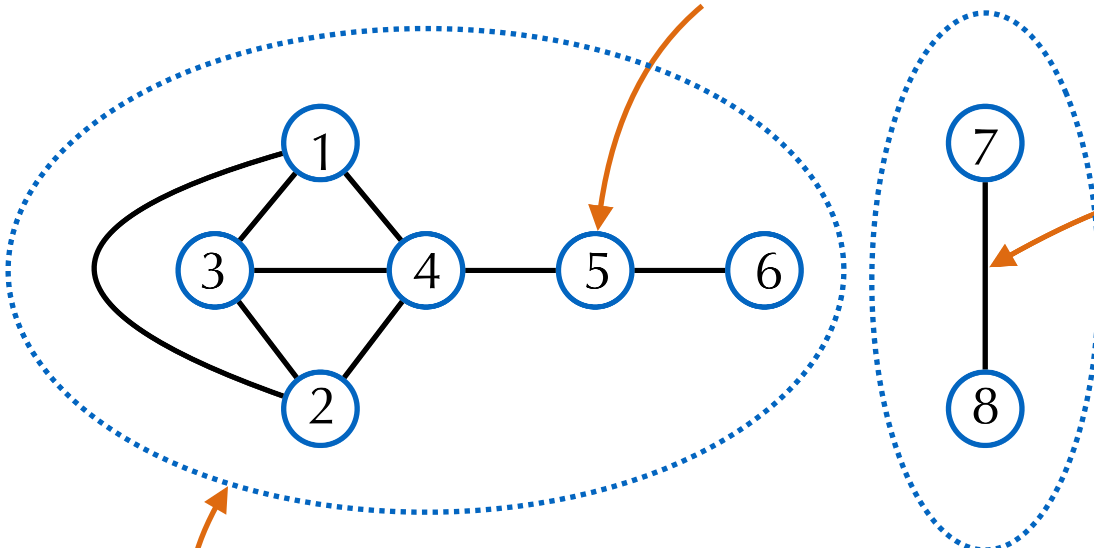
CSE202
Design and Analysis of Algorithms

Week 8 — Randomized Algorithms 3: Random Search

I. Random Walk in a Maze

Recall Graph Vocabulary (CSE102)

vertex of degree $d(v) = 2$



edge

connected components

Finite Graph

n vertices $\in \mathbb{N}$

m edges

$$m \leq \binom{n}{2}$$

Adjacency matrix

$$A(G) := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$A(G)_{ij} = 1$: edge $(i, j) \in G$

G undirected : $A(G)$ symmetric.

Distance $\Delta(u, v)$: minimal number of edges in a path from u to v.

Probabilistic Algorithm

Input: u *initial vertex,* v *target vertex*

While $u \neq v$

  Pick a neighbor w of u uniformly at random

  Set $u := w$

Return

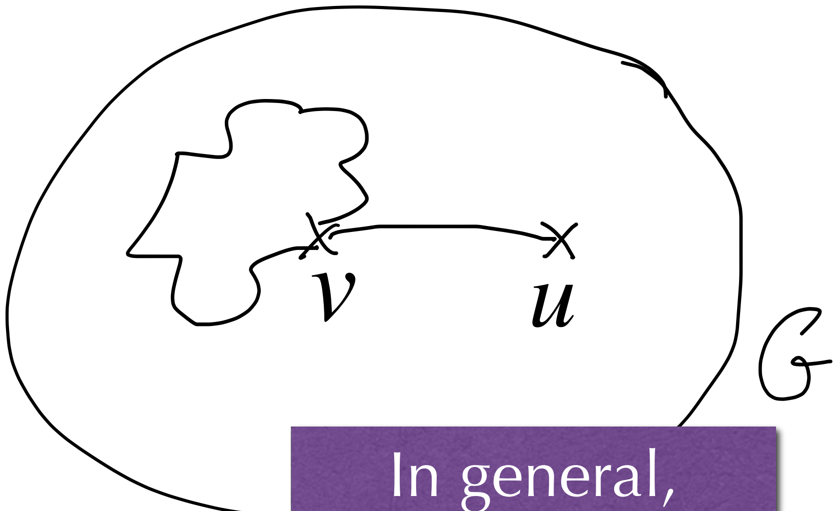
Memory: $O(\log n)$

Random variable X_k = vertex visited at kth step $(X_0 = u)$.

Complexity: $T(u, v) := \mathbb{E}(\inf\{k \geq 1 \mid X_k = v\}) = ??$

turns out to be polynomial in n.

Exiting the Maze



In general,
$T(u,v) \neq T(v,u)$

Lemma. $$\sum_{v|(u,v)\in G} T(v,u) = 2m - d(u)\,.$$

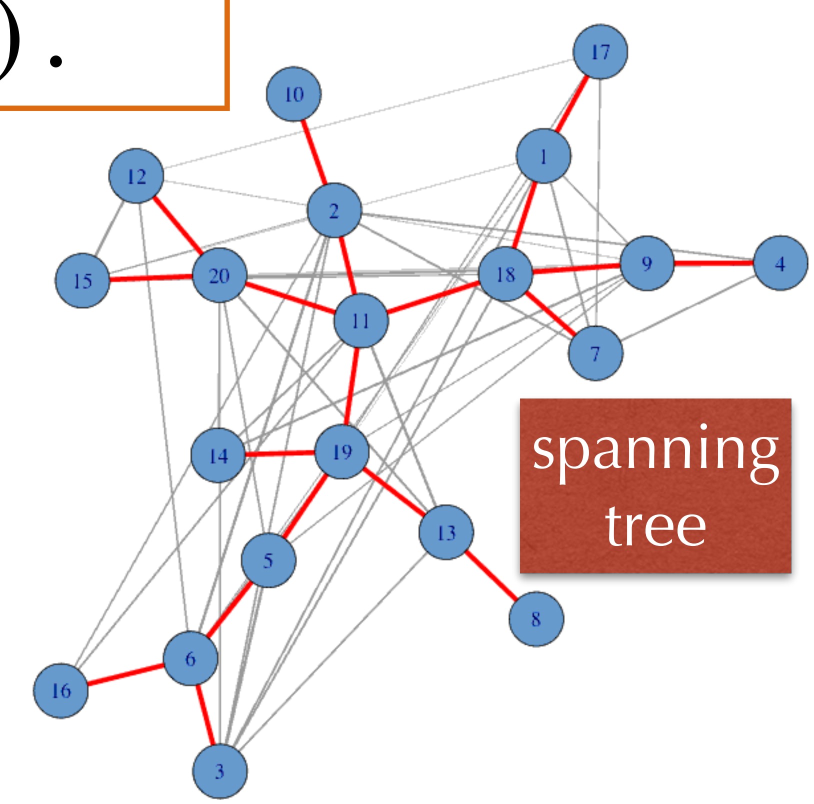
Proof next page

$\Rightarrow$ for any edge (u,v), $T(u,v) \leq 2m - 1$.

Prop1. For arbitrary vertices u, v,
$$T(u,v) \leq (2m-1)\Delta(u,v)\,.$$

Prop2. Expected time to visit all nodes:
$$T(u,\cdot\,) \leq 2m(n-1)\,.$$

Proof by using a spanning tree and summing Lemma over u



spanning tree

Proof of the Lemma

Lemma. $\displaystyle\sum_{v|(u,v)\in G} T(v,u) = 2m - d(u)\,.$

Notation $p_{uv} := \begin{cases} \frac{1}{d(u)} & \text{if } (u,v)\in G, \\ 0 & \text{otherwise.} \end{cases}$

Decompose by first step:

$$T(w,u) = p_{wu} + \sum_{\substack{v \,|\, (w,v)\in G \\ v\neq u}} \frac{1}{d(w)}(1+T(v,u)) = 1 + \frac{1}{d(w)} \sum_{v|(w,v)\in G} T(v,u) - p_{wu}T(u,u)$$

Multiply by $d(w)$ and sum over $w \in G$:

$$\begin{aligned} \sum_w d(w)T(w,u) &= \sum_w d(w) + \sum_w \sum_{v|(w,v)\in G} T(v,u) - \left(\sum_w d(w)p_{wu}\right) T(u,u) \\ &= 2m + \sum_v d(v)T(v,u) - d(u)T(u,u) \end{aligned}$$

$$\Longrightarrow T(u,u) = \frac{2m}{d(u)}$$

Specialize at $w = u$

$$\frac{2m}{d(u)} = 1 + \frac{1}{d(u)} \sum_{v|(u,v)\in G} T(v,u)\,.$$

Exiting the Maze

Recall **Prop2**. Expected time to visit all nodes:
$$T(u, \cdot\,) \leq 2m(n-1)\,.$$

Consequence (Markov's inequality):

$\mathbb{P}(v$ not visited in $4nm$ steps$) \leq 1/2.$

Boost by repeats.

Monte-Carlo algorithm in time $O(nm)$, memory $O(\log n)$.

Negative answer: not in the same connected component.

Comparison: depth first search uses $O(m)$ time *and* memory.

II. Satisfiability

The story of satisfiability is the tale of a triumph of software engineering, blended with rich doses of beautiful mathematics. D. Knuth

Boolean Formulas

Variables: $x_1, \ldots, x_n$ with values in $\{0,1\}(= \{\text{false}, \text{true}\})$.

Operations: negation ($\bar{x}$), or ($\vee$), and ($\wedge$) .

Ex.: $F := (x_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_1 \wedge \bar{x}_2)$
$G := (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_2 \vee x_3)$.

Exercise:
check $F \equiv G$.

Satisfiability: existence of an assignment s.t. F=1.

Ex.: $(x_1, x_2, x_3) = (0,0,1)$ satisfies F .

Checking such an assignment is linear in the size of the formula.

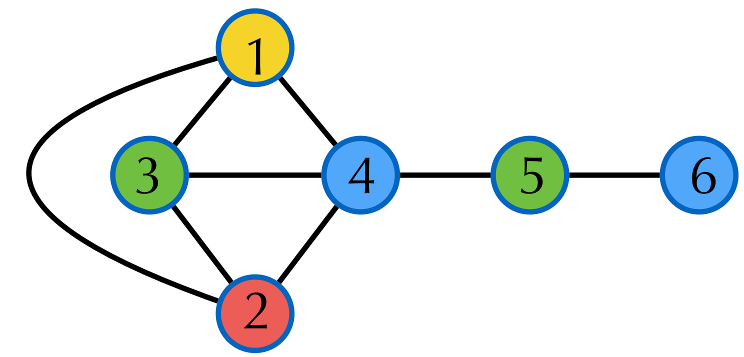
Clause: disjunction ($\vee$) of variables or their negations.

Conjunctive normal form: conjunction ($\wedge$) of clauses.

(G is in CNF.)

Example: Graph Coloring

Assign a color to all vertices so that every edge joins vertices of distinct colors.



One variable for each (vertex,color)

Four-color theorem (1976).
Every *planar* graph is 4-colorable.

(no purely human proof known)

One clause by vertex: $x_{i1} \vee x_{i2} \vee x_{i3} \vee x_{i4}$

Four clauses by edge: $\bar{x}_{i1} \vee \bar{x}_{j1}, \ldots, \bar{x}_{i4} \vee \bar{x}_{j4}$

Special case: Sudoku.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

k-SAT

Def. A CNF where every clause involves
at most k of the n variables.

Simple algorithm: try all 2^n assignments.

For $k \geq 3$, no polynomial-time algorithm is known.

In practice, modern SAT-solvers solve problems with
10,000 variables and millions of clauses.
Used in hardware or software checking, planning,…

One of the key algorithms is WalkSat.

$k > 3$ reduces to $k = 3$, using
$x_1 \vee x_2 \vee x_3 \vee x_4 \equiv (x_1 \vee x_2 \vee T_1) \wedge (\overline{T}_1 \vee x_3 \vee x_4)$,
with a new variable T_1 .

III. WalkSat

WalkSat

Input: a k-SAT formula F in n variables
Output: an assignment or FAIL

To be determined by the analysis.

1. Pick an assignment $B \in \{0,1\}^n$ uniformly at random.
2. Repeat N times:
   If the formula is satisfied by the assignment, return B.
   Choose a clause C not satisfied.
   Pick a variable x uniformly at random among C's.
   Update B by flipping x.
3. Return FAIL

If p_N is the probability of success, boost it by t/p_N repeats.

Exercise: with $t = 5$, $\mathbb{P}(\text{failure}) < 1\,\%$.

Example

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3)$$

1. Start with (0,1,0)

   $(x_1 \vee \overline{x}_2 \vee x_3)$ is not satisfied

2. Flip $x_1 \rightarrow (1,1,0)$

   $(\overline{x}_1 \vee \overline{x}_2 \vee x_3)$ is not satisfied

3. Flip $x_2 \rightarrow (1,0,0)$

   Solved!

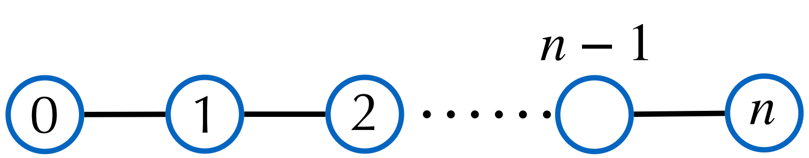
Analysis of Walksat when $k = 2$

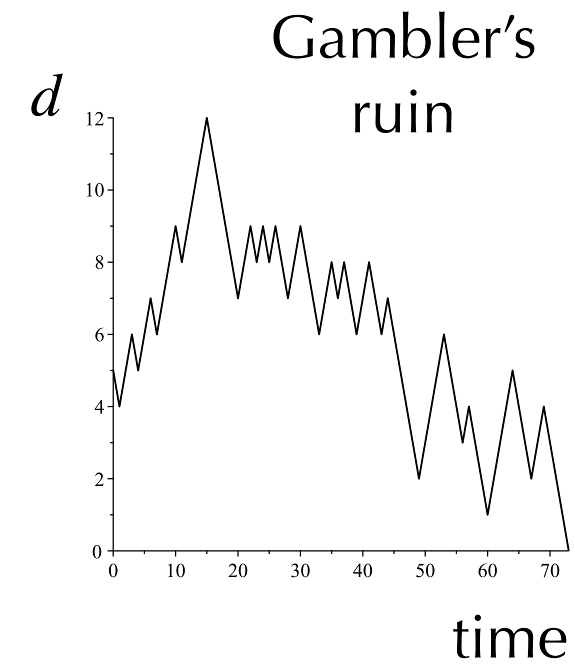
$$(\bar{x}_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\bar{x}_3 \vee x_4) \wedge \cdots$$

Assume the existence of a satisfying assignment A.
$d := \mathrm{dist}(A, B) =$ number of variables where $A \neq B$.

At each flip, $\Delta d = \pm 1$ and $\mathbb{P}(\Delta d = -1) \geq 1/2$.

Random walk on the graph





Expected number of steps $\leq 2nd_0 \leq 2n^2$.

$\boxed{\text{Stopping after } N = 4n^2 \text{ steps gives } \mathbb{P}(\text{success}) \geq 1/2.}$

WalkSat gives a Monte Carlo algorithm in time $O(n^2)$.

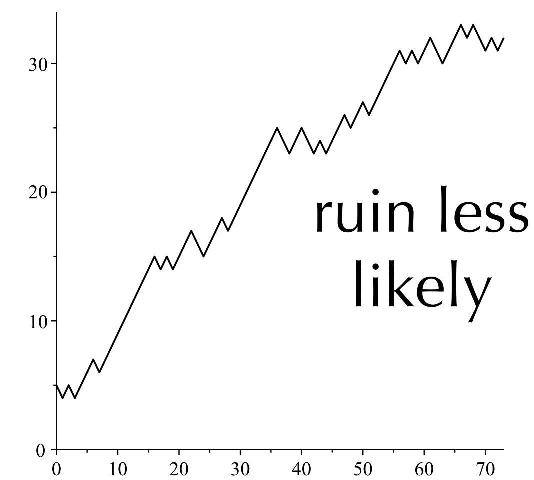
Analysis for Larger k

Same worst-case reasoning gives: $\mathbb{P}(\Delta d = -1) \geq 1/k$.

Proba $p(d)$ of reaching 0 starting from d when

Worst-case situation

$$\mathbb{P}(\Delta d = -1) = 1/k,\ \mathbb{P}(\Delta d = 1) = 1 - 1/k.$$



Lemma. $p(d) = (k-1)^{-d}$.

Proof on the blackboard

Proba WalkSat succeeds (with $N = \infty$):

$$\mathbb{P}(\text{success}) \geq 2^{-n} \sum_{d=0}^{n} \binom{n}{d} p(d) = \left(\frac{k}{2(k-1)}\right)^n.$$

When should it give up and restart?

Stopping after $3n$ Steps for 3-SAT

$3n$ steps also sufficient for $k > 3$, with a different proof.

$$\mathbb{P}(\text{success in } 3n \text{ steps starting from } d)$$

$$\geq \mathbb{P}(\text{success in } 3d \text{ steps starting from } d)$$

$$\geq \binom{3d}{d}\left(\frac{2}{3}\right)^{d}\left(\frac{1}{3}\right)^{2d} \geq \frac{2^{-d}}{3d+1} \geq \frac{2^{-d}}{3n+1}.$$

Lemma. $\binom{3d}{d} \geq \left(\frac{27}{4}\right)^{d} \frac{1}{3d+1}$

Proof: blackboard

Then,

$$\mathbb{P}(\text{success}) \geq 2^{-n}\sum_{d=0}^{n}\binom{n}{d}\frac{2^{-d}}{3n+1} = \frac{(3/4)^{n}}{3n+1}.$$

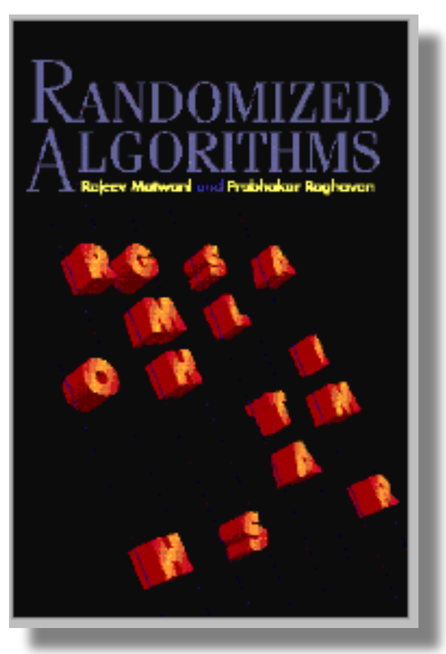
WalkSat gives a Monte Carlo algorithm in time $\left(\frac{4}{3}\right)^{n}\text{poly}(n)$.

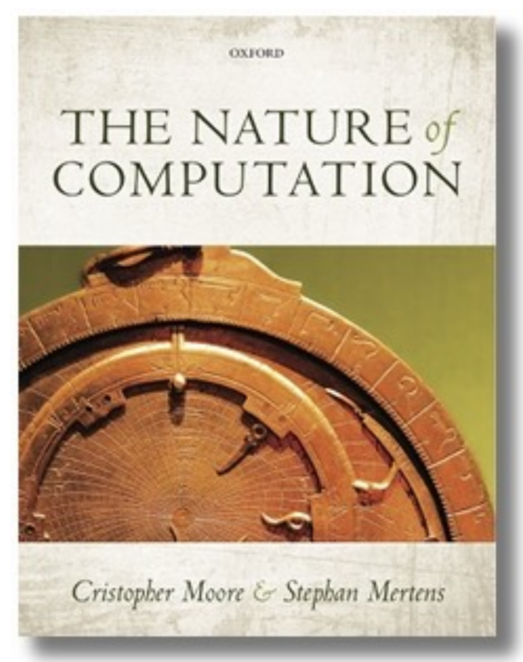
Today's best algo is in time 1.32^{n}.

References for this lecture

The slides are designed to be self-contained.

They were prepared using the following books that I recommend if you want to learn more:





Next

Assignment this week: where are the good starting points?

Next tutorial: midterm exam

Next week: Amortization

Feedback

Moodle for the slides, TDs and exercises.

Questions or comments: Bruno.Salvy@inria.fr