  - Nevertheless, we have to prepare for a possibly new normal in many ways.
  - We will continue to teach and learn in the online mode for Spring 2021 too.
  - Stay safe, but curious!

- My details
  - Kishore Kothapalli, Professor, IIIT Hyderabad
  - Email: kkishore@iiit.ac.in (the best way to reach me)
  - Research interests span parallel computing and distributed algorithms.

Agenda for Today

- Rest of Today's Class
  - Syllabus
  - Policies
  - Expectations
  - Actual lecture

Syllabus

- Roughly, a three module course
  - Module 1: Randomized Algorithms
  - Module 2: Parallel and Distributed Algorithms
  - Module 3: Advanced Topics
    - Big data and Sampling
    - Algorithm engineering
    - Any other

Syllabus

- Roughly, a three part course
  - Randomized Algorithms
    - Chernoff bounds and Randomized Routing
    - Perfect Hashing
    - Graph algorithms: MIS, Spanners
    - Randomized Rounding
    - Approximate counting
  - Parallel and Distributed Algorithms

Syllabus

- Roughly, a three part course
  - Randomized Algorithms
  - Parallel and Distributed Algorithms
    - Flynn’s Taxonomy and Models of Computations
    - Basic parallel algorithms: Search/Sort/Scan/Merge
    - Tree and Graph Algorithms
    - Lower Bounds

Policies

- Grading (Tentative)
  - Homeworks: 30% (We will have some lateness policy here)
  - In-class Quizzes : 25% (We will have some redundancy here)
  - End Exam: 15%
  - Quizzes 1 and 2 : 30%
  - Exceptional Performance: 5% extra

- Any submission that is graded and evaluated should not be copied from any source.
- Copied submissions will get zero for the first instance and negative for repeat offences.

Support Staff

We have two Teaching Assistants named so far:

Sayantan Jana, sayantan.jana@research.iiit.ac.in

Athreya Chandramouli, athreya.chandramouli@research.iiit.ac.in

Policies

- Textbooks: Do not own these books just for the class!
  - Randomized Algorithms, Motwani and Raghavan
  - Introduction to Parallel Algorithms, J. JaJa
  - Other material to be posted on the course website
- Most welcome to write to me if you have any questions.

Expectations

- Utilize class time effectively.
  - Starts with all of you settling by the class time.
  - Ask any question you may have. No question is small to ask.
  - Do not show up late.

Agenda for Today

- On to the actual lecture….randomization in computing
- We will see how randomization can help in designing and analyzing algorithms.
- Starting with a very simple example...

A Simple Example

- Let us recall the partition procedure and quick sort.
- We assume that the elements of the set are all distinct.

Algorithm RandQuickSort(S)

Choose a pivot element x_i u.a.r from S

Split the set S into two subsets $S_1 = \{x_j | x_j < x_i\}$ and $S_2 = \{x_j | x_j > x_i\}$ by comparing each x_j with x_i

Recurse on sets S1 and S2

Output the sorted set S_1, x_i, and then sorted S_2.

end Algorithm

A Simple Example

Algorithm RandQuickSort(S)

Choose a pivot element x_i u.a.r from S

Split the set S into two subsets $S_1 = \{x_j | x_j < x_i\}$ and $S_2 = \{x_j | x_j > x_i\}$ by comparing each x_j with x_i

Recurse on sets S1 and S2

Output the sorted set S_1, x_i and then sorted S_2.

end Algorithm

- Let T(n) be the time taken by the procedure.
- What can we say about T(n)?

A Simple Example

Algorithm RandQuickSort(S)

Choose a pivot element x_i u.a.r from S

Split the set S into two subsets $S_1 = \{x_j | x_j < x_i\}$ and $S_2 = \{x_j | x_j > x_i\}$

Recurse on sets S1 and S2

Output the sorted set S_1, x_i and then sorted S_2.

end Algorithm

- What can we say about T(n)?
- The maximum value of T(n) occurs when the pivot element x_i is the largest/smallest element of the remaining set during each recursive call of the algorithm.
- In this case, $T(n) = n + (n - 1) + \cdot \cdot \cdot + 1 = O(n^2)$.
- This value of T(n) is reached with a very low probability of $1/n \cdot 1/ n-1 \cdot \cdot \cdot \cdot 1/2.\ 1 = 1/n!$.
- Also, the best case occurs when every pivot element splits the applicable set into two equal sized subsets and then $T(n) = O(n \ln n)$.

A Simple Example

Algorithm RandQuickSort(S)

Choose a pivot element x_i u.a.r from S

Split the set S into two subsets $S_1 = \{x_j | x_j < x_i\}$ and $S_2 = \{x_j | x_j > x_i\}$

Recurse on sets S1 and S2

Output the sorted set S_1, x_i and then sorted S_2.

end Algorithm

- This implies that T(n) has a distribution between O(n ln n) and $O(n^2)$.
- Now we derive the expected value of T(n).

A Simple Example

- This implies that T(n) has a distribution between O(n ln n) and $O(n^2)$.
- Now we derive the expected value of T(n).
- Note that if the i^{th} smallest element is chosen as the pivot element then S_1 and S_2 will be of sizes $i - 1$ and $n - i - 1$ respectively.
- And this choice has a probability of 1/n.
- Hence, the recurrence relation for T(n) is:
- $T(n) = n + T(X) + T(n - 1 - X)$
- In the above, X is a random variable indicating the size of S_1.

A Simple Example

- Note that if the ith smallest element is chosen as the pivot element then S1 and S2 will be of sizes i − 1 and n − i − 1 respectively.
- Hence, the recurrence relation for T(n) is:
- $T(n) = n + T(X) + T(n - 1 - X)$
- In the above, X is a random variable indicating the size of S_1.
- Further, note that Pr[X = i] = 1/n = Pr[n – 1 – X = i] as Pr[X = i] = 1/n.
- The last part is true since the choice of the pivot is uniform.

A Simple Example

- Hence, the recurrence relation for $T(n)$ is:
- $T(n) = n + T(X) + T(n - 1 - X)$
- Further, note that $\Pr[X = i] = 1/n = \Pr[n - 1 - X = i]$ as $\Pr[X = i] = 1/n$.
- Taking expectations on both sides,
- $E[T(n)] = n + 1/n \ \Sigma_{i=1}^{n-1} E[T(i)] + (1/n) \ \Sigma_{i=1}^{n-1} E[T(i)]$.

- Use the fact that for a random variable Y with its support partitioned into sets $A_1, A_2, \ldots, A_n$, we have that $E[Y] = \Sigma_i \Pr(A_i) \cdot E[Y \mid A_i]$.

- Let $f(i) = E[T(i)]$.

A Simple Example

- Taking expectations on both sides of,
- $E[T(n)] = n + 1/ n \ \Sigma_{i=1}^{n-1} E[T(i)] + (1/n) \ \Sigma_{i=1}^{n-1} E[T(i)]$.

- Let $f(i) = E[T(i)]$.
- We can simplify the expression as $f(n) = n + (2/n) \ \Sigma_i f(i)$.
- Further simplification results in $nf(n) = n^2 + 2(f(1) + f(2) + \ldots + f(n-1))$.

A Simple Example

- Further simplification results in

- $nf(n) = n^2 + 2(f(1) + f(2) + ... + f(n – 1)).$

- Write the above by replacing n with n – 1 to get

  $(n – 1) f(n-1) = (n-1)^2 + 2(f(1) + f(2) + ... + f(n – 2)).$

- Subtract the two equation to get:

  $nf(n) = n^2 + 2(f(1) + f(2) + ... + f(n – 1)).$

A Simple Example

- Further simplification results in
- $nf(n) = n^2 + 2(f(1) + f(2) + \ldots + f(n-1))$.

- Write the above by replacing n with $n-1$ to get
- $(n-1) f(n-1) = (n-1)^2 + 2(f(1) + f(2) + \ldots + f(n-2))$.

- Subtract the two equation to get:

  $nf(n) = n^2 + 2(f(1) + f(2) + \ldots + f(n-1))$.

  or $f(n) = (n+1)/n \, f(n-1) + (2n-1)/n$.
- We prove by induction that $f(n) \le 2n \ln n$.

A Simple Example

- f(n) = (n+1)/n f(n – 1) + (2n – 1)/n.
- We prove by induction that f(n) ≤ 2n ln n.+l
- Check the base case for n = 1.
- Let the result hold for all values of n up to n – 1.
- Induction step:

Let the claim hold for all values up to $n-1$. Then,

$$
\begin{aligned}
f(n) &= \tfrac{n+1}{n} f(n-1) + \tfrac{2n-1}{n} \\
&\leq \tfrac{n+1}{n} 2(n-1)\ln(n-1) + \tfrac{2n-1}{n} \text{ by induction hypothesis} \\
&= \tfrac{2(n^2-1)}{n} \ln(n-1) + \tfrac{2n-1}{n} \\
&= \tfrac{2(n^2-1)}{n} (\ln n + \ln(1 - \tfrac{1}{n})) + \tfrac{2n-1}{n}
\end{aligned}
$$

We make use of the standard inequality stated below.

A Simple Example

Let the claim hold for all values up to $n-1$. Then,

$$\begin{aligned} f(n) &= \tfrac{n+1}{n} f(n-1) + \tfrac{2n-1}{n} \\ &\leq \tfrac{n+1}{n} 2(n-1)\ln(n-1) + \tfrac{2n-1}{n} \text{ by induction hypothesis} \\ &= \tfrac{2(n^2-1)}{n} \ln(n-1) + \tfrac{2n-1}{n} \\ &= \tfrac{2(n^2-1)}{n} (\ln n + \ln(1-\tfrac{1}{n})) + \tfrac{2n-1}{n} \end{aligned}$$

We make use of the standard inequality stated below.

$$1 + x \leq e^x \; \mathit{for} \; x \in R.$$

Hence,

$$\begin{aligned} f(n) &\leq \tfrac{2(n^2-1)}{n} (\ln n - \tfrac{1}{n}) + \tfrac{2n-1}{n} \\ &= 2n\ln n - \tfrac{2}{n}\ln n - 2 + \tfrac{2}{n^2} + 2 - \tfrac{1}{n} \\ &\leq 2n\ln n, \text{ establishing the inductive step.} \end{aligned}$$

A Simple Example

- Hence, the expected running time of the randomized quick sort algorithm is O(n ln n).
- But one of the limitations of the recurrence relation approach is that we do not how the running time of the algorithm is spread around its expected value.
- Can this analysis be extended to answer questions such as, with what probability does the algorithm RandQuickSort needs more than 12n lnn time steps?
- Later on, we apply a different technique and establish that this probability is very small.
- To be able to answer such queries, we study Tail inequalities in the following.