Applied Algorithms

CSCI-B505 / INFO-I500

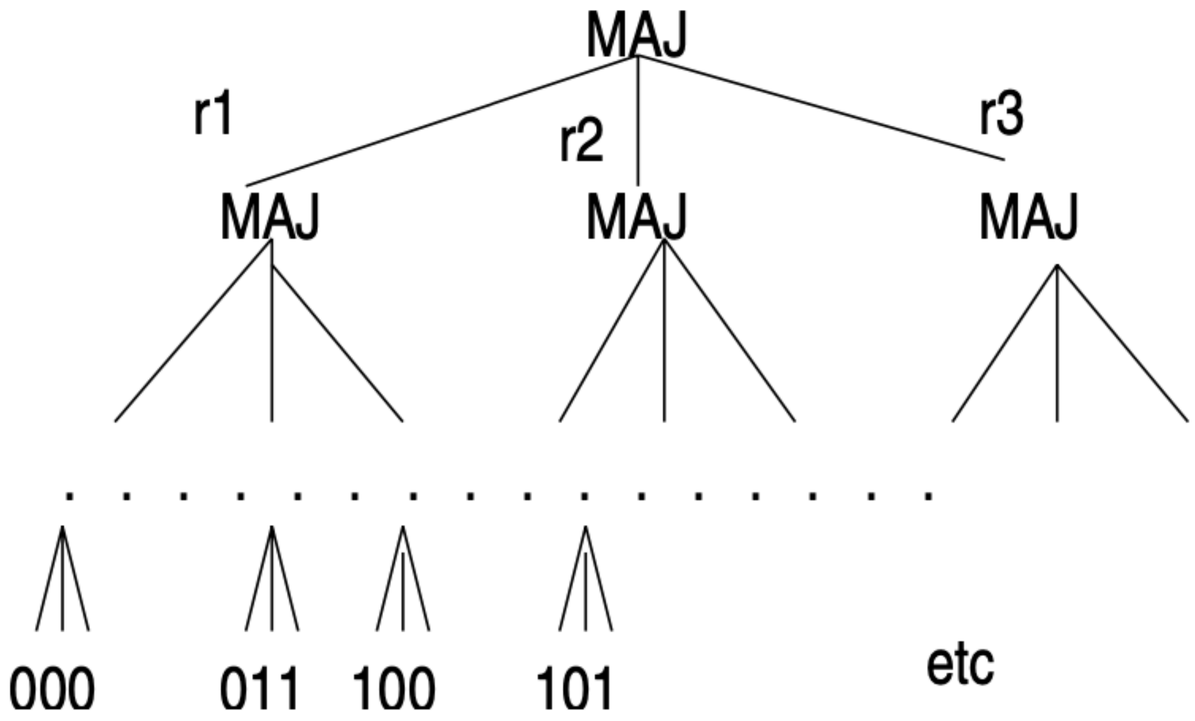
Lecture 19.

Randomized Algorithms - II

M. Oguzhan Kulekci

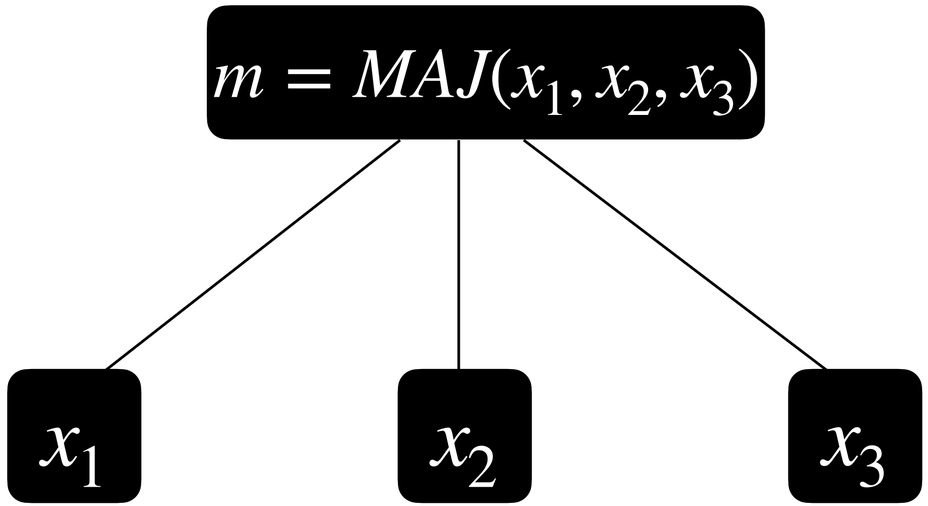
- Majority Tree Evaluation
- Approximate Median Revisited
- Matrix Multiplication Verification
- Randomized 3SAT

Majority Tree



- A ternary tree of depth n
- Leaf nodes are 0 or 1
- Inner nodes are **MAJ** gates that output the value of the majority value of its children, e.g. if children are 0, 0, and 1 then the output is 0, which will be fed to the upper level.
- We can evaluate the result by traversing all nodes that is $O(3^n)$
- Can randomization help to reduce the number of visited nodes and still provide the correct result?
- Since we need the correct result, it should be a **Las Vegas** type randomization.

Majority Tree



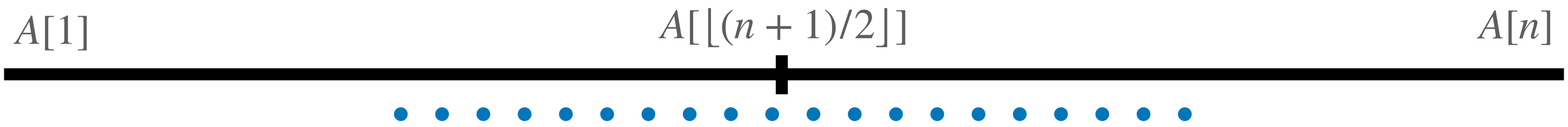
- Assume we perform a breads-first bottom-up traversal! A DFS would be more advantageous though
- On the average we visit (8/3 = 2.66) children instead of 3 to decide the majority, and thus, the expected complexity reduces to $O(2.66^n)$

$$\frac{2}{3} \cdot 3 + \frac{1}{3} \cdot 2 = \frac{8}{3}$$

- Chose **randomly** 2 from x_1, x_2, x_3.
- There are 3 different ways of that selection.
- Regardless of the values of the x_i, at least one of those selections removes the necessity of visiting the third one !
- Then, with such a randomization with at least 1/3 probability, no need to visit the third.

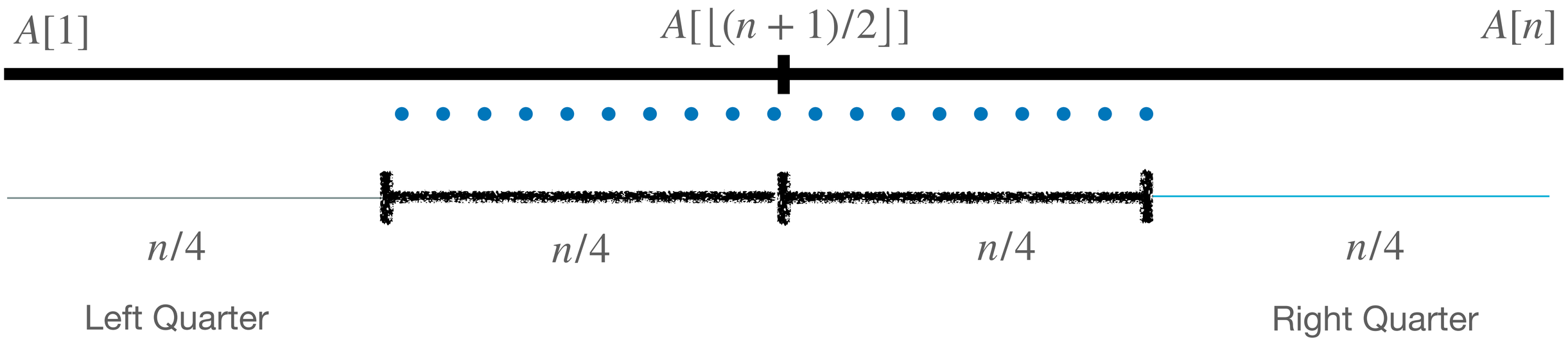
$x_1x_2x_3$	x_1x_2	x_1x_3	x_2x_3
000	00	00	00
001	00	01	01
010	01	00	10
011	01	01	11
100	10	10	00
101	10	11	01
110	11	10	10
111	11	11	11

Approximate Median Revisited



- For the γ-approximate median, where $0 < \gamma \leq 1/2$, we had
  - **Monte Carlo** algorithm that runs in $O(c \log n)$ with a failure prob $(1 - 2\gamma)^c$
  - **Las Vegas** algorithm that returns a correct answer expectedly in $O(n/\gamma)$ time
- Now we assume $\gamma = 1/4$, and try to come up with a Monte Carlo type algorithm that runs in $O(\log n \log \log n)$ time with a failure probability of $2n^{-c}$ over n numbers

Approximate Median Revisited



- Assume we randomly select k integers, sort them and return the median.
- When is that computed not in the shaded region and thus, erroneous ?
  - That happens if at least $k/2$ are chosen from the left quarter (or the right quarter).
  - The probability that $k/2$ are from the left quarter is $(1/2)^{k/5}$ (with some detailed probability calculations)
  - So, if we chose $k = c \log n$, it is $n^{-c/5}$.
  - Also considering the right quarter, the total failure probability is $2n^{-c/5}$

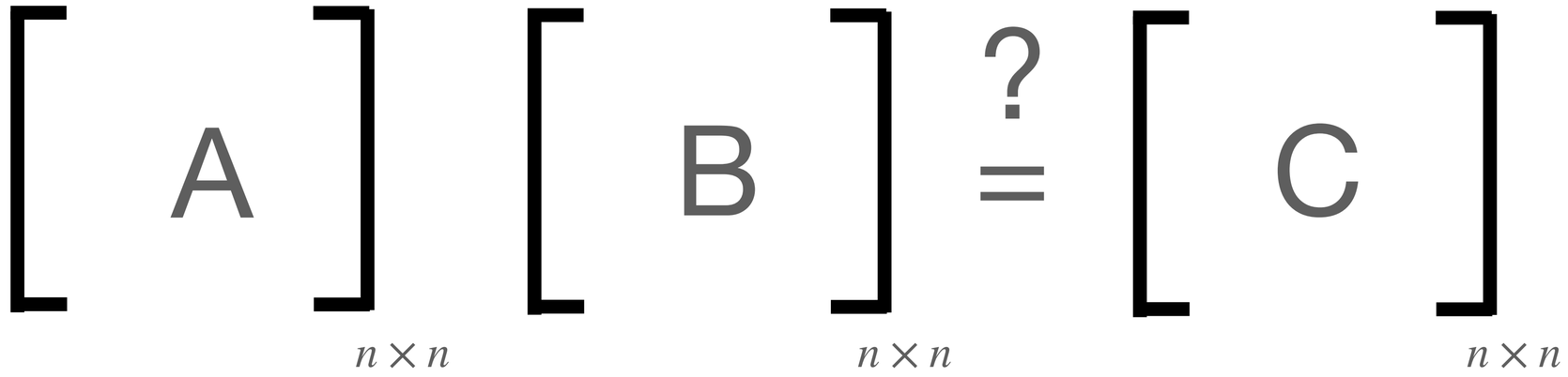
Approximate Median Revisited

```
Rand-Approx-Median(A)
1.  Let k ← c log n;
2.  S ← ∅;
3.  For i=1 to k
4.        x ← an element selected randomly uniformly from A;
5.        S ← S U {x};
6.  Sort S.
7.  Report the median of S.
```

- Complexity is $O(\log n \log \log n)$
- Assume $c = 10$, then the failure probability is $2n^{-c/5} = 2n^{-2}$
- Particularly efficient for large n

Matrix Multiplication Verification

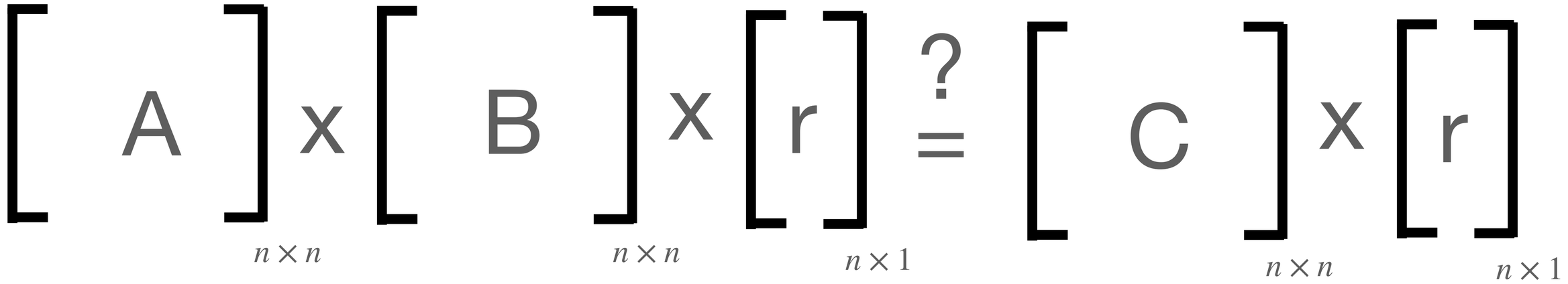
Freivald's Algorithm



- We would like to verify whether $A \times B = C$, where all are $n \times n$ matrices.
- We can repeat the operation, and then check, $O(n^3)$ time
- Freivald's randomized algorithm performs verification in $O(kn^2)$ time with a failure probability of 2^{-k}. (Yes, a Monte Carlo type)

Matrix Multiplication Verification

Freivald's Algorithm

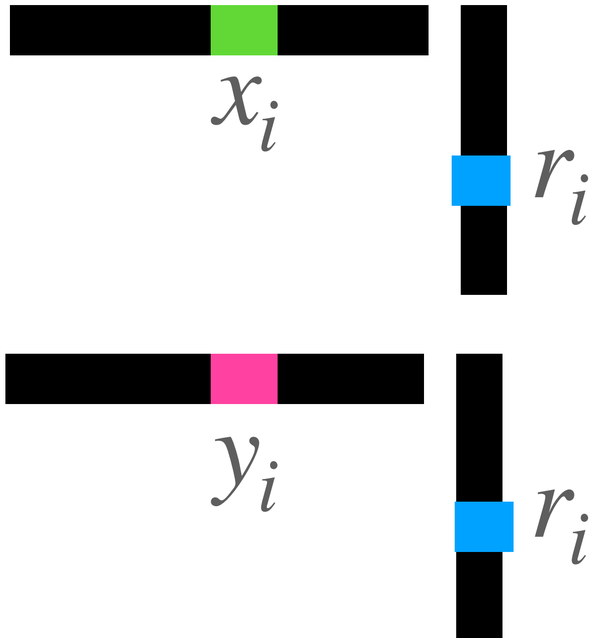


- Pick a random vector $r = [r_1\ r_2\ r_3 \ldots r_n]$, where $r_i \in \{0,1\}$.
- If $A \times B \times r = C \times r$, then output YES, otherwise NO.
- If NO, then $A \times B = C$ is not correct, but otherwise, there is a probability that the multiplication is wrong, but the test generated a false true.
- The time complexity is $O(n^2)$.

Matrix Multiplication Verification

Freivald's Algorithm

- **What is the probability of that failure in Freivald's method ?**
- Let x and y be two vectors and r is a vector of randomly selected 0s and 1s.
- If $x \neq y$, it is still possible that $x^t r = y^t r$.
- If $x \neq y$, then there is at least one dimension, where $x_i \neq y_i$



$x_i r_i + \alpha = y_i r_i + \beta$, where α and β are the sum of the remaining dimensions

	$\alpha = \beta$	$\alpha \neq \beta$
$r_i = 0$		X
$r_i = 1$	X	

X marks the cases that reveal $x \neq y$.

With 1/2 probability, the verification might be wrong.

Matrix Multiplication Verification

Freivald's Algorithm

- **What is the probability of that failure in Freivald's method ?**
- Assume $D = AB$, and we check whether $Dr = Cr$.
- The previous analysis showed that it may hide the truth with 1/2 probability.
- Therefore, if we apply it k times, the failure probability is 2^{-k} and the time complexity becomes $O(k \cdot n^2)$.
- Notice that this is better than $O(n^3)$.

Randomized Approximation for MAX 3-SAT

- 3-SAT: Given k clauses $C_1, C_2, \ldots, C_k$, where each clause is the disjunction of three variables from $X = \{x_1, x_2, \ldots, x_n\}$, does there a truth assignment that satisfies all clauses?

- The core problem in NP-completeness.

- MAX 3-SAT: Satisfy as many clauses as possible.

Interestingly, a random assignment of truth values to X is expected to satisfy 7/8 of all k clauses !

Randomized Approximation for MAX 3-SAT

- $(x_i \vee x_j \vee x_k)$ turns false only when $x_i = x_j = x_k = 0$.
- Since we randomly selected the truth values for variables, each variable is 0 or 1 with probability 1/2
- Therefore, per each clause, the probability that it becomes false is 1/8, which means any random assignment satisfies the clause with 7/8 probability.
- Assume $Z_i = 1$ if clause C_i is satisfied, where $E[Z_i] = 7/8$.
- $E[Z_1 + Z_2 + \ldots + Z_k] = E[Z_1] + E[Z_2] + \ldots + E[Z_k] = k \cdot \frac{7}{8}$

This implies there should be a truth assignment for every SAT formula that satisfies 7/8 of the clauses !