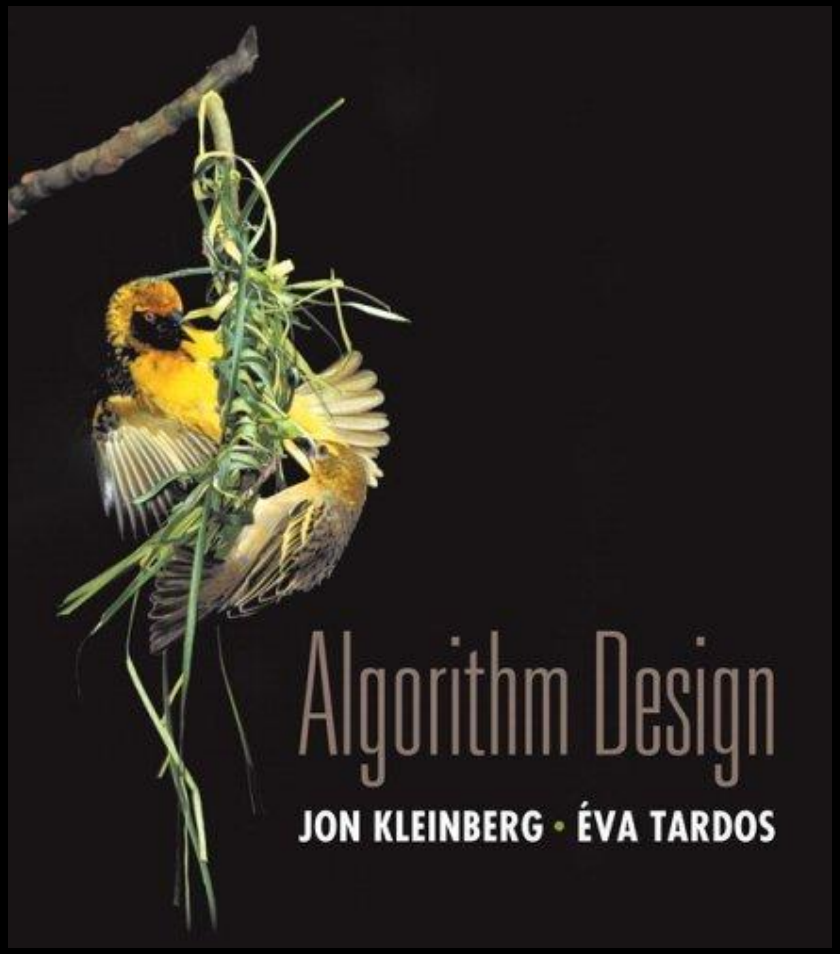



# Chapter 13

## Randomized Algorithms

# 13.3 Linearity of Expectation

# Random Variable

Definition. A random variable X on a sample space $\Omega$ is real-valued function, that is, it associates each subset of $\Omega$ with a real value.

- We focus only on finite or uncountably finite sample spaces to avoid some technical issues.

Example. Consider an experiment that consists of rolling two fair dices with 6 faces

We can define the sample space $\Omega=\{(i,j)|\ 1 <=i<=6$ e $1<=j<=6\}$ and a random variable X, where X(a,b)=max{a,b}

We have that Pr{X=3} = 5/36 and Pr{X=6}=11/36

# Random Variable

Definition. A random variable X on a sample space $\Omega$ is real-valued function, that is, it associates each subset of $\Omega$ with a real value.

- We focus only on finite or uncountably finite sample spaces to avoid some technical issues.

Example. Consider an experiment that consists of rolling two fair dices with 6 faces

We can define the sample space $\Omega=\{(i,j)|\ 1<=i<=j<=6\}$ and a random variable X, where X(a,b)=max{a,b}

We have that Pr{X=3} = 5/36 and Pr{X=6}=11/36

# Expectation

Seção 6.3 do Cormen

Expectation. Given a discrete random variables X, its expectation E[X] is defined by:

$$E[X] = \sum_{j=0}^{\infty} j \Pr[X = j]$$

X(a,b) = max{a,b}, where (a,b) is the outcomoe of two fair dices

E[X] = 1*1/36 +2*3/36+3*3/36+ 4*7/36 +5*9/36 + 6*11/36

# Expectation

Seção 6.3 do Cormen

**Expectation.** Given a discrete random variables X, its expectation E[X] is defined by:

$$E[X] = \sum_{j=0}^{\infty} j \Pr[X = j]$$

**Waiting for a first success.** Coin is head with probability p and tails with probability 1-p. How many independent flips X until first heads?

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} j (1-p)^{j} = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

j-1 tails  1 head

# Expectation: Two Properties

**Useful property.** If X is a 0/1 random variable, E[X] = Pr[X = 1].

**Pf.**

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{1} j \cdot \Pr[X = j] = \Pr[X = 1]$$

not necessarily independent

**Linearity of expectation.** Given two random variables X and Y defined over the same probability space, E[X + Y] = E[X] + E[Y].

Decouples a complex calculation into simpler pieces.

# Guessing Cards

Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Memoryless guessing. No psychic abilities; can't even remember what's been turned over already. Guess a card from full deck uniformly at random.

Claim. The expected number of correct guesses is 1.

Pf. (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if $i^{th}$ prediction is correct and 0 otherwise.
- Let $X$ = number of correct guesses = $X_1 + \ldots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/n = 1$. ▪

↑ linearity of expectation

# Guessing Cards

Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Guessing with memory. Guess a card uniformly at random from cards not yet seen.

Claim. The expected number of correct guesses is $\Theta(\log n)$.

Pf.

- Let $X_i = 1$ if $i^{th}$ prediction is correct and 0 otherwise.
- Let $X$ = number of correct guesses = $X_1 + \ldots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1 / (n - i - 1)$.
- $E[X] = E[X_1] + \ldots + E[X_n] = 1/n + \ldots + 1/2 + 1/1 = H(n)$. ▪

↑ linearity of expectation

↑ $\ln(n+1) < H(n) < 1 + \ln n$

# Coupon Collector

Coupon collector. Each box of cereal contains a coupon. There are n different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have $\geq 1$ coupon of each type?

Claim. The expected number of steps is $\Theta(n \log n)$.

Pf.

- Phase j = time between j and j+1 distinct coupons.
- Let $X_j$ = number of steps you spend in phase j.
- Let X = number of steps in total = $X_0 + X_1 + \ldots + X_{n-1}$.

$$E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = n H(n)$$

prob of success = (n-j)/n
$\Rightarrow$ expected waiting time = n/(n-j)

# 13.5 Randomized Divide-and-Conquer

# Quicksort

Sorting. Given a set of n distinct elements S, rearrange them in ascending order.

```
RandomizedQuicksort(S) {
   if |S| = 0 return

   choose a splitter a_i ∈ S uniformly at random
   foreach (a ∈ S) {
      if      (a < a_i) put a in S^-
      else if (a > a_i) put a in S^+
   }
   RandomizedQuicksort(S^-)
   output a_i
   RandomizedQuicksort(S^+)
}
```

Remark. Can implement in-place.

↑

O(log n) extra space

# Quicksort

Running time.

- [Best case.] Select the median element as the splitter: quicksort makes $\Theta(n \log n)$ comparisons.
- [Worst case.] Select the smallest element as the splitter: quicksort makes $\Theta(n^2)$ comparisons.

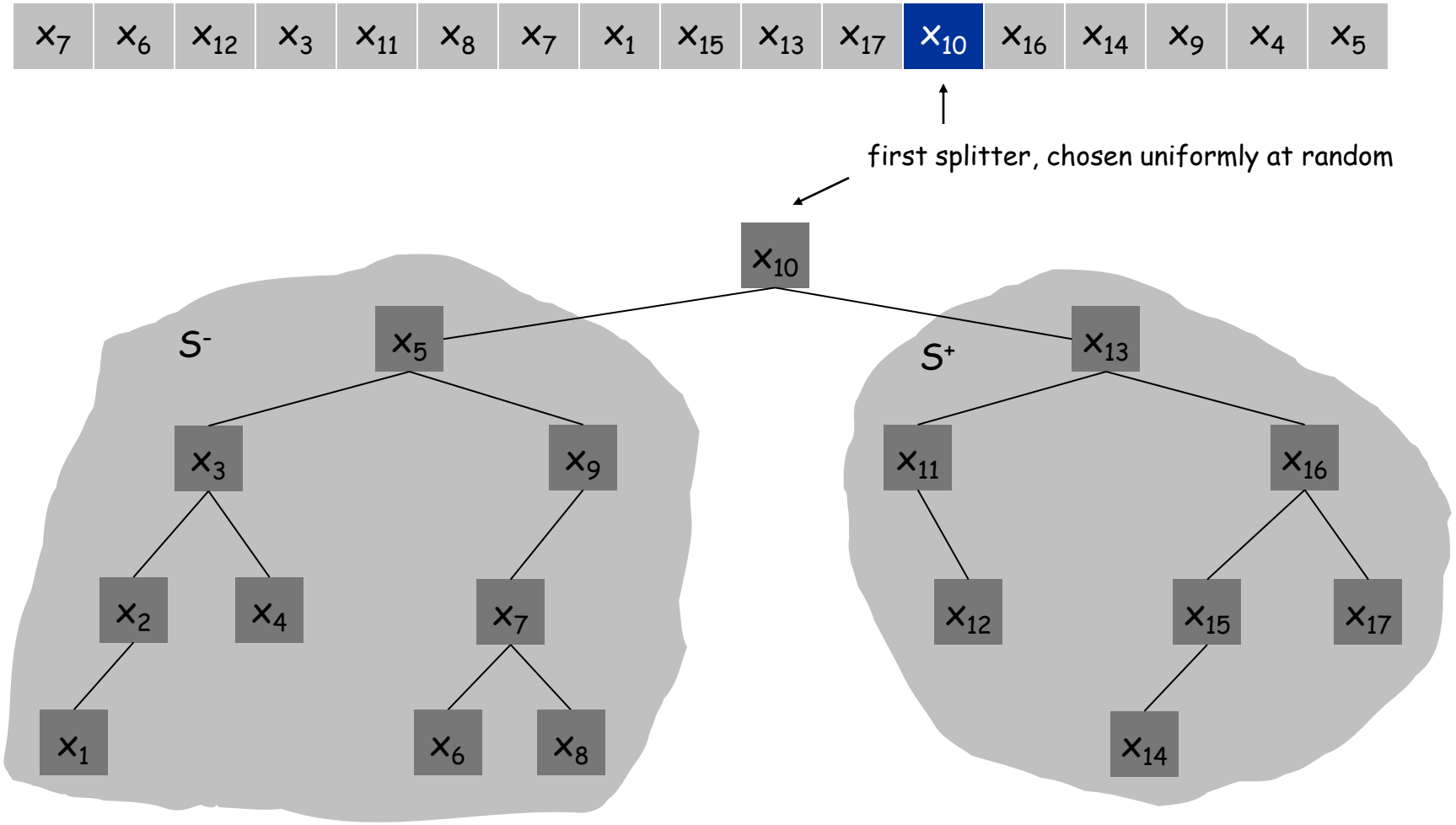
Randomize. Protect against worst case by choosing splitter at random.

Intuition. If we always select an element that is bigger than 25% of the elements and smaller than 25% of the elements, then quicksort makes $\Theta(n \log n)$ comparisons.

Notation. Label elements so that $x_1 < x_2 < \ldots < x_n$.

# Quicksort: BST Representation of Splitters

BST representation. Draw recursive BST of splitters.

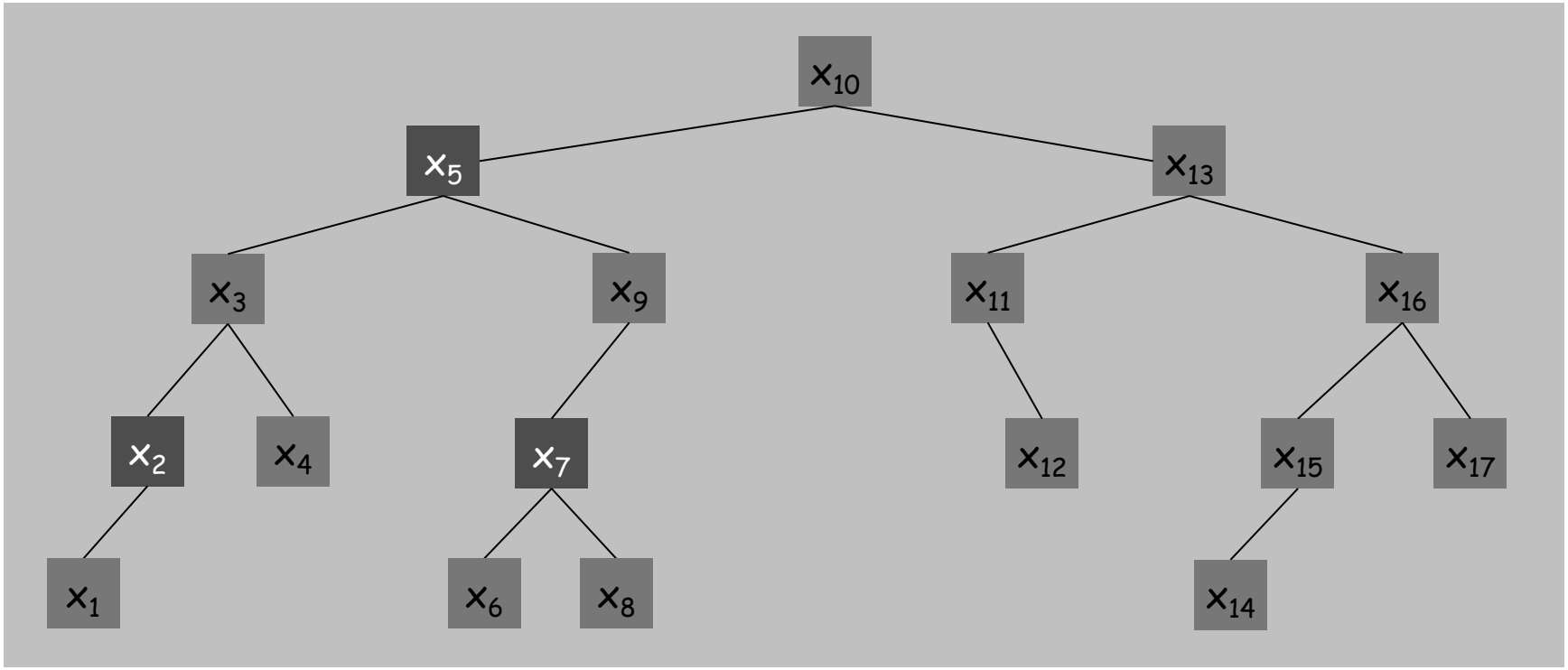


# Quicksort: BST Representation of Splitters

Observation. Element only compared with its ancestors and descendants.

- $x_2$ and $x_7$ are compared if their lca = $x_2$ or $x_7$.
- $x_2$ and $x_7$ are not compared if their lca = $x_3$ or $x_4$ or $x_5$ or $x_6$.

Claim. Pr[$x_i$ and $x_j$ are compared] = $2 / |j - i + 1|$.



# Quicksort: Expected Number of Comparisons

Theorem. Expected # of comparisons is $O(n \log n)$.

Pf.

$$\sum_{1 \le i < j \le n} \frac{2}{j - i + 1} = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j} \le 2n \sum_{j=1}^{n} \frac{1}{j} \approx 2n \ln n$$

# Quicksort: Expected Number of Comparisons

Theorem. Expected # of comparisons is $O(n \log n)$.

Pf.

$$\sum_{1 \le i < j \le n} \frac{2}{j - i + 1} = 2 \sum_{i=1}^{n} \sum_{j=2}^{i} \frac{1}{j} \le 2n \sum_{j=1}^{n} \frac{1}{j} \approx 2n \int_{x=1}^{n} \frac{1}{x} dx = 2n \ln n$$

probability that i and j are compared

Theorem. [Knuth 1973] Stddev of number of comparisons is ~ 0.65N.

Ex. If n = 1 million, the probability that randomized quicksort takes less than 4n ln n comparisons is at least 99.94%.

Chebyshev's inequality. $\Pr[|X - \mu| \ge k\delta] \le 1 / k^2$.

# 13.4 MAX 3-SAT

# Maximum 3-Satisfiability

exactly 3 distinct literals per clause

MAX-3SAT. Given 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

$$
\begin{aligned}
C_1 &= x_2 \vee \overline{x_3} \vee \overline{x_4} \\
C_2 &= x_2 \vee x_3 \vee \overline{x_4} \\
C_3 &= \overline{x_1} \vee x_2 \vee x_4 \\
C_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
C_5 &= x_1 \vee \overline{x_2} \vee \overline{x_4}
\end{aligned}
$$

Remark. NP-hard search problem.

# Maximum 3-Satisfiability

exactly 3 distinct literals per clause

MAX-3SAT. Given 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

$$
\begin{aligned}
C_1 &= x_2 \vee \overline{x_3} \vee \overline{x_4} \\
C_2 &= x_2 \vee x_3 \vee \overline{x_4} \\
C_3 &= \overline{x_1} \vee x_2 \vee x_4 \\
C_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
C_5 &= x_1 \vee \overline{x_2} \vee \overline{x_4}
\end{aligned}
$$

Remark. NP-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability $\frac{1}{2}$, independently for each variable.

# Maximum 3-Satisfiability: Analysis

Claim. Given a 3-SAT formula with k clauses, the expected number of clauses satisfied by a random assignment is 7k/8.

Pf. Consider random variable $Z_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$

- Let Z = number of clauses satisfied by the assignment.

$$\begin{aligned} E[Z] &= \sum_{j=1}^{k} E[Z_j] \\ &= \sum_{j=1}^{k} \Pr[\text{clause } C_j \text{ is satisfied}] \\ &= \tfrac{7}{8} k \end{aligned}$$

linearity of expectation

# The Probabilistic Method

Corollary. For any instance of 3-SAT, there exists a truth assignment that satisfies at least a 7/8 fraction of all clauses.

Pf. Random variable is at least its expectation some of the time. ▪

Probabilistic method. We showed the existence of a non-obvious property of 3-SAT by showing that a random construction produces it with positive probability!

# Maximum 3-Satisfiability: Analysis

Q. Can we turn this idea into a 7/8-approximation algorithm? In general, a random variable can almost always be below its mean.

Lemma. The probability that a random assignment satisfies $\geq 7k/8$ clauses is at least $1/(8k)$.

Pf. Let $p_j$ be probability that exactly j clauses are satisfied; let p be probability that $\geq 7k/8$ clauses are satisfied.

$$\begin{aligned}
\tfrac{7}{8}k \;=\; E[Z] \;&=\; \sum_{j \geq 0} j\, p_j \\
&=\; \sum_{j < 7k/8} j\, p_j \;+\; \sum_{j \geq 7k/8} j\, p_j \\
&\leq\; (\tfrac{7k}{8} - \tfrac{1}{8}) \sum_{j < 7k/8} p_j \;+\; k \sum_{j \geq 7k/8} p_j \\
&\leq\; (\tfrac{7}{8}k - \tfrac{1}{8}) \cdot 1 \;+\; k\, p
\end{aligned}$$

Rearranging terms yields $p \geq 1 / (8k)$. ▪

# Maximum 3-Satisfiability: Analysis

Johnson's algorithm. Repeatedly generate random truth assignments until one of them satisfies $\geq 7k/8$ clauses.

Theorem. Johnson's algorithm is a 7/8-approximation algorithm.

Pf. By previous lemma, each iteration succeeds with probability at least $1/(8k)$. By the waiting-time bound, the expected number of trials to find the satisfying assignment is at most $8k$. ▪

# Maximum Satisfiability

Extensions.

- Allow one, two, or more literals per clause.
- Find max weighted set of satisfied clauses.

Theorem. [Asano-Williamson 2000] There exists a 0.784-approximation algorithm for MAX-SAT.

Theorem. [Karloff-Zwick 1997, Zwick+computer 2002] There exists a 7/8-approximation algorithm for version of MAX-3SAT where each clause has at most 3 literals.

Theorem. [Håstad 1997] Unless P = NP, no $\rho$-approximation algorithm for MAX-3SAT (and hence MAX-SAT) for any $\rho > 7/8$.

very unlikely to improve over simple randomized algorithm for MAX-3SAT

# Monte Carlo vs. Las Vegas Algorithms

Monte Carlo algorithm. Guaranteed to run in poly-time, likely to find correct answer.

Ex: Contraction algorithm for global min cut.

Las Vegas algorithm. Guaranteed to find correct answer, likely to run in poly-time.

Ex: Randomized quicksort, Johnson's MAX-3SAT algorithm.

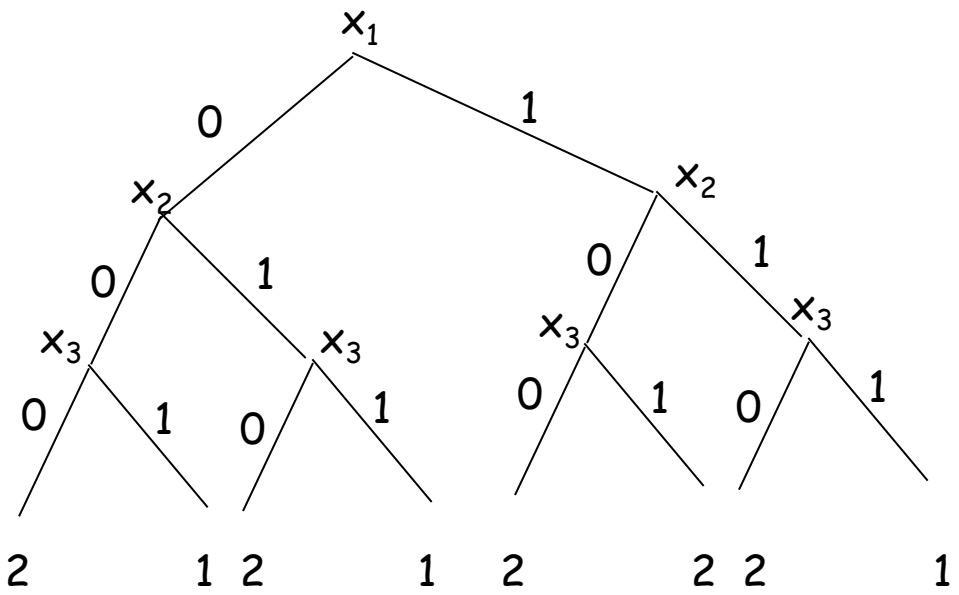
stop algorithm after a certain point



Remark. Can always convert a Las Vegas algorithm into Monte Carlo, but no known method to convert the other way.

# MAX SAT: Desaleatorização 5.6

$C_1 = (x_1 \vee x_2 \vee \neg x_3)$
$C_2 = (\neg x_2 \vee \neg x_3)$

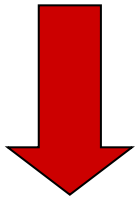



- Cada folha da árvore corresponde a um atribuição:
- Cada folha esta associada ao número de clausulas satisfeitas pela atribuição correspondente

# MAX SAT: Desaleatorização

**Método das Probabilidades Condicionais**

$$E[X] = 1/2.E[X|x_1=1]+1/2.E[X|x_1=0] = 7/8\ k$$



(i) $E[X|x_1=1] >= 7/8k$ ou

(ii) $E[X|x_1=0] >= 7/8k$

- Se (i) fixamos $x_1=1$, caso contrário fixamos $x_1=0$.
- Consideramos $x_2$ recursivamente e descemos até chegar uma folha