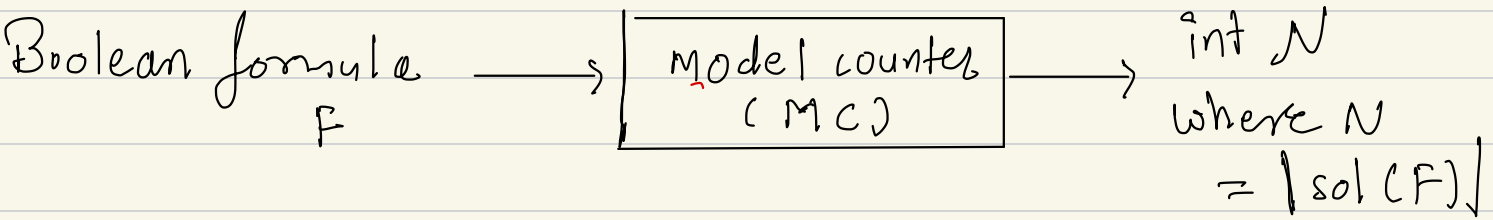
Model counting :-

* How often system S satisfies property P?

→ We are now interested in finding how many solutions (models) are there for given set of constraints.



```
MC-Naive(F)                    n = |variables of F|
{
  count = 0
  for all τ ← 2^n do
      if τ ⊨ F then                  ──→ Given access to sat solver.
          count++                        (NP oracle)
  return count
}
```

Model counting by Recursion

```
MC(F)                    n = |variables(F)|
{

  pick x ← vars(F)        → variables of F
  C0 = MC(F(x↦0))
  C1 = MC(F(x↦1))
  return C0 + C1
}
```

Model counting by Recursion

MC (F) $n = |\text{variables}(F)|$

{

if F is 0 then return 0
if F is 1 then return 1.

pick $x \leftarrow$ vars(F) → variables of F

C_0 = MC (F(x ← 0))
C_1 = MC (F(x ← 1))
return $C_0 + C_1$

}

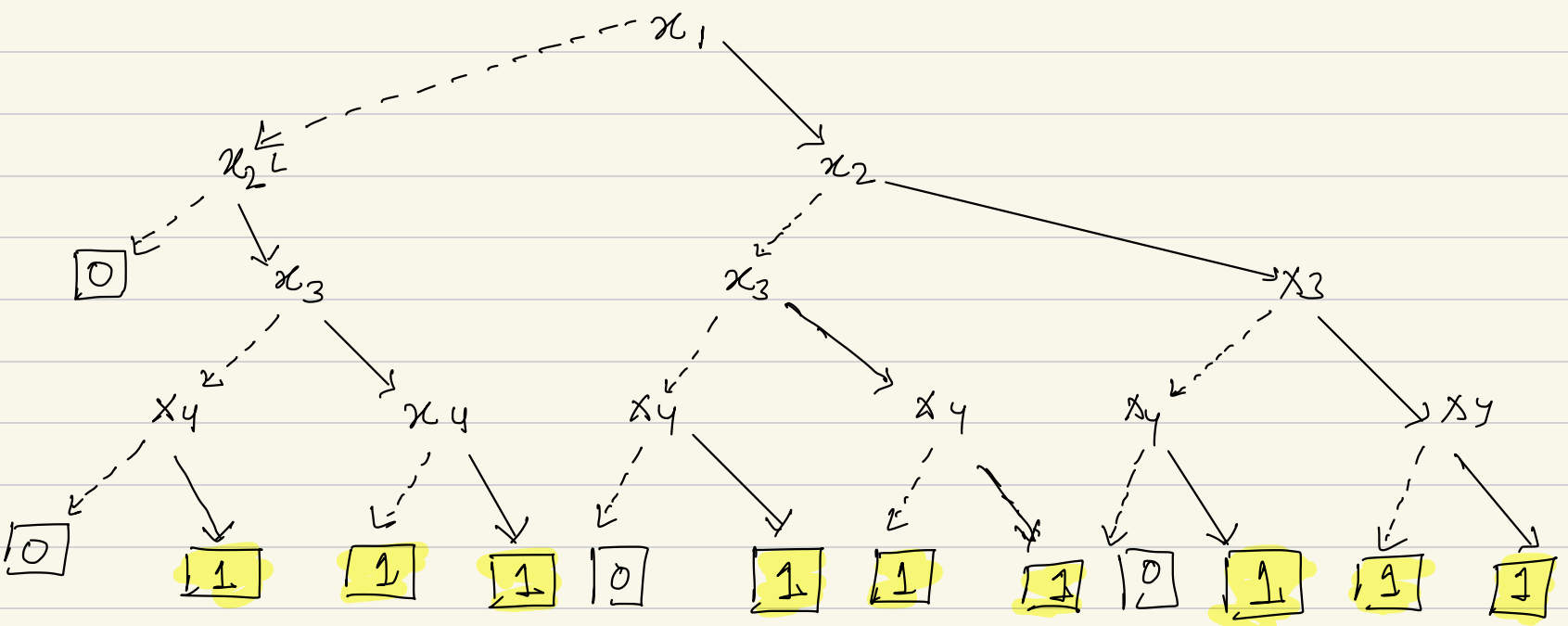
Model counting by Recursion

```
MC(F)                    n = |variables(F)|
{
  if F is 0 then return 0
  if F is 1 then return 1.

  pick x ← vars(F)        → variables of F

  C0 = MC(F(x↦0))
  C1 = MC(F(x↦1))
  return C0 + C1
}
```

Find $|sol(F)|$ where $F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$

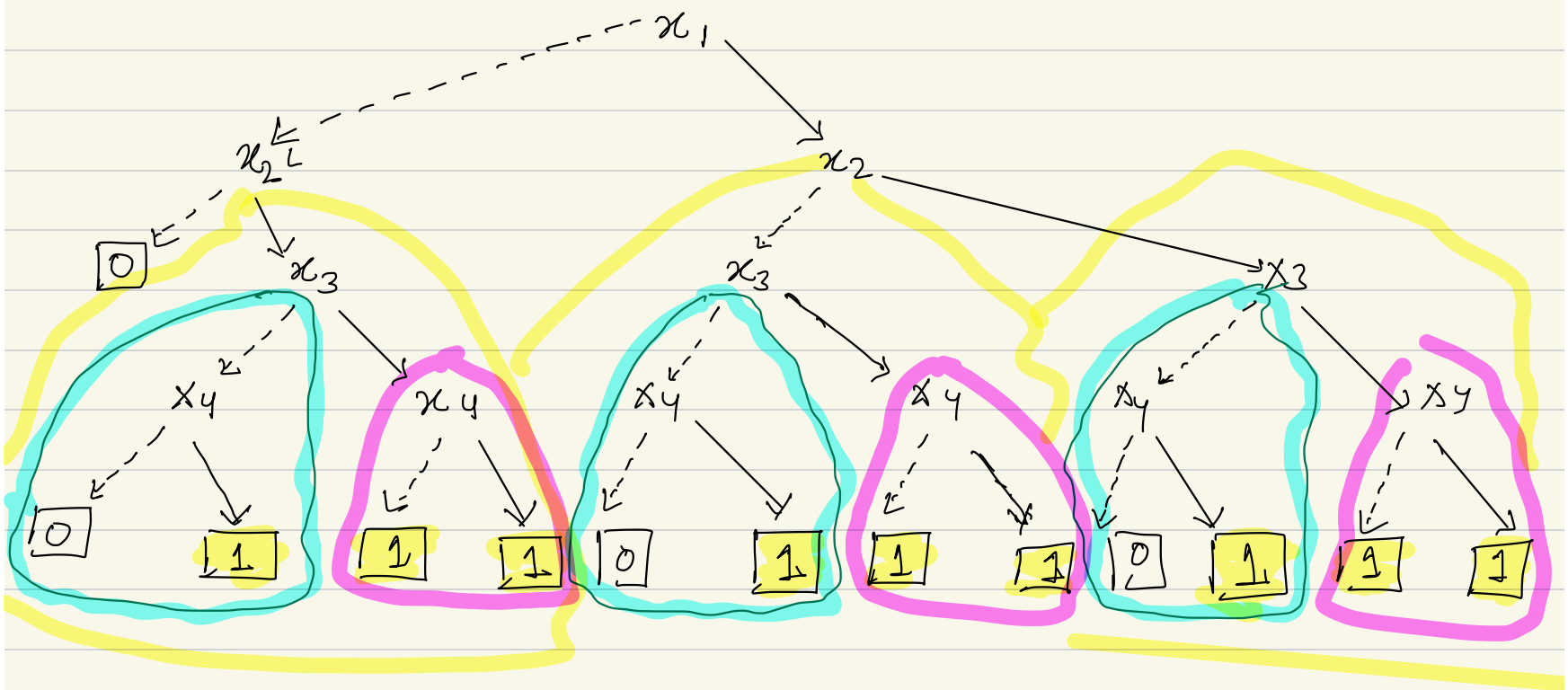
$$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$$

$|sol(F)| = 9$



$$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$$

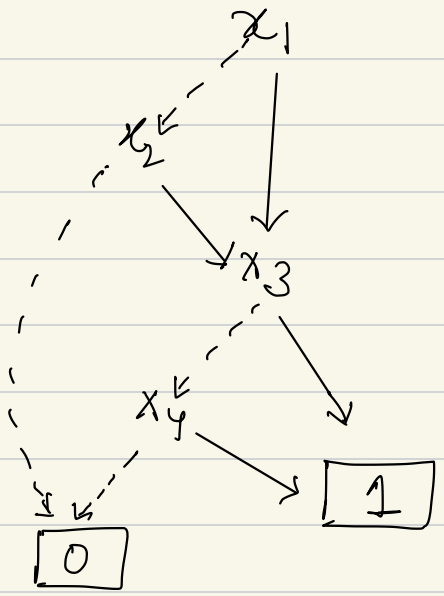
$|sol(F)| = 9$



RoBDD:

$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$

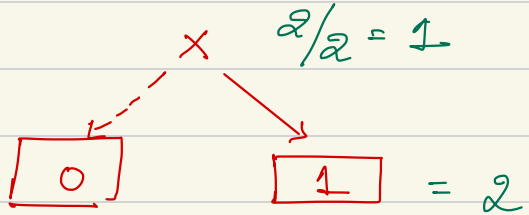
order $x_1 > x_2 > x_3 > x_4$



for every formula F, given a variable ordering, there is a unique RoBDD

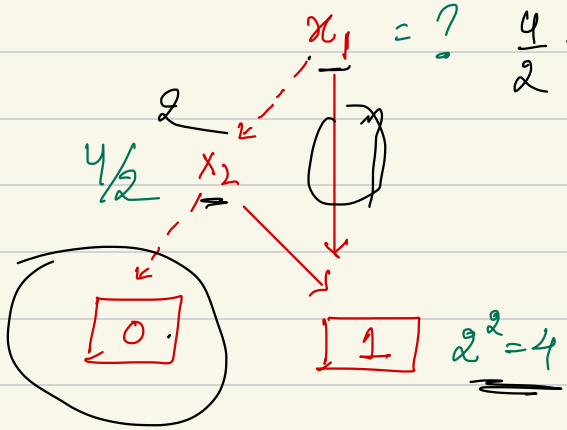
← what about model count?

$F = x$

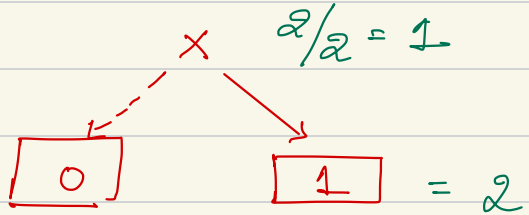


= we are fixing one particular variable as we move from child to parent node.

$F = x_1 \vee x_2$

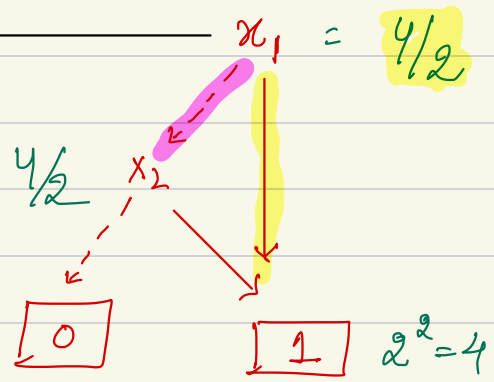


$F = x$



= we are fixing one particular variable as we move from child to parent node.

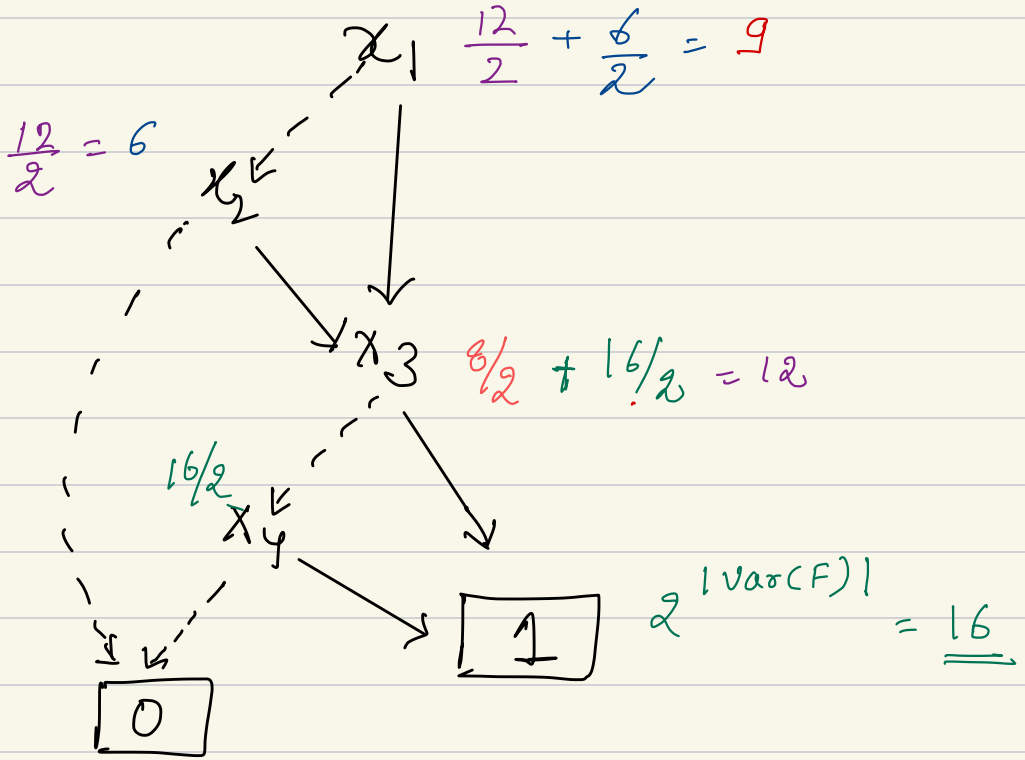
$F = x_1 \vee x_2$



= if there is an "intersection" b/w two paths, we will add the counts from each path.

When we reach the root node, we are done.

$|sol(F)|$ = count of root node.


x_1
$\frac{12}{2} + \frac{6}{2} = 9$
$\frac{12}{2} = 6$
x_2
x_3
$8/2 + 16/2 = 12$
$16/2$
x_4
1
$2^{|Var(F)|} = 16$
0

ROBDD vs CNF

	CNF	ROBDD
SAT	NP-HARD	Size of the tree $O(\|F_{ROBDD}\|)$
Model count	#P	$O(\|F_{ROBDD}\|)$
UNSAT	CO-NP-HARD	$O(1)$

ROBDD vs CNF

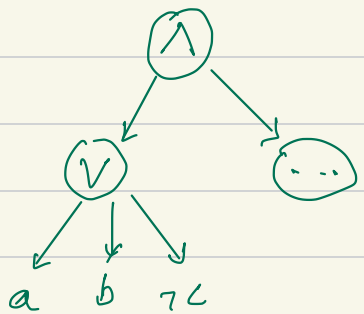
Building ROBDD takes exponential time in the worst case, as compared to polynomial time & space for CNF.

	CNF	ROBDD
SAT	NP-HARD	Size of the tree $O(\|F_{ROBDD}\|)$
Model count	#P	$O(\|F_{ROBDD}\|)$
UNSAT	CO-NP-HARD	$O(1)$

Different Compilation forms

NNF : Normal Negation form

- → each non-terminal node is either $\vee$ or $\wedge$
- → each terminal node is either a literal or 1 or 0.
- → each negation is push down to leaf node.



a NNF of a CNF

Darwiche (1998)

d-NNF (deterministic NNF)

A NNF is deterministic if for every V (OR) node with children $\{C_1 \dots C_k\}$ following holds:

$$\forall i \neq j \quad models(C_i) \cap models(C_j) = \phi$$

→ any two child of V node do not share models.

→ both paths of V node, can not occur together.

Darwiche (2011)

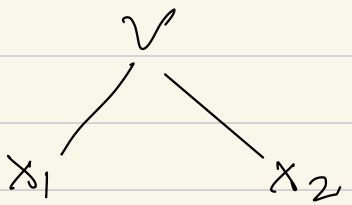
DNNF : (decomposable NNF)

A NNF is called DNNF if for every node $\wedge$ (and) with children $\{c_1 \cdots c_k\}$ following holds:

$$\forall i \neq j \quad vars(c_i) \cap vars(c_j) = \phi$$

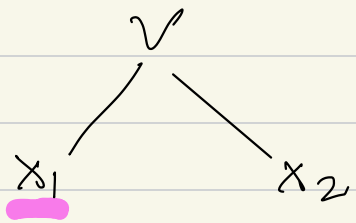
→ children don't share any literals.

$F = x_1 \vee x_2$



is it in d-DNNF ?

$F = x_1 \vee x_2$



is it in d-DNNF ?

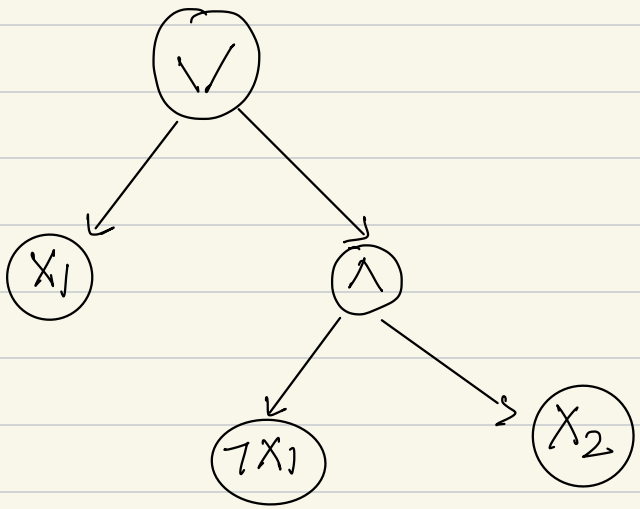
NO : Models of left node = $\{(x_1, \neg x_2), (x_1, x_2)\}$

Models of Right node = $\{(x_1, x_2), (\neg x_1, x_2)\}$

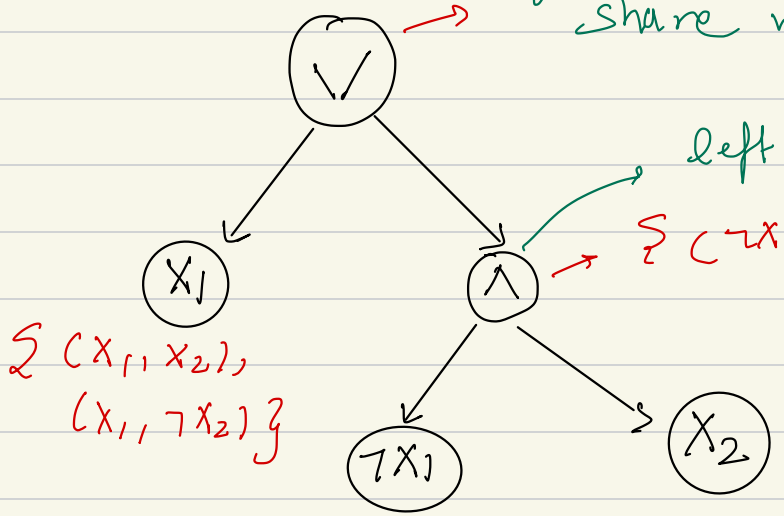
Left & Right node share a model, Hence NOT in d-DNNF.

$F = x_1 \vee x_2$

How about now?



$F = x_1 \vee x_2$



left & Right children doesn't share models.

left & Right children doesn't share variable

How about now?

Yes

$\{(x_1, x_2), (x_1, \neg x_2)\}$

$\{(\neg x_1, x_2)\}$

$\{(\neg x_1, x_2), (\neg x_1, \neg x_2)\}$

$\{(x_1, x_2), (\neg x_1, x_2)\}$

Model counting in d-DNNF

leaf node

- node is 0 → model count is 0
- node is 1 → model count is $2^{|vars(F)|}$
- node is a literal → model count is $2^{|vars(F)-1|}$.

Model counting for $\vee$ node in d-DNNF

* $\vee$ node is deterministic; children don't share models,

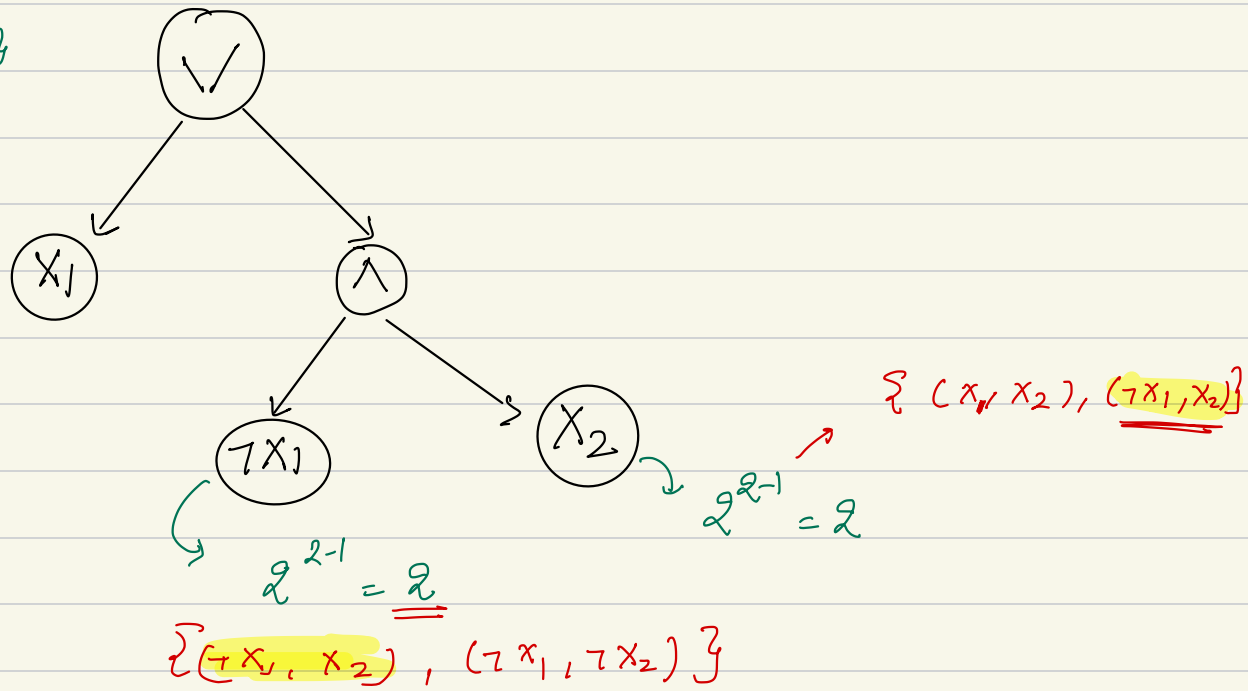
$$\text{Model count}(\vee) = \sum_{v \in \text{children}(\vee)} \text{Model count}(v)$$

↳ Sum of the # of models of its children.

Model counting for $\wedge$ node in d-DNNF

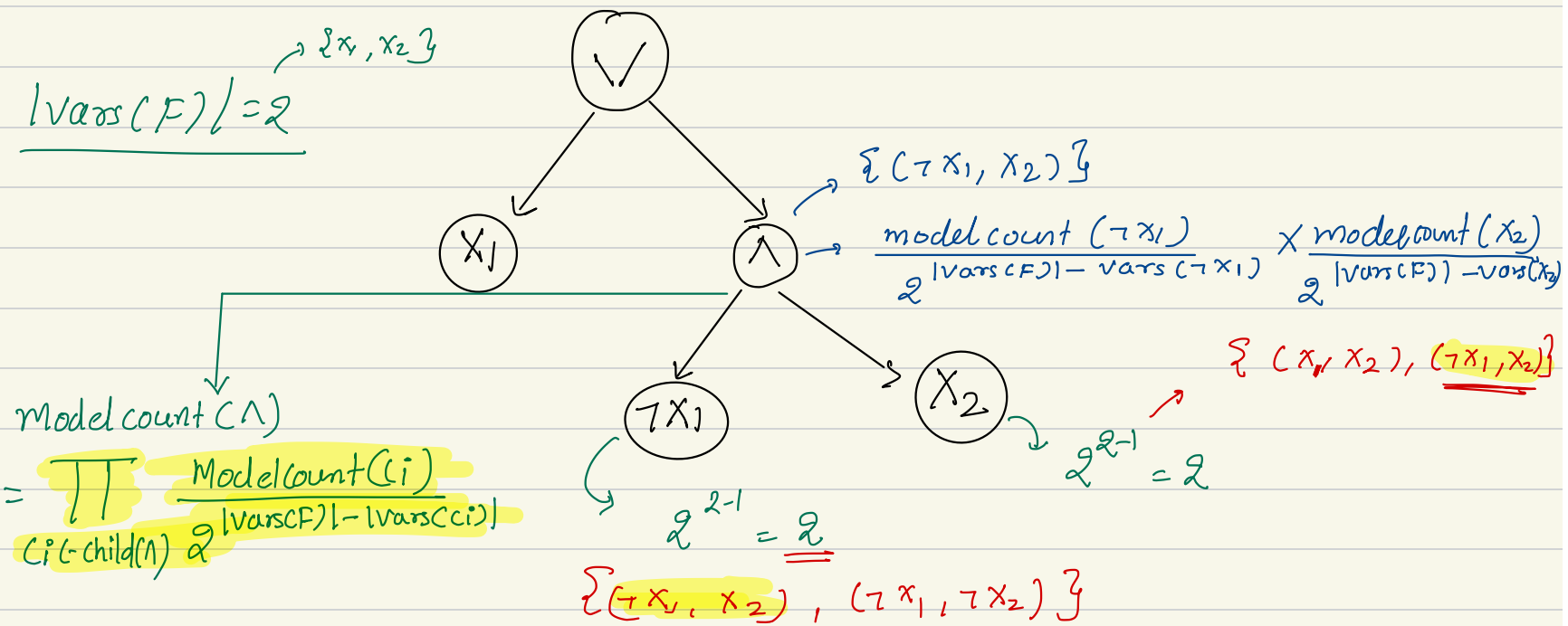
Let us understand it via an example:

$|vars(F)| = 2$ → $\{x_1, x_2\}$



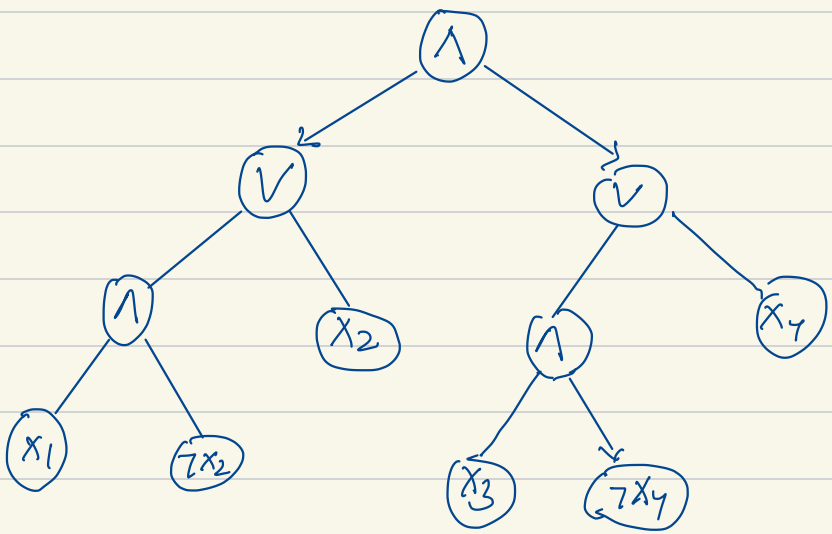
Model counting for $\wedge$ node in d-DNNF

Let us understand it via an example:



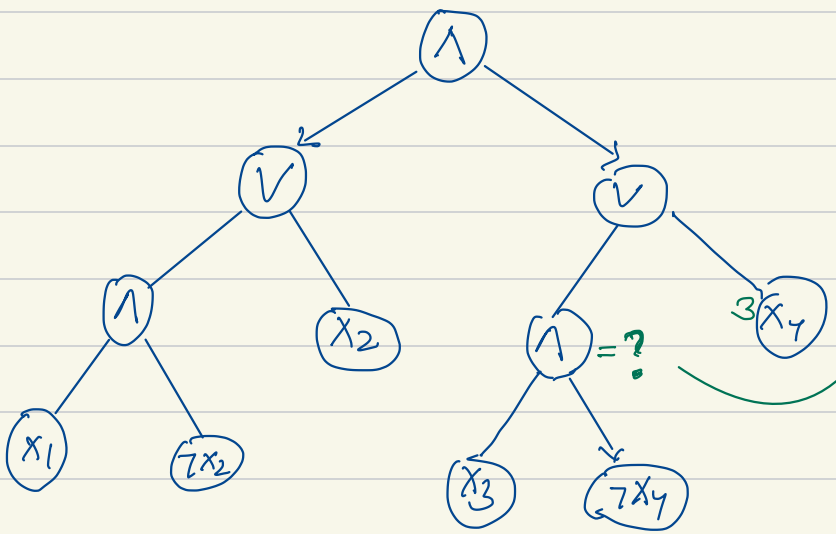
$(x_1 \vee x_2) \wedge (x_3 \vee x_4)$

↳ d-DNNF



$(X_1 \vee X_2) \wedge (X_3 \vee X_4)$

↳ d-DNNF



$1 \times 2^3 = 8$ $\quad$ $1 \times 2 = 8$

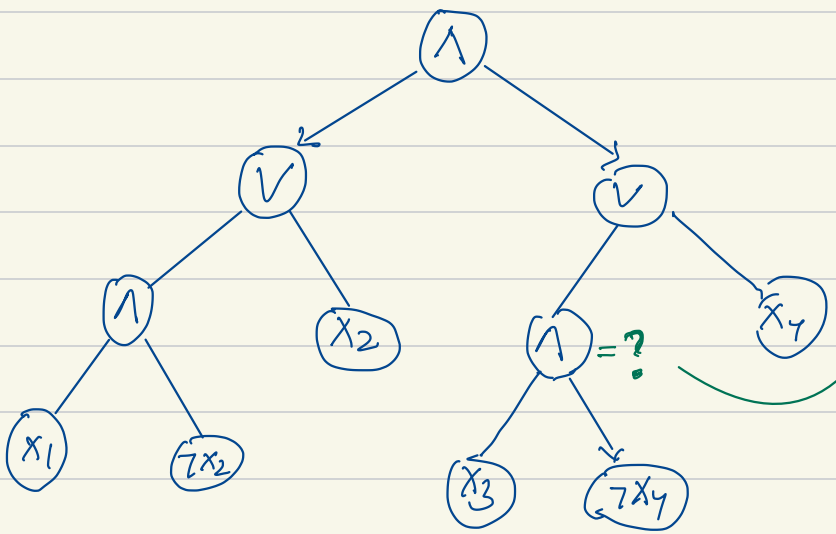
$$\prod \frac{\text{Model count}(C_i)}{2^{|\text{vars}(F)| - |\text{vars}\,C|}}$$

$$\frac{8}{2^{4-1}} \times \frac{8}{2^{4-1}} = 1$$

But what about X_1, X_2?

$(X_1 \vee X_2) \wedge (X_3 \vee X_4)$

↳ d-DNNF



$$\prod \frac{\text{Model count}(C_i)}{2^{|\text{vars}(F)| - |\text{vars}(C_i)|}}$$

$$\frac{8}{2^{4-1}} \times \frac{8}{2^{4-1}} = 1$$

But what about X_1, X_2?

$$\frac{8}{2^{4-1}} \times \frac{8}{2^{4-1}} \times 2^2 = 4$$

$$2^{|\text{vars}(F)| - \sum_{C_i \in \text{child}(\wedge)} |\text{vars}(C_i)|}$$

Model counting for formula in d-DNNF.

leaf

$$\forall u \in \{leaf(F) \setminus \{0,1\}\} : \text{ModelCount}(u) = 2^{|vars(F)|-1}$$

if 0 is $leaf(F)$: $\text{Model count}(0) = 0$

if 1 is $leaf(F)$: $\text{Model count}(1) = 2^{|vars(F)|}$

OR

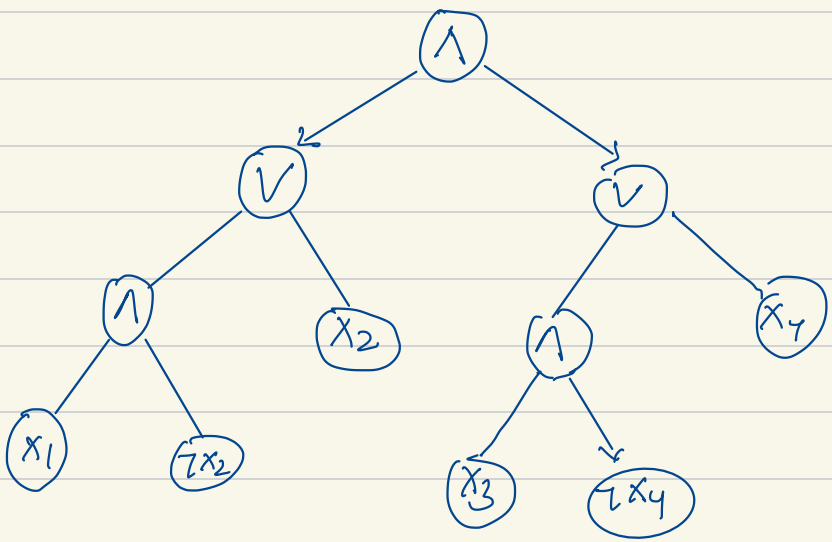
$$\text{Model count}(\vee) = \sum_{u \in child(\vee)} \text{model count}(u)$$

AND

$$\text{Model count}(\wedge) = \left(\prod_{u \in child(\wedge)} \frac{\text{Model count}(u)}{2^{|vars(F)| - |vars(u)|}} \right) \times 2^{|vars(F)| - \left| \bigcup_{u \in child(\wedge)} vars(u) \right|}$$

$(X_1 \vee X_2) \wedge (X_3 \vee X_4)$

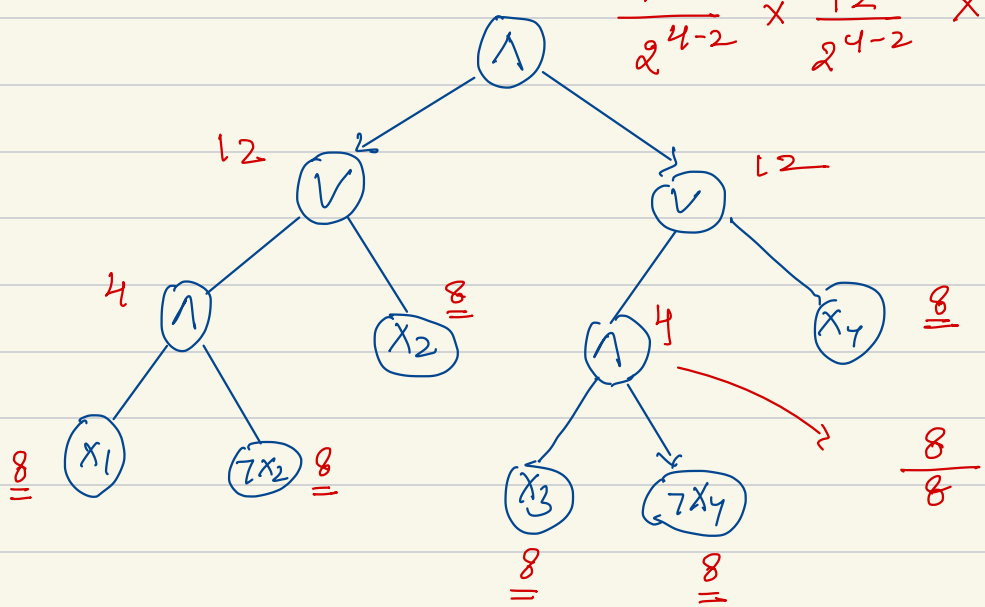
↳ d-DNNF



Compute Modelcount of F.

$(X_1 \vee X_2) \wedge (X_3 \vee X_4)$

↳ d-DNNF



$\frac{12}{2^{4-2}} \times \frac{12}{2^{4-2}} \times 2^{4-4} = 9$

$\frac{8}{8} \times \frac{8}{8} \times 2^2 = 4$

Model count (F) = 9

Weighted Model Counting :

$$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$$

$w(x_1) = 0.5$ $\quad w(\neg x_1) = 0.5$

$w(x_2) = 0.5$ $\quad w(\neg x_2) = 0.5$

$w(x_3) = 0.5$ $\quad w(\neg x_3) = 0.5$

$w(x_4) = 0.5$ $\quad w(\neg x_4) = 0.5$

$$\text{WMC}(F) = \sum_{\sigma \models F} \text{WMC}(\sigma)$$

Weighted Model Count

$$\text{WMC}(\sigma) = \prod_{x_i \in \sigma} w(x_i) \cdot \prod_{\neg x_i \in \sigma} w(\neg x_i)$$

$$= \sum (0.5 \times 0.5 \times 0.5 \times 0.5) + \cdots + (0.5) \times (0.5 \times 0.5 \times 0.5)$$

$$= 9 \times (0.5 \times 0.5 \times 0.5 \times 0.5)$$

$$= 0.5625$$

Weighted Model Counting :

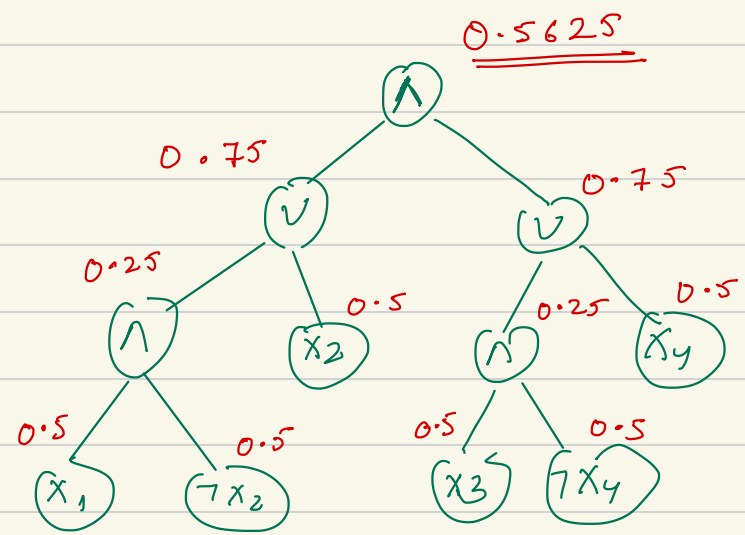
$$F = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$$

$w(x_1) = 0.5$ $\quad w(\neg x_1) = 0.5$

$w(x_2) = 0.5$ $\quad w(\neg x_2) = 0.5$

$w(x_3) = 0.5$ $\quad w(\neg x_3) = 0.5$

$w(x_4) = 0.5$ $\quad w(\neg x_4) = 0.5$



leaf $\begin{cases} \forall u \in \{leaf(F) \setminus \{0,1\}\} : \text{Modelcount}(u) = \text{weight}(u) \\ \text{if } 0 \text{ is leaf}(F) : \text{Modelcount}(0) = 0 \\ \text{if } 1 \text{ is leaf}(F) = \text{Modelcount}(1) = 1 \end{cases}$

OR $\begin{cases} \text{Modelcount}(\vee) = \sum_{u \in \text{child}(\vee)} \text{modelcount}(u) \end{cases}$

AND $\begin{cases} \text{Modelcount}(\wedge) = \left(\prod_{u \in \text{child}(\wedge)} \text{Modelcount}(u) \right) \end{cases}$

Approximate Model Counting:

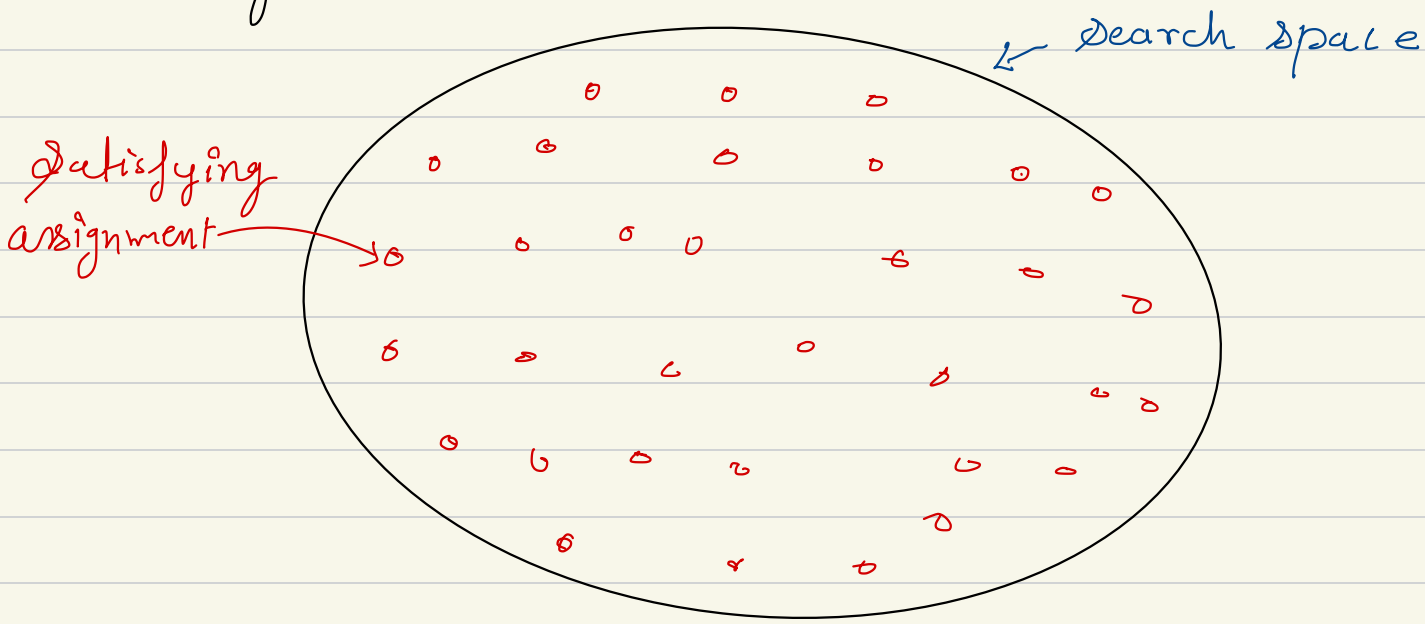
$$\Pr\left[\frac{|\text{Model Count}(F)|}{1+\epsilon} \leq c \leq |\text{Model count}(F)|(1+\epsilon)\right] \geq 1-\delta$$

PAC
probabilistic
Approximately
correct

→ We return c. with δ confidence, we can say that c is within ϵ error to the actual model count of formula c.

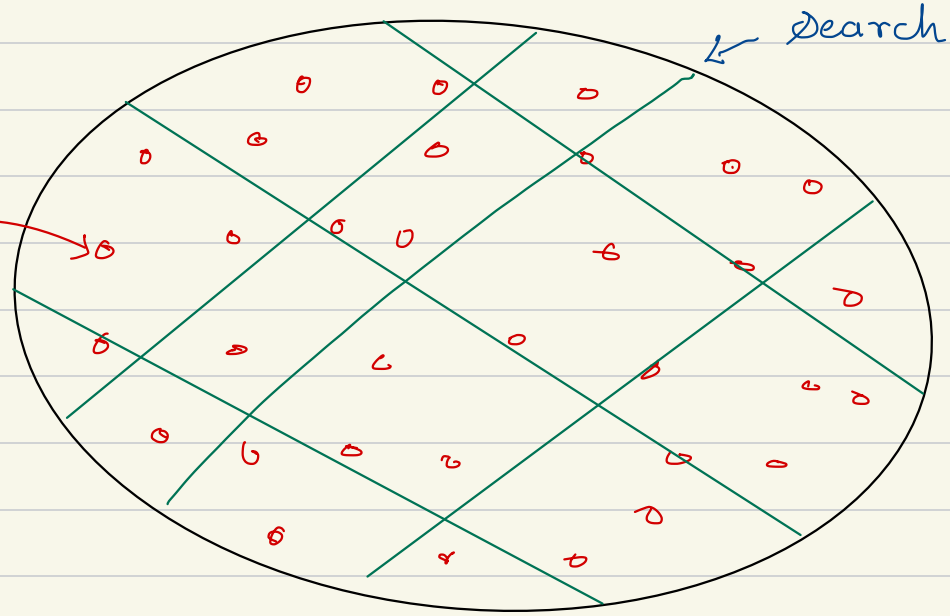
Usually, we have very high δ, & very very small ϵ.

Counting :



Task is to count the red dots.

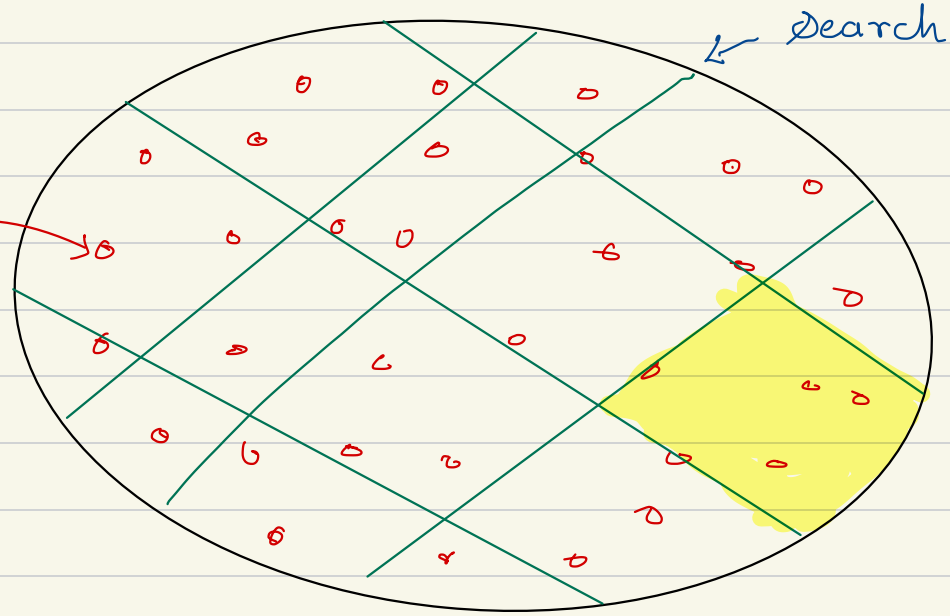
Counting :



1. partition the search space.

Task is to count the red dots.

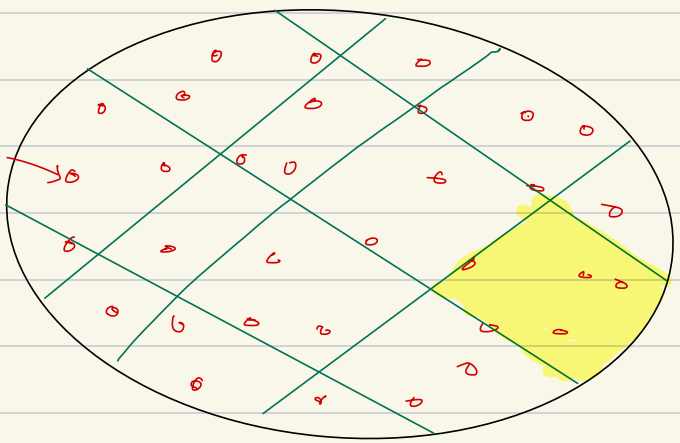
Counting :



Task is to count the red dots.

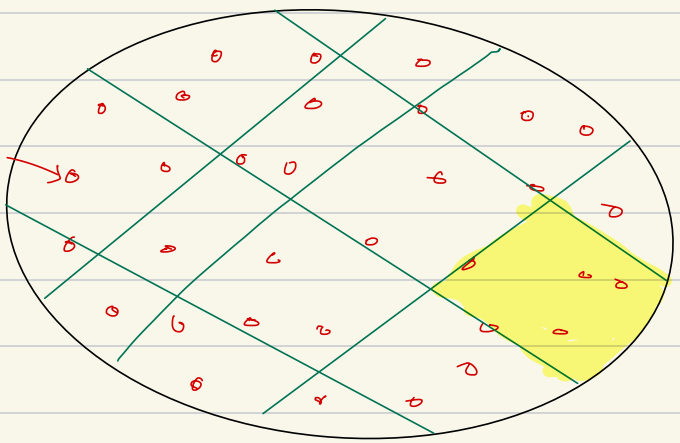
1. partition the search space.
2. pick a random cell.
3. count Red dots in cell.
4. Total Red dots = #cells X dots in one cell.

Counting :



1. partition the search cell
   - → small enough to count the satisfying assignment within the cell. ← using SAT solvers enumerate. solutions
   - → roughly equal # of solutions in each cell.
2. pick a random cell. count soln within cell
3. Total solutions = # cells x # soln within a cell.

Counting :



1. partition the search cell
   - → small enough to count the satisfying assignment within the cell. ← using SAT solvers enumerate. solutions
   - → roughly equal # of solutions in each cell.

Challenges

1. How to partition in small, roughly equal # soln many cells?
2. How many cells?

2. pick a random cell. count soln within cell
3. Total solutions = # Cells X # soln within a cell.

1. How many cells?

↳ we will fix a threshold, and each cell should have $\cong$ threshold many solutions.

$$\#\text{ of cells} = \frac{\text{Model count (F)}}{\text{threshold}}$$

We want to partition into 2^m cells such that $2^m \cong \frac{\text{Model count (F)}}{\text{threshold}}$.

→ we will start with $m = 20, 1, 2, \ldots >$ until # of soln in a cell is less than threshold.

1. How many cells?

↳ we will fix a threshold, and each cell should have $\cong$ threshold many solutions.

$$\#\text{ of cells} = \frac{\text{Model count}(F)}{\text{threshold}}$$

We want to partition into 2^m cells such that $2^m \cong \frac{\text{Model count}(F)}{\text{threshold}}$.

1. Query Model count $(F \wedge Q_1) \lesssim$ threshold, if No
2. Query Model count $(F \wedge Q_1 \wedge Q_2) \lesssim$ threshold, if No
3. Query Model count $(F \wedge Q_1 \wedge Q_2 \cdots Q_m) \lesssim$ threshold, if yes

return Model count $(F \wedge Q_1 \wedge Q_2 \wedge \cdots \wedge Q_m) \times 2^m$

2. So, what is this Q: which is partitioning search space?

Or How are we partitioning search space

↳ hashing (h)

assignments → cell (hashing)

$\{0,1\}^n \rightarrow \{0,1\}^m$

2-wise independent hash function

→ $x_1 \dots x_n$ variables of formula F.

→ $h: \{0,1\}^n \rightarrow \{0,1\}^m$

→ { Create XORs: Choose each x_i with prob $1/2$ & XOR them

→ $x_1 \oplus x_3 \oplus \dots \oplus x_n$

↳ Choose "m" random XORs.

ApproxMC (Active Area, ∵ every year a new version comes)

Key points

1. CNF + XORs formulas to partition search space into cells.
2. "m" cells, m many random XOR.
3. Choose value of m such that Model count of a randomly chosen cell is less than predefined threshold.
4. return m X model count of a cell.

——— X ——— X