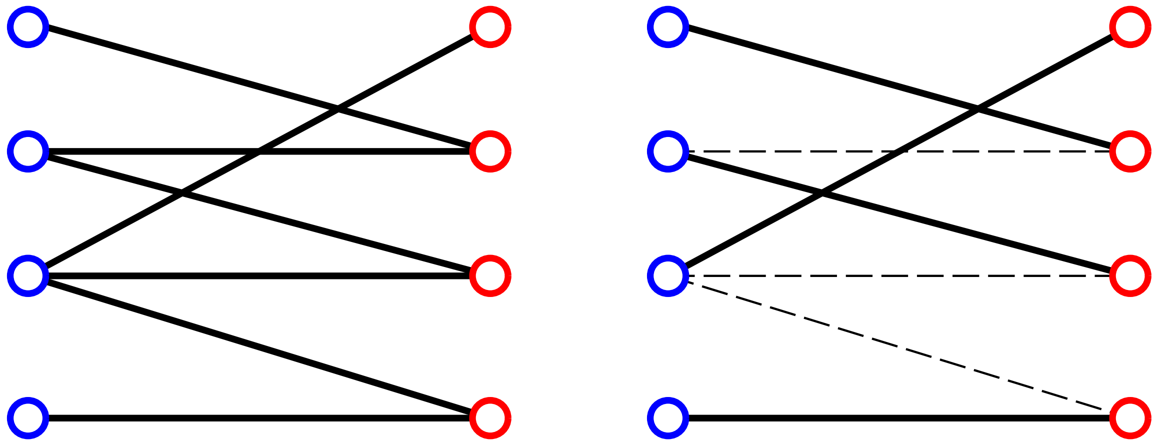
Randomized Computation

Sometimes we can use *randomization* to solve problems that are difficult to solve deterministically.

Examples:

- symbolic determinants
- 2SAT
- primality testing

Bipartite Matching



Bipartite Matching

Given

- graph (U, V, E)
- $U = \{u_1, \ldots, u_n\}$
- $V = \{v_1, \ldots, v_n\}$
- $E \subseteq U \times V$

Find a *perfect matching*

- $M \subseteq E$
- every vertex in U on an edge in M
- for any two edges (u, v) and (u', v'), $u \neq u'$ and $v \neq v'$

Alternatively:

- a permutation π of $\{1, \ldots, n\}$ such that $(u_i, v_{\pi(i)}) \in M$.

Randomized Computation

We can represent any bipartite graph G

- as an $n \times n$ *symbolic* matrix A^G
- (i, j)-th element is variable a_{ij} if $(u_i, v_j) \in E$
- and 0 otherwise.

Note that a_{ij} is an uninterpreted *symbol*!

Randomized Computation

Recall the *determinant* of a matrix:

$$\mathit{det} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

$$\mathit{det}\ A^G = \sum_{\pi} \sigma(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}$$

Where

- π ranges over all permutations
- σ is 1 if π is product of even number of transpositions
- σ is -1 otherwise

Randomized Computation

$$\mathit{det}\ A^G = \sum_{\pi} \sigma(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}$$

Claim: G has a perfect matching if and only if $\mathit{det}\ A^G$ is not identically zero.

- only non-zero terms correspond to perfect matchings
- all non-zero terms distinct, so they can't cancel

Symbolic determinants useful for variety of other purposes.

Randomized Computation

We can compute determinants by *Gaussian Elimination*

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 1 & 7 & -2 & 4 \\ -1 & -3 & -2 & 2 \\ 0 & 1 & 6 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 0 & 7 \\ 0 & 1 & 6 & 2 \end{pmatrix}$$

- subtract multiples of first row from $2, \ldots, n$
- to make first column element zero
- continue with subsequent rows . . .

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 0 & 7 \\ 0 & 1 & 6 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 0 & 7 \\ 0 & 0 & 6 & 2\frac{1}{4} \end{pmatrix}$$

Randomized Computation

If *pivot* element is zero:

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & \mathit{0} & 7 \\ 0 & 0 & 6 & 2\frac{1}{4} \end{pmatrix}$$

Then

- transpose rows
- multiply determinant by -1
- if this is impossible, determinant is zero.

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 6 & 2\frac{1}{4} \\ 0 & 0 & 0 & 7 \end{pmatrix}$$

Randomized Computation

At the end, matrix has *upper triangular form*:

$$\begin{pmatrix} 1 & 3 & 2 & 5 \\ 0 & 4 & -4 & -1 \\ 0 & 0 & 6 & 2\frac{1}{4} \\ 0 & 0 & 0 & 7 \end{pmatrix}$$

Determinant is

- product of diagonal elements: 196
- adjusted for row interchanges: -196

Randomized Computation

Claim: We can calculate the determinant of an $n \times n$ matrix in polynomial time.

Must check

- if entries are integers of b bits,
- result not exponentially long in b

This isn't difficult.

Randomized Computation

What about symbolic matrices?

$$\begin{pmatrix} x & w & z \\ z & x & w \\ y & z & w \end{pmatrix} \Rightarrow \begin{pmatrix} x & w & z \\ 0 & \frac{x^2-zw}{x} & \frac{wx-z^2}{x} \\ 0 & \frac{zx-wy}{x} & -\frac{zy}{x} \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} x & w & z \\ 0 & \frac{x^2-zw}{x} & \frac{wx-z^2}{x} \\ 0 & 0 & -\frac{yz(xz-xw)+(zx-wy)(wx-x^2)}{x(x^2-xw)} \end{pmatrix}$$

Is there a problem here?

Randomized Computation

We have a problem.

Applying Gaussian elimination to symbolic matrices works, but

- intermediate terms are rational functions
- it can be shown that they become exponentially large!

Uh, oh.

Randomized Computation

Observation:

- we don't need to compute the entire matrix
- we care only if it is identically zero

Idea:

- substitute arbitrary integers for the variables
- do poly-time numerical Gaussian elimination

Randomized Computation

Idea:

- substitute arbitrary integers for the variables
- do poly-time numerical Gaussian elimination

Interpretation:

- if the result is non-zero, we know there is a matching!
- but what if the result is zero?

Randomized Computation

Lemma Let $\pi(x_1, \ldots, x_m)$ be a polynomial

- not identically zero
- in m variables
- of degree at most d
- and let $M > 0$ be an integer.

Then

- the number of m-tuples $(x_1, \ldots, x_m) \in \{0, 1, \ldots, M\}$ such that
- $\pi(x_1, \ldots, x_m) = 0$
- is at most mdM^{m-1}

Randomized Computation

Let $A^G(x_1, \ldots, x_m)$ be the matrix A^G with its m variables.

Notice that *det* $A^G(x_1, \ldots, x_m)$ has degree 1.

Randomized Algorithm

- choose m random integers $i_1, \ldots, i_m$
  - between 0
  - and $M = 2m$
- compute *det* $A^G(i_1, \ldots, i_m)$ by Gaussian elimination.
  - if result non-zero, then *accept*
  - otherwise if result is zero, then *reject*

Randomized Computation

Claim If this algorithm

- accepts, then G has a matching.
- rejects, then G *probably* has no matching.

Probably?

- how may roots does *det* $A^G(x_1, \ldots, x_m)$ have?
- by the earlier lemma, substituting $d = 1$ and $M = 2m$,
  - there are at most $mdM^{m-1} = M^m/2$ roots
  - among the possible M^m random choices

So the likelihood of getting a *false negative* is $\leq \frac{1}{2}$

Randomized Computation

Repeating the experiment improves our chances.

After k experiments

- If even one non-zero, matching exists.
- If all zero, no matching with probability $1 - (\frac{1}{2})^k$

After 100 negatives, it is equally likely that

- the Turing machine will be destroyed by a meteorite
- it will fail to find a matching

Randomized Computation

This is a **Monte Carlo** algorithm.

- runs efficiently
- very small likelihood of error

A **Las Vegas** algorithm.

- no error error
- very small likelihood of long running time

Random Walk Algorithms

Here is a Monte Carlo algorithm for satisfiability:

M: on input $\langle\phi\rangle$ where ϕ is a CNF formula:

- Repeat r times:
  - If all clauses satisfied, *accept*.
  - pick unsatisfied clause (all literals false)
  - pick literal at random
  - "flip" its value in truth assignment.

Random Walk Algorithms

Claim: If we find a find a satisfying assignment, the formula is satisfiable.

Claim: If we fail to find a satisfiable assignment after "enough" repetitions, the formula is "probably" not satisfiable.

Question: Is a polynomial number of repetitions sufficient to get probability $\frac{1}{2}$?

Random Walk Algorithms

Let's try an easy case first.

Claim: For 2SAT, $r = 2n^2$ repetitions are enough to find a satisfying assignment with probability $\geq \frac{1}{2}$.

Random Walk Algorithms

Let

- $\hat{T}$ be the satisfying assignment
- T the current assignment
- the *distance* i between $\hat{T}$ and T is the the number of variable assignments in which they differ
- let $t(i)$ be the expected time to reach $\hat{T}$ from T.

Random Walk Algorithms

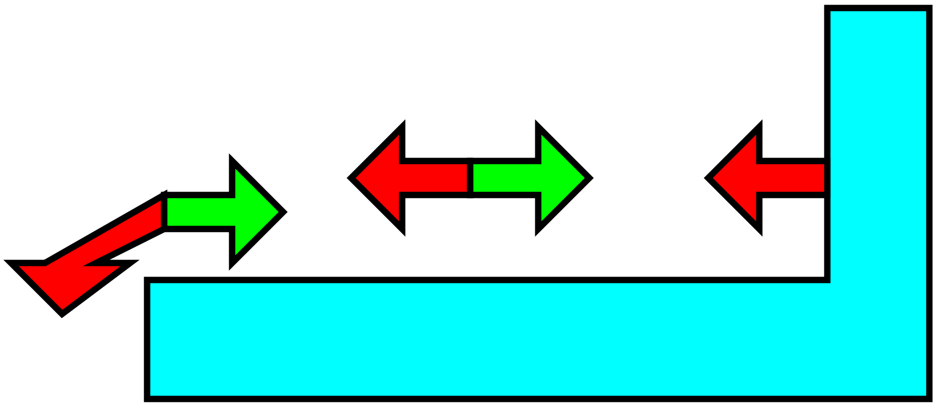
In any unsatisfied clause

- both literals are false
- at least one literal is true in $\hat{T}$

Random choice takes T

- closer to $\hat{T}$ with probability $\geq \frac{1}{2}$
- farther with probability $\leq \frac{1}{2}$

Random Walk Algorithms



Random walk

- one absorbing barrier
- one reflecting barrier

Random Walk Algorithms

$$\begin{aligned} t(i) &= \frac{t(i-1)}{2} + \frac{t(i+1)}{2} + 1 \\ t(0) &= 0 \\ t(n) &= t(n-1) + 1 \end{aligned}$$

Let's compute $t(i)$.

Random Walk Algorithms

$$
\begin{aligned}
t(i) &= \frac{t(i-1)}{2} + \frac{t(i+1)}{2} + 1 \\
\sum_{i=1}^{n-1} t(i) &= \frac{1}{2}\left(\sum_{i=1}^{n-1} t(i-1) + \sum_{i=1}^{n-1} t(i+1)\right) + n - 1 \\
&= \frac{1}{2}\left(\sum_{i=0}^{n-2} t(i) + \sum_{i=2}^{n} t(i)\right) + n - 1 \\
&= \sum_{i=2}^{n-2} t(i) + \frac{1}{2}(t(0) + t(n-1) + t(n)) + n \\
t(1) + t(n-1) &= \frac{1}{2}(t(1) + t(n-1) + t(n)) + n - 1
\end{aligned}
$$

Random Walk Algorithms

Recall that

$$\begin{aligned} t(n) &= t(n-1)+1 \\ t(n-1) &= t(n)-1 \end{aligned}$$

Subtract from:

$$t(1)+t(n-1) = \frac{1}{2}(t(1)+t(n-1)+t(n))+n-1$$

to get

$$t(1) = 2n-1$$

Random Walk Algorithms

From these two equations

$$
\begin{aligned}
t(1) &= 2n - 1 \\
t(1) &= \frac{t(2)}{2} + 1
\end{aligned}
$$

we have that

$$t(2) = 4n - 4$$

In general

$$t(i) = 2in - i^2$$

so

$$t(n) = n^2$$

Even if you guess completely wrong, the *expected time* to find the satisfying assignment is n^2 steps.

Random Walk Algorithms

What about probabilities?

Lemma: Let x be a random non-negative integer variable with expected value $E(x)$.

$$\text{prob}[x \geq k \cdot E(x)] \leq \frac{1}{k}$$

Corollary: If x is the number of steps until the satisfying assignment is discovered, then $2n^2$ steps will discover with probability $\geq \frac{1}{2}$.

Random Walk Algorithms

What about 3CNF?

Consider

$$(x_1) \wedge \ldots \wedge (x_n)$$

only satisfied by all-true assignment.

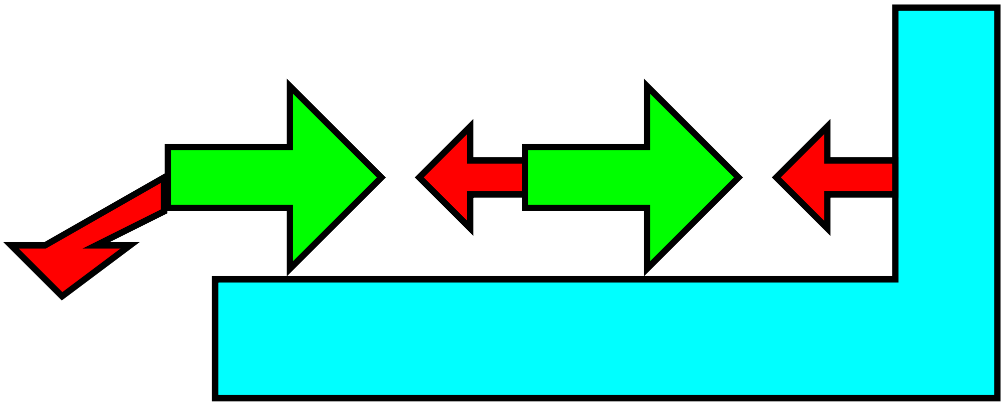
Replace each (x_i) with $(x_i \vee \overline{x_j} \vee \overline{x_k})$, for all distinct i, j, k.

Equivalent formula.

Random Walk Algorithms

Consider random walk algorithm:

- chooses correct literal x_i with probability $\frac{1}{3}$
- chooses wrong literal $\overline{x_j}, \overline{x_k}$ with probability $\frac{2}{3}$



This takes exponential time!