Learning Goals

- State the implementation of the Quicksort algorithm
- Define Las Vegas and Monte Carlo algorithms
- Basic analysis of the running time of randomized algorithms
- Develop intuitive understanding of the balls and bins asymptotics

Setup and the algorithm

- Input: A set S of n integers $a_1, \ldots, a_n$.
- Output: Sorted array of the n integers in increasing order.

Setup and the algorithm

- Input: A set S of n integers $a_1, \ldots, a_n$.
- Output: Sorted array of the n integers in increasing order.
- Recall: Deterministic algorithms: Merge Sort (divide and conquor, running time $O(n \log n)$.

Setup and the algorithm

- Input: A set S of n integers $a_1, \ldots, a_n$.
- Output: Sorted array of the n integers in increasing order.
- Recall: Deterministic algorithms: Merge Sort (divide and conquor, running time $O(n \log n)$.
- Recall lower bound: any deterministic algorithm must make $\Omega(n \log n)$ comparisons in the worst case.

Setup and the algorithm

- Input: A set S of n integers $a_1, \ldots, a_n$.
- Output: Sorted array of the n integers in increasing order.
- Recall: Deterministic algorithms: Merge Sort (divide and conquor, running time $O(n \log n)$.
- Recall lower bound: any deterministic algorithm must make $\Omega(n \log n)$ comparisons in the worst case.
- One of the best known sorting algorithm — Quicksort(S):
  - Base case: If $|S| \leq 3$, return sorted S.
  - Otherwise, pick an element a uniformly at random from S, form two sets: $S^+ := \{b : b > a\}$, $S^- := \{b : b < a\}$. Return Quicksort(S^-), a_j, Quicksort(S^+).

Setup and the algorithm

- Input: A set S of n integers $a_1, \ldots, a_n$.
- Output: Sorted array of the n integers in increasing order.
- Recall: Deterministic algorithms: Merge Sort (divide and conquor, running time $O(n \log n)$.
- Recall lower bound: any deterministic algorithm must make $\Omega(n \log n)$ comparisons in the worst case.
- One of the best known sorting algorithm — Quicksort(S):
    - Base case: If $|S| \leq 3$, return sorted S.
    - Otherwise, pick an element a uniformly at random from S, form two sets: $S^+ := \{b : b > a\}$, $S^- := \{b : b < a\}$. Return Quicksort(S^-), a_j, Quicksort(S^+).
- The randomly chosen a used to split S is called a *pivot*.

Categorization of Randomized Algorithms

- A *randomized algorithm* is simply an algorithm with access to random coins.

Categorization of Randomized Algorithms

- A *randomized algorithm* is simply an algorithm with access to random coins.
- Two categories of randomized algorithms:

Categorization of Randomized Algorithms

- A *randomized algorithm* is simply an algorithm with access to random coins.
- Two categories of randomized algorithms:
  - A *Las Vegas algorithm* always terminates with a correct solution; its running time is a random variable.

Categorization of Randomized Algorithms

- A *randomized algorithm* is simply an algorithm with access to random coins.
- Two categories of randomized algorithms:
    - A *Las Vegas algorithm* always terminates with a correct solution; its running time is a random variable.
    - A *Monte Carlo algorithm* returns a correct solution only probabilistically; its running time may or may not be a random variable.

Categorization of Randomized Algorithms

- A *randomized algorithm* is simply an algorithm with access to random coins.
- Two categories of randomized algorithms:
  - A *Las Vegas algorithm* always terminates with a correct solution; its running time is a random variable.
  - A *Monte Carlo algorithm* returns a correct solution only probabilistically; its running time may or may not be a random variable.
- Quicksort is a Las Vegas algorithm.

Analysis of Quicksort

Theorem

With high probability, the running time of Quicksort is $O(n \log n)$.

Analysis of Quicksort

Theorem

With high probability, the running time of Quicksort is $O(n \log n)$.

- Observation: In each recursion, forming S^+ and S^- altogether takes $O(n)$ time.

Analysis of Quicksort

Theorem

With high probability, the running time of Quicksort is $O(n \log n)$.

- Observation: In each recursion, forming S^+ and S^- altogether takes $O(n)$ time.
- Intuition: if a_j always cuts S in the middle, then the running time is $T(n) \approx 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$.

Analysis of Quicksort

- The execution of Quicksort can be represented by a binary tree:
  - Each node is represented by the set of elements being sorted in the current recursion.

Analysis of Quicksort

- The execution of Quicksort can be represented by a binary tree:
  - Each node is represented by the set of elements being sorted in the current recursion.
  - For the node sorting S, its two children sort S^- and S^+, respectively.

Analysis of Quicksort

- The execution of Quicksort can be represented by a binary tree:
  - Each node is represented by the set of elements being sorted in the current recursion.
  - For the node sorting S, its two children sort S^- and S^+, respectively.
- The total running time for the nodes in one *level* of the three is $O(n)$.

Analysis of Quicksort

- The execution of Quicksort can be represented by a binary tree:
    - Each node is represented by the set of elements being sorted in the current recursion.
    - For the node sorting S, its two children sort S^- and S^+, respectively.
- The total running time for the nodes in one *level* of the three is $O(n)$.
- It suffices to show the height of the tree is $O(\log n)$ w.h.p..

Analysis of Quicksort

- The execution of Quicksort can be represented by a binary tree:
  - Each node is represented by the set of elements being sorted in the current recursion.
  - For the node sorting S, its two children sort S^- and S^+, respectively.
- The total running time for the nodes in one *level* of the three is $O(n)$.
- It suffices to show the height of the tree is $O(\log n)$ w.h.p..
- There are $O(n)$ leaves. We show that the depth of each leaf is $O(\log n)$ w.h.p., and then apply union bound.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.
- We say a step in the path is "good" if the sets of the two children both have sizes $\leq \frac{2}{3}$ that of the parent.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.
- We say a step in the path is "good" if the sets of the two children both have sizes $\leq \frac{2}{3}$ that of the parent.
- There can be at most $\log_{3/2} n$ good steps before we reach the leaf.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.
- We say a step in the path is "good" if the sets of the two children both have sizes $\leq \frac{2}{3}$ that of the parent.
- There can be at most $\log_{3/2} n$ good steps before we reach the leaf.
- Let's bound the probability that, in $27 \ln n$ steps, there are fewer than $\log n$ good steps.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.
- We say a step in the path is "good" if the sets of the two children both have sizes $\leq \frac{2}{3}$ that of the parent.
- There can be at most $\log_{3/2} n$ good steps before we reach the leaf.
- Let's bound the probability that, in $27 \ln n$ steps, there are fewer than $\log n$ good steps.
- Let X_i be the indicator variable for the i-th step being good, then $\mathbf{E}[X_i] = \frac{1}{3}$, and the X_i's are i.i.d.

Analysis of Quicksort

- Consider a fixed leaf of the tree, and the path from the root down to it.
- We say a step in the path is "good" if the sets of the two children both have sizes $\leq \frac{2}{3}$ that of the parent.
- There can be at most $\log_{3/2} n$ good steps before we reach the leaf.
- Let's bound the probability that, in $27 \ln n$ steps, there are fewer than $\log n$ good steps.
- Let X_i be the indicator variable for the i-th step being good, then $\mathbf{E}[X_i] = \frac{1}{3}$, and the X_i's are i.i.d.
- Let X be $\sum_{i=1}^{27 \ln n} X_i$. By Chernoff bound, we have

$$\mathbf{Pr}\left[X < \log_{\frac{3}{2}} n\right] \leq \mathbf{Pr}\left[X < \frac{1}{3}\mathbf{E}[X]\right] \leq \exp\left(-\frac{1}{2} \cdot (\frac{2}{3})^2 \cdot 9 \ln n\right)$$
$$= n^{-2}.$$

Analysis of Quicksort (Cont.)

- Recall that there are n leaves.

Analysis of Quicksort (Cont.)

- Recall that there are n leaves.
- By the union bound, the probability that *any* leaf has depth more than $27 \ln n$ is no more than $n \cdot n^{-2} = n^{-1}$.

Analysis of Quicksort (Cont.)

- Recall that there are n leaves.
- By the union bound, the probability that *any* leaf has depth more than $27 \ln n$ is no more than $n \cdot n^{-2} = n^{-1}$.
- Therefore, with high probability, the height of the tree is bounded by $27 \ln n$.

Analysis of Quicksort (Cont.)

- Recall that there are n leaves.
- By the union bound, the probability that *any* leaf has depth more than $27 \ln n$ is no more than $n \cdot n^{-2} = n^{-1}$.
- Therefore, with high probability, the height of the tree is bounded by $27 \ln n$.
- Obviously the constants in the analysis were not fine-tuned.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.
- In our first lecture, we considered n tasks sending requests uniformly at random to one of the servers.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.
- In our first lecture, we considered n tasks sending requests uniformly at random to one of the servers.
- Such scenarios arise often.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.
- In our first lecture, we considered n tasks sending requests uniformly at random to one of the servers.
- Such scenarios arise often.
- This is often abstracted as a *balls and bins* model: we have n balls and m bins, and each ball is thrown uniformly at random to a bin.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.
- In our first lecture, we considered n tasks sending requests uniformly at random to one of the servers.
- Such scenarios arise often.
- This is often abstracted as a *balls and bins* model: we have n balls and m bins, and each ball is thrown uniformly at random to a bin.
- Any bin receives in expectation $\frac{n}{m}$ balls. If $m = n$, this is 1.

Bins and Balls

- When discussing hashing, we considered a naïve/ideal hash: mapping elements of U uniformly at random to an address.
- In our first lecture, we considered n tasks sending requests uniformly at random to one of the servers.
- Such scenarios arise often.
- This is often abstracted as a *balls and bins* model: we have n balls and m bins, and each ball is thrown uniformly at random to a bin.
- Any bin receives in expectation $\frac{n}{m}$ balls. If $m = n$, this is 1.
- How about the bin that received the most balls? How many balls should we expect to see there?

Balls and Bins when $m = n$

- Let's consider a particular bin. Let X_i be the indicator variable for the event that the i-th ball falls in this bin.

Balls and Bins when $m = n$

- Let's consider a particular bin. Let X_i be the indicator variable for the event that the i-th ball falls in this bin.
- Then $\mathbf{Pr}[X_i = 1] = \frac{1}{n}$.

Balls and Bins when $m = n$

- Let's consider a particular bin. Let X_i be the indicator variable for the event that the i-th ball falls in this bin.
- Then $\mathbf{Pr}[X_i = 1] = \frac{1}{n}$.
- Let X be $\sum_i X_i$. Note that $\mathbf{E}[X] = 1$.

Balls and Bins when $m = n$

- Let's consider a particular bin. Let X_i be the indicator variable for the event that the i-th ball falls in this bin.
- Then $\mathbf{Pr}[X_i = 1] = \frac{1}{n}$.
- Let X be $\sum_i X_i$. Note that $\mathbf{E}[X] = 1$.
- For $t > 0$, we use Chernoff bound

$$\mathbf{Pr}\left[X > (1+t)\,\mathbf{E}\left[X\right]\right] \leq \left(\frac{e^t}{(1+t)^{1+t}}\right)^{\mathbf{E}[X]} \leq \left(\frac{e}{1+t}\right)^{1+t}.$$

Balls and Bins when $m = n$

- Let's consider a particular bin. Let X_i be the indicator variable for the event that the i-th ball falls in this bin.
- Then $\mathbf{Pr}[X_i = 1] = \frac{1}{n}$.
- Let X be $\sum_i X_i$. Note that $\mathbf{E}[X] = 1$.
- For $t > 0$, we use Chernoff bound

$$\mathbf{Pr}\left[X > (1+t)\,\mathbf{E}\left[X\right]\right] \leq \left(\frac{e^t}{(1+t)^{1+t}}\right)^{\mathbf{E}[X]} \leq \left(\frac{e}{1+t}\right)^{1+t}.$$

- We would like to find t so that this probability is smaller than n^{-2}. Essentially we are asking what solves $x^x = n$.

Balls and Bins when $m = n$ (Cont.)

- To estimate the solution of $x^x = n$, we first take logarithm, $x \log x = \log n$, $\log x + \log\log x = \log\log n$.

Balls and Bins when $m = n$ (Cont.)

- To estimate the solution of $x^x = n$, we first take logarithm, $x \log x = \log n$, $\log x + \log \log x = \log \log n$.
- Note that $x < \log n$.

Balls and Bins when $m = n$ (Cont.)

- To estimate the solution of $x^x = n$, we first take logarithm, $x \log x = \log n$, $\log x + \log\log x = \log\log n$.
- Note that $x < \log n$.
- We have $2\log x \geq \log x + \log\log x = \log\log n \geq \log x$, so

$$\frac{1}{2}x \leq \frac{\log n}{\log\log n} \leq x \Rightarrow x = \Theta\left(\frac{\log n}{\log\log n}\right).$$

Balls and Bins when $m = n$ (Cont.)

- To estimate the solution of $x^x = n$, we first take logarithm, $x \log x = \log n$, $\log x + \log\log x = \log\log n$.
- Note that $x < \log n$.
- We have $2\log x \geq \log x + \log\log x = \log\log n \geq \log x$, so

$$\frac{1}{2}x \leq \frac{\log n}{\log\log n} \leq x \Rightarrow x = \Theta\left(\frac{\log n}{\log\log n}\right).$$

- Let the solution to $x^x = n$ be $\gamma(n)$, and let $1 + t = e\gamma(n)$, we have

$$\left(\frac{e}{1+t}\right)^{1+t} = \left(\frac{1}{\gamma(n)}\right)^{e\gamma(n)} = n^{-e} < n^{-2}.$$

Balls and Bins when $m = n$ (Cont.)

- By union bound, with probability at least $1 - \frac{1}{n}$, no bin receives more than $e\gamma(n) = \Theta(\frac{\log n}{\log \log n})$ balls.

Balls and Bins when $m = n$ (Cont.)

- By union bound, with probability at least $1 - \frac{1}{n}$, no bin receives more than $e\gamma(n) = \Theta(\frac{\log n}{\log \log n})$ balls.

Remark

Using Poisson approximation, one can show that w.h.p. there is a bin with $\Omega(\log n / \log \log n)$ balls!

Balls and Bins When $n \gg m$

- As n grows, the number of balls concentrates more sharply around its means.

Balls and Bins When $n \gg m$

- As n grows, the number of balls concentrates more sharply around its means.
- E.g., take $n = 16m \log m$, with the previous notation, $\mathbf{E}[X] = 16 \log m$.

Balls and Bins When $n \gg m$

- As n grows, the number of balls concentrates more sharply around its means.
- E.g., take $n = 16m \log m$, with the previous notation, $\mathbf{E}[X] = 16 \log m$.

$$\mathbf{Pr}\left[X \geq 32 \log m\right] = \mathbf{Pr}\left[X \geq 2\,\mathbf{E}\left[X\right]\right] \leq e^{-\mathbf{E}[X]/3} = m^{-16/3} < \frac{1}{m^2};$$

$$\mathbf{Pr}\left[X \leq 8 \log m\right] = \mathbf{Pr}\left[X \leq \frac{1}{2}\,\mathbf{E}\left[X\right]\right] \leq e^{-\mathbf{E}[X]/8} = \frac{1}{m^2}.$$

Balls and Bins When $n \gg m$

- As n grows, the number of balls concentrates more sharply around its means.
- E.g., take $n = 16m\log m$, with the previous notation, $\mathbf{E}[X] = 16\log m$.

$$\mathbf{Pr}\left[X \geq 32\log m\right] = \mathbf{Pr}\left[X \geq 2\,\mathbf{E}\left[X\right]\right] \leq e^{-\mathbf{E}[X]/3} = m^{-16/3} < \frac{1}{m^2};$$
$$\mathbf{Pr}\left[X \leq 8\log m\right] = \mathbf{Pr}\left[X \leq \frac{1}{2}\mathbf{E}\left[X\right]\right] \leq e^{-\mathbf{E}[X]/8} = \frac{1}{m^2}.$$

Theorem

For $n = \Omega(m\log m)$, with high probability, the number of balls every bin receives is between half and twice the average.