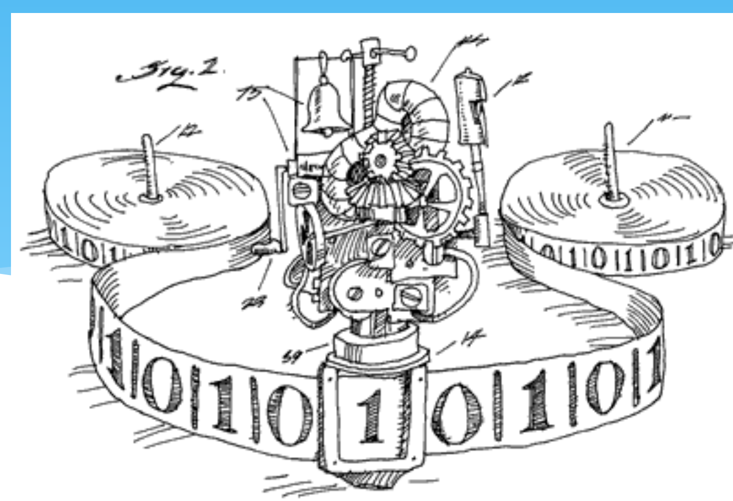
EECS 376: Foundations of Computer Science

Chris Peikert
29 March 2023



Agenda

- More randomness in computation: fast algos/DSs
- Randomized QuickSort
- Skip Lists
- Verifying Matrix Multiplication (if time)

QuickSort

- Recursively sorts an array of numbers
- **Q:** What's the ***worst-case*** runtime of **QuickSort**?
  (e.g., if we always choose the first element as pivot)

QuickSort$(A[1 \ldots n])$:
1. $p \leftarrow$ pick a "pivot" element from A
2. $(L, R) \leftarrow$ **Partition**(A, p) // compare elems to p
3. **QuickSort**(L) and **QuickSort**(R)

1	6	9	4	3	8	7	2	5

1	6	9	4	3	8	7	2	5

From EECS 281 on quicksort:

Time Analysis

- Cost of partitioning N elements: $O(N)$
- Worst case: pivot always leaves one side empty
  - $T(N) = N + T(N - 1) + T(0)$
  - $T(N) = N + T(N - 1) + c$ [since $T(0)$ is $O(1)$]
  - $T(N) \sim N^2/2 \Rightarrow O(N^2)$ [via substitution]
- Best case: pivot divides elements equally
  - $T(N) = N + T(N / 2) + T(N / 2)$
  - $T(N) = N + 2T(N / 2) = N + 2(N / 2) + 4(N / 4) + \ldots + O(1)$
  - $T(N) \sim N \log N \Rightarrow O(N \log N)$ [master theorem or substitution]
- Average case: $\boxed{\text{tricky}}$ ← We have the background for this now!
  - Between $2N \log N$ and $\sim 1.39\, N \log N \Rightarrow O(N \log N)$

Randomized QuickSort

* What if we choose a *random* pivot?
* Let X be the number of *comparisons* made in **QuickSort** on $A[1 \ldots n]$ (note: X is a *random variable*).

Theorem: $\mathbb{E}[X] = O(n \log n)$
(*expected* runtime of randomized **QuickSort**)

RQuickSort$(A[1 \ldots n])$:
1. $p \leftarrow$ a *random* "pivot" from A
2. $(L, R) \leftarrow$ **Partition**(A, p) // compare elems to p
3. **RQuickSort**(L) and **RQuickSort**(R)

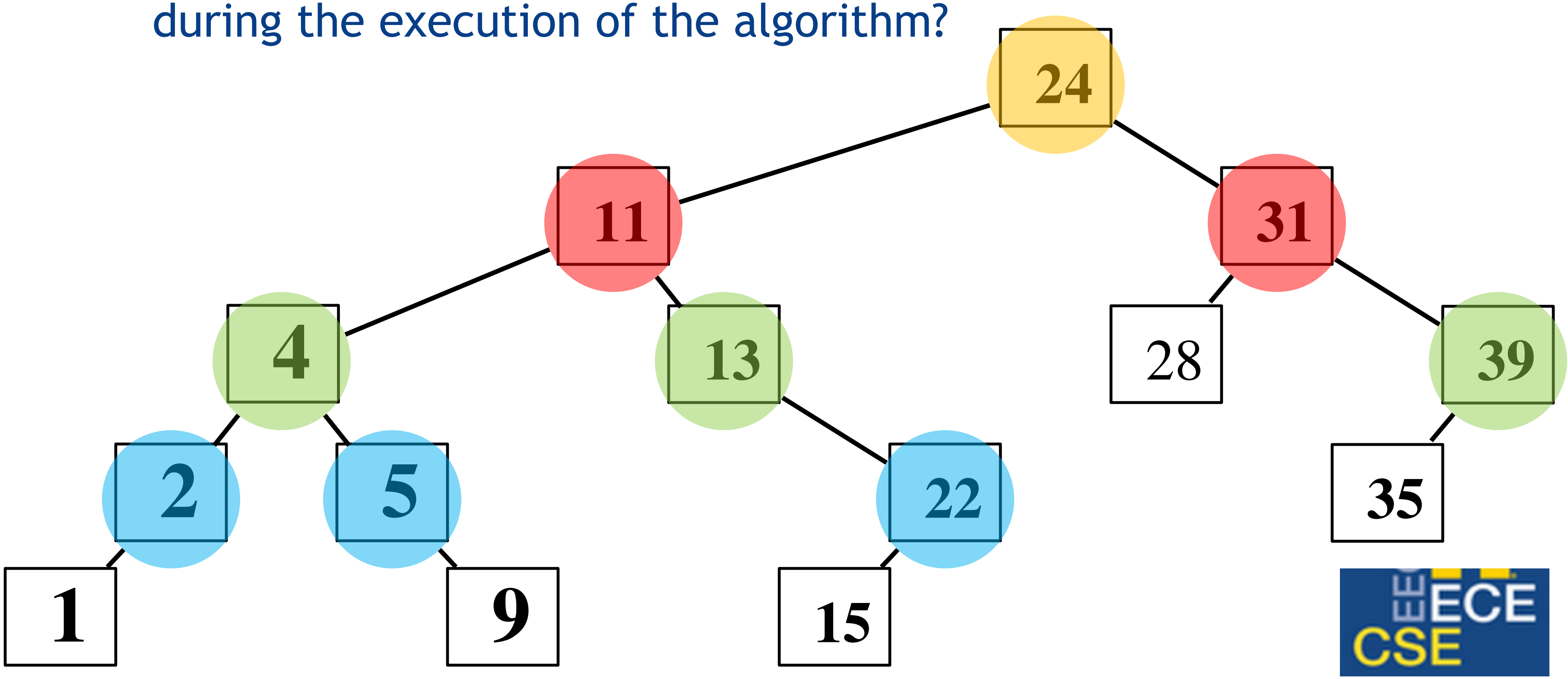
RQuicksort Example

5	28	22	11	1	35	39	24	2	4	13	9	31	15
5	22	1	11	2	4	13	9	15	**24**	28	35	39	31
5	1	2	4	9	**11**	22	13	15	**24**	28	**31**	35	39
1	2	**4**	5	9	**11**	**13**	15	22	**24**	28	**31**	**35**	**39**
1	**2**	**4**	**5**	**9**	**11**	**13**	**15**	**22**	**24**	**28**	**31**	**35**	**39**

CSE

A Different Perspective

- **Question:** *Which pairs of elements* were compared during the execution of the algorithm?



Computing $\mathbb{E}[X]$

Let r.v. X be the number of *comparisons* made in **RQuickSort** $(A[1 \dots n])$

RQuickSort$(A[1\dots n])$:
1. $p \leftarrow$ *random* "pivot" element
2. $(L, R) \leftarrow$ **Partition**(A, p) // compare elems to p
3. **RQuickSort**(L) and **RQuickSort**(R)

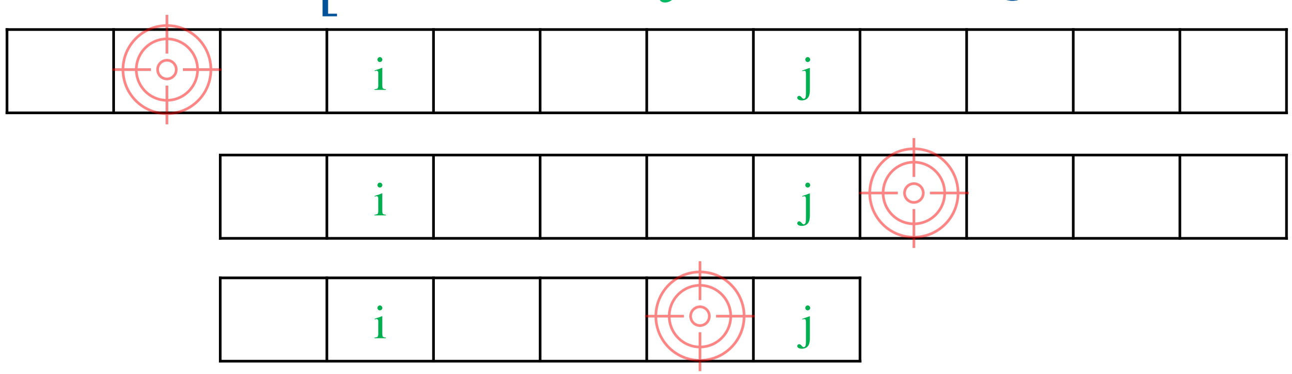
* W.l.o.g., $A[1\dots n]$ is a permutation of $\{1,2,\dots,n\}$.
* For $i < j$, let $X_{ij} \in \{0,1\}$ be the *indicator R.V.* for whether elements i and j were compared.
* **Claim:** $X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}$ (pair i, j compared at most once)

$$\mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Pr[i \text{ and } j \text{ were compared}]$$

EECS ECE CSE

Model: A Dart Game

- Let $1 \leq i < j \leq n$. Throw darts at $\{1, \ldots, n\}$:
  - If $k < i$ hit, *remove* all elements $\leq k$ and repeat.
  - If $k > j$ hit, *remove* all elements $\geq k$ and repeat.
  - If $k \in \{i, i+1, \ldots, j\}$ hit, the game ends.
- **Q:** What's $\Pr\left[\text{we hit } i \text{ or } j \text{ when the game ends}\right]$?

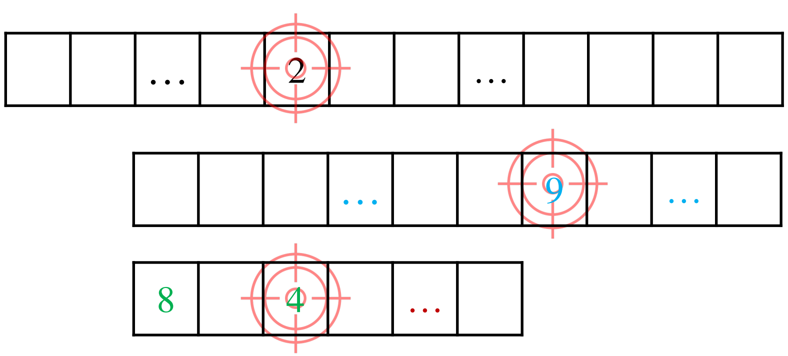


When Are i and j Compared?

RQuickSort$(A[1 \ldots n])$:
1. $p \leftarrow$ <u>*random*</u> "pivot" element
2. $(L, R) \leftarrow$ **Partition**(A, p) // compare elems to p
3. **RQuickSort**(L) and **RQuickSort**(R)

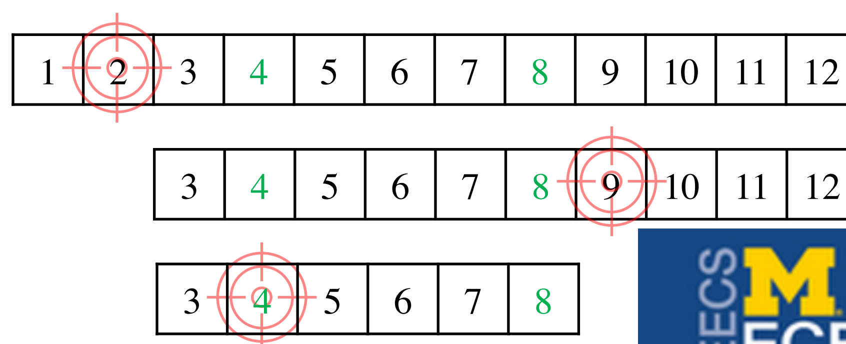
* When i, j in the same subarray <u>*and*</u> i or j chosen as pivot
* $\Longleftrightarrow$ Dart game ends with i or j being hit

RQuickSort (i=4,j=8)



Note: i and j compared only <u>*once*</u>

Dart game



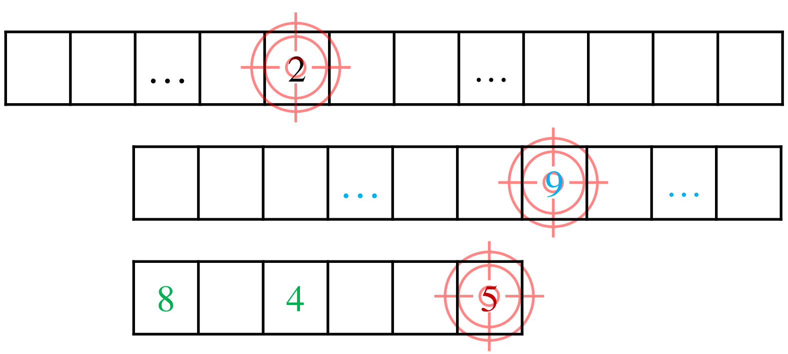
When Aren't i and j Compared?

RQuickSort$(A[1\ldots n])$:

1. $p \leftarrow$ *random* "pivot" element
2. $(L, R) \leftarrow$ **Partition**(A, p) // compare elems to p
3. **RQuickSort**(L) and **RQuickSort**(R)

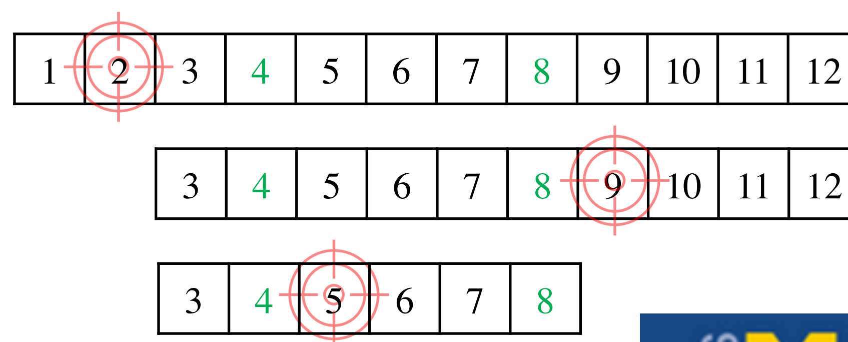
* When $i < k < j$ in same subarray *and* k chosen as pivot
* $\Longleftrightarrow$ Dart game ends with i or j *not* being hit

RQuickSort



Note: i and j *never* compared

Dart game



Putting It All Together

Let r.v. X be the number of *comparisons* made in **RQuickSort** $(A[1\ldots n])$

Let $X_{ij} \in \{0,1\}$ be the *indicator R.V.* for whether elements i and j were compared.

- **Conclusion:** i and j were compared *if and only if* dart game ends with i or j being hit. Therefore:

$$\Pr\left[i \text{ and } j \text{ were compared}\right] = 2/(j - i + 1)$$

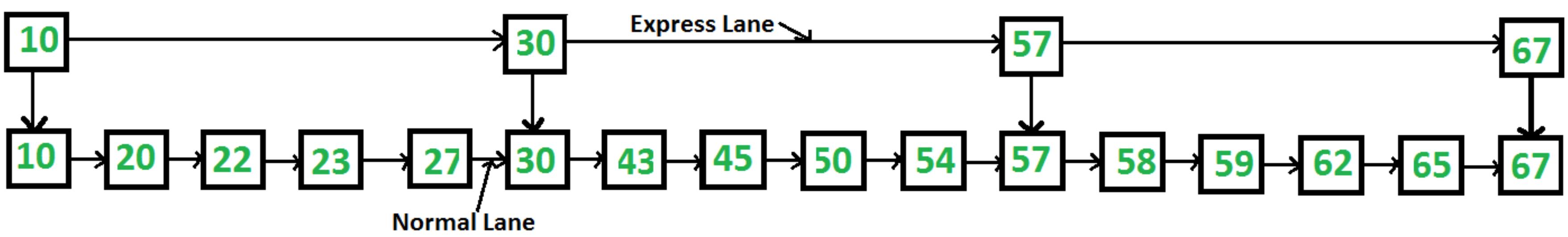
$$\begin{aligned}\mathbb{E}[X] &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}] \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Pr\left[i \text{ and } j \text{ were compared}\right] \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \quad \leq 2n\ln n = \mathrm{O}(n\log n)\end{aligned}$$

Skip Lists (Randomized Sets/Dictionaries)

* **Definition:** ***set/dictionary*** – a data structure that supports *insert*, *find* and *delete* of elements from an (ordered) universe
* **Simple implementations:** linked list, array
* **Better implementations:** balanced binary search trees (AVL, red-black, etc.)
  * $O(\log n)$ time per operation in the worst case.
  * Complicated to implement, especially "delete".
* **Idea:** Use *randomness* to get balance. But don't maintain invariants that *guarantee* balance!

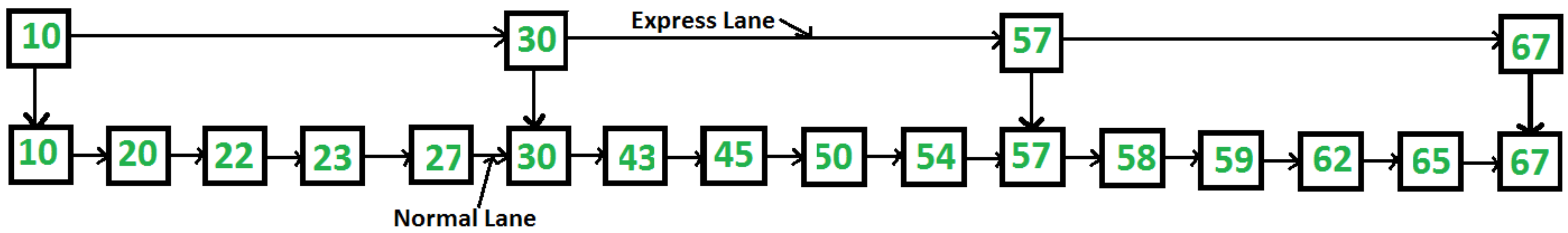
2-level Linked List

* Consider a linked list, but with an "express lane".
  * What is the worst-case number of comparisons for the example below, *without* express lane?
  * If there were $\sqrt{n}$ "well distributed" elements in the express lane, what would the search time be?
    * Insert time?
  * What is the (extra) memory size of the express lane?



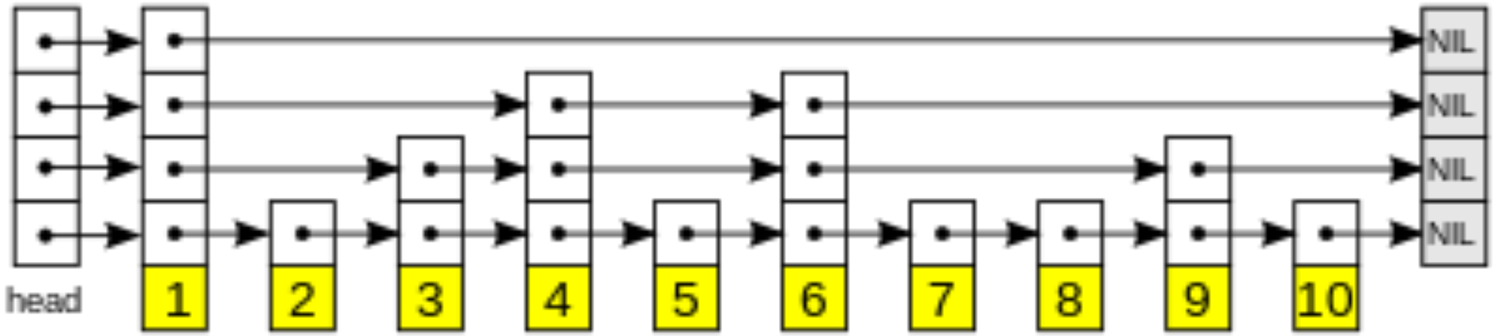
2-level Linked List

- Which elements go in the express lane?
  - Would be nice to *guarantee* balance, but random selection might work "well enough."
  - With what probability should we include an element in the express lane?



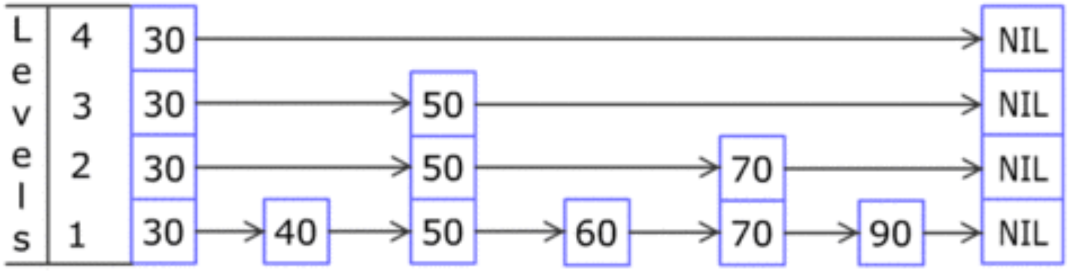
Idea: More Express Lanes

- Must put every element in the “slowest” lane.
  - Put about ½ of them in the 2nd lane
  - Put about ½ of those in the 3rd lane, etc.
  - How to build it?
  - How to add an element? How to delete?



Skip Lists

- A skip list consists of several levels.
- The first level is a linked list that contains all the elements.
- Each level contains random ~half of the elements from the level below.
- Each inserted element x is replicated by coin flipping:
  - “heads” -> replicate x to the next level and flip the coin again
  - “tails”-> done



Properties of Skip Lists

* **Question 1:** After inserting elements $x_1, \ldots, x_n$, how many are on level i?
  * $n_i =$ number of elements on level i
    $n_1 = n$ (lowest level contains all the elements)
  * Define X_{ij} indicator variable for event "x_j is on level i".
    $\mathbb{E}\left[n_i\right] = \mathbb{E}\left[X_{i1} + \ldots + X_{in}\right]$
    $= \sum_j \mathbb{E}\left[X_{ij}\right]$ *(linearity of expectation)*
    $= \sum_j 1/2^{i-1}$ *(had to flip $\geq i-1$ heads to reach level i)*
    $= n/2^{i-1}$

Properties of Skip Lists

- **Question 2:** How many levels for a list of n elements?
  - We keep only levels containing at least one element; expect at least $\log n$ (base 2) levels.
  - What is the probability that we need at least k additional levels?

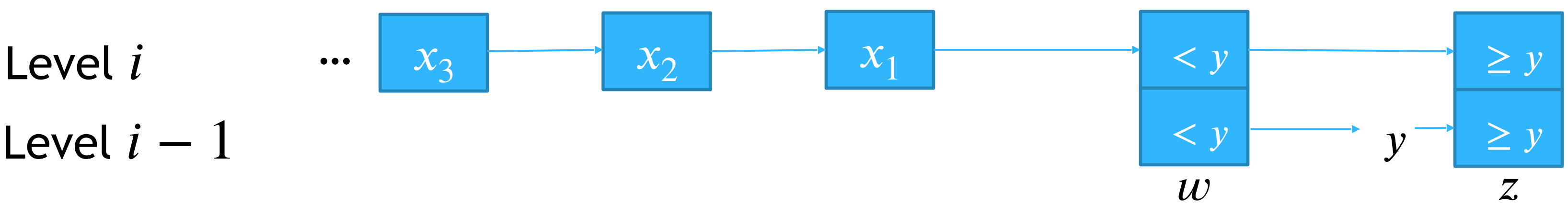
$$\Pr\left[n_{\log n+k} \geq 1\right] \leq \frac{\mathbb{E}\left[n_{\log n+k}\right]}{1} \qquad \textit{(Markov's inequality)}$$

$$= \frac{n}{2^{\log n+k-1}} = \frac{n}{n2^{k-1}} = \frac{1}{2^{k-1}}$$

  - How many levels do we need *in expectation*?
    - Define Z_k as an indicator for the event that $n_{\log n+k} \geq 1$.
    - Expected number of levels is at most $\log n + \mathbb{E}\left[Z_1\right] + \mathbb{E}\left[Z_2\right] + \mathbb{E}\left[Z_3\right] + \ldots = O(\log n)$

Properties of Skip Lists

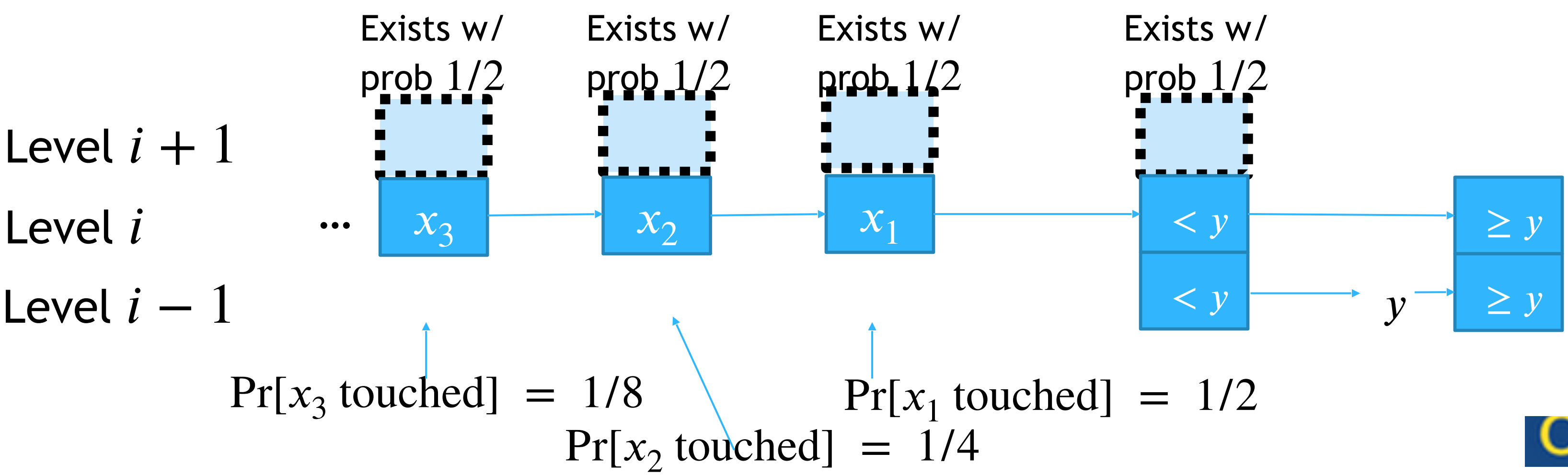
- **Question 3:** How many nodes does a search touch on level i?
  - Suppose we are searching for y. Let w be the largest element on level i that is $< y$, and let z be the smallest element that is $\geq y$.
  - The search encounters both w and z. How many elements before w does it touch?
  - Let $x_1, x_2, \ldots$ be the elements on level i that are $< w$.



CSE

Properties of Skip Lists

- **Question 3:** How many nodes does a search touch on level i?
  - Let Z_j = indicator r.v. for the event that x_j is touched on level i.
  - Let Y = total number of elements touched on level i.
  - $= 2 + \sum_{j \geq 1} \Pr\left[x_j \text{ touched}\right] \quad = 2 + \sum_{j \geq 1} 1/2^j \leq 3$



Bonus: Verifying Matrix Mult

Goal: Given n-by-n matrices A, B, C, check whether $AB = C$.

Trivial: Compute AB, check if $AB = C$. Naïve matrix-mult time: $O(n^3)$.

Using randomization, can do it in $O(n^2)$ time! **Algorithm:**

* Choose a uniformly random vector r with each entry 0 or 1.
* Check if $A(Br) = Cr$.

Running time: $O(n^2)$. (Compute $v = Br$, then Av.)

Correctness: If $AB = C$, we accept with certainty.
Claim: If $AB \neq C$, then $\Pr[\text{accept}] \leq 1/2$. (Repeat to reduce!)

Proof of Claim

Claim: If $AB \neq C$, then $\Pr[ABr = Cr] \leq 1/2$.

Proof: Let $D = AB - C \neq 0$. (D does not have all-zero entries.)
We want to show that $\Pr[Dr \neq \mathbf{0}] \geq 1/2$.

Suppose (wlog) that column $D_1 \neq \mathbf{0}$.
Fix *any* choice of the entries $r_2, \ldots, r_n$ (so only random r_1 remains).

$$Dr = r_1 D_1 + \underbrace{r_2 D_2 + \cdots + r_n D_n}_{\text{fixed } z}.$$

Conclusion: Dr cannot be **0** for *both* $r_1 = 0$ and $r_1 = 1$. QED.