INF554: Randomized algorithms **1st semester, 2014**

Lecture 3 — 6th of October

Lecturer: Frederic Magniez *Scribe: Lucas Smaira*

3.1 Las Vegas search algorithms

Definition 3.1. Las Vegas *An algorithm A solve a problem taking an average time T, $\forall$ input x and choice of probability r. $A(x,r)$ is a solution for the problem and also $\forall$ input x, we have :*

$$\mathbb{E}_r\left(\textit{run time of } A(x,r)\right) \leq T$$

Example : Quick Sort randomness is used to select the recursive pivots

$$\mathbb{E}_{\text{pivots choices}}\left(\text{QS}(\text{vector of size } n)\right) = O\left(n \log n\right)$$

Theorem 3.2. *If there exists an algorithm Las Vegas A which solves the problem P taking an average time T, then there exists an algorithm Monte Carlo B which solves the same problem and takes a time $3T$ and error at most $1/3$*

Remark : The algorithm B is never wrong, but sometimes it doesn't answer.

Proof: B : execute A and stop it after time $3T$. Give the answer if it was possible to find it and doesn't answer otherwise.

We fix x and we want to calculate $\mathbb{P}_r\left(A(x,r)\text{takes a time at most}3T\right)$.

We define $\tau(r) = \text{execution time of} A(x,r)$.

We know that $\mathbb{E}_r\left(\tau(r)\right) \leq T$.

Using the Markov's inequality, we conclude :

$$\mathbb{P}_r\left(\tau(r) \geq 3T\right) \leq \frac{1}{3T}\mathbb{E}_r\left(\tau(r)\right) \leq \frac{1}{3}$$

□

3.2 *st*-connectivity

Undirected *st*-connectivity (USTCON) is the decision problem asking whether two vertices $(s,t) \in V^2$ in an undirected graph $G = (V,E)$ are connected by a path. Let $|V| = n$ and $|E| = m$. Note that if we assume the graph is connected we have $n-1 \leq m \leq n(n-1)/2$.

The deterministic depth-first search (DFS) and breadth-search first (BFS) algorithms can solve this problem in linear time complexity $O\left(m+n\right) = O\left(m\right)$ because in the worst case scenario they will visit every edge and vertex. Their space complexity is $O\left(n\right)$ as the algorithms need to store their past and future search paths in the graph. The space complexity of these algorithms can be problematic for very large graphs (for example, the Internet graph). Therefore, we wish to find an algorithm with a smaller space complexity.

In summary :

USTCON

Input $G(V, E)$ undirected graph with $|V| = n$ vertices and $|E| = m$ edges. $s, v \in V$

Output Accept if s and v are connected. Reject otherwise.

Definition 3.3. L *(logarithmic-space, also known as* LSPACE*) is the complexity class containing decision problems which can be solved by a deterministic Turing machine using a logarithmic amount of memory space.*

Definition 3.4. NL *(non-deterministic logarithmic-space, also known as* NSPACE*) is the complexity class containing decision problems which can be solved by a non-deterministic Turing machine using a logarithmic amount of memory space.*

Definition 3.5. RL (randomized logarithmic-space)*, sometimes called* RLP (randomized logarithmic-space polynomial-time)*, is the complexity class of computational complexity theory problems solvable in logarithmic space and polynomial time with probabilistic Turing machines with one-sided error.*

Note that L $\subseteq$ RL $\subseteq$ NL. Considering the space complexity of the DFS and BFS algorithms we know that USTCON $\in$ NL. In fact, it has been shown in 2005 by Reingold that USTCON $\in$ L. However, here we will only prove the following theorem :

Theorem 3.6. *USTCON $\in$ RL.*

Remark : More precisely, there is an probabilistic algorithm which takes time $O(mn)$ and space $O(\log n)$ with error at most $1/3$

To do so we consider a randomized algorithm.

Algorithm 1 USTCON Las Vegas Algorithm

```
Require: $s, t \in V$
Ensure: boolean indicating whether there exists a path from $s$ to $t$
  $u \leftarrow s$
  while $u \neq t$ do
      $v \leftarrow$ random element from $\{v \mid (u, v) \in E\}$
      $u \leftarrow v$
  end while
  return true
```

The space complexity of this algorithm is $O(\log n)$ because it suffices to keep the current vertex in memory, which requires a maximum of $\log_2 n$ bits. If s and t are not connected then the algorithm never terminates. If there exists a path from s to t in G then the average number of steps the algorithm requires is inferior to the cover time $C(G, u)$, which is defined as the average number of steps it takes to visit every vertex in the graph, starting at u.

Theorem 3.7. *Supposing G is connected, we define $C(G)$ as $\max_{u \in V} C(G, u)$. We have $C(G) \leq 4|V||E| = 4nm$*

Remark : Let τ be the number of iterations of the algorithm (aleatory variable). We have $\mathbb{E}(\tau) \leq C(G)$.

This theorem gives us a Monte-Carlo algorithm by terminating the Las Vegas procedure after $8knm$ steps. An iterator has to be kept in memory but the space complexity is still $O(\log n)$. Markov's inequality insures that this algorithm has a one-sided error of 2^{-k}.

Proof: To prove this we shall consider the random walk on a graph as a Markov chain with a state space V and a transition matrix P where

$$P_{ij} = \begin{cases} \dfrac{1}{\deg(i)} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$\deg(i)$ represents the degree of vertex i, i.e. the number of edges of i.

This generalization can represent random walks, reversible Markov chains, electrical grids, etc.

Let $X_t \in V$ be the position of the random walk at step $t \in \mathbb{N}$.

For $u, v \in G$ we define the hitting time as the expected time it takes for a random walk on G starting at u to reach v :

$$h_{u,v} = \mathbb{E}\left(\min\left\{t \in \mathbb{N} \mid X_t = v, X_0 = u\right\}\right)$$
$$h_i = \mathbb{E}\left(\min\left\{t \in \mathbb{N}^* \mid X_t = i, X_0 = i\right\}\right)$$

Lemma 3.8.

$$\forall (i,j) \in E, \ \ h_{i,j} \leq 2m$$

where m is the total number of edges.

Proof: In order to prove this lemma, we need to consider the stationary distribution of our Markovian process.

We will show that since all states $v \in V$ are positive recurrent (every vertex will be visited an infinite number of times, and the expected time to do so is finite) there exists a unique stationary distribution, described by a probability vector $\pi = (\pi_1, \ldots, \pi_n)$ such that $\pi_i = \sum_{(i,j)\in E} \pi_j P_{ji}$ i.e. $\pi P = \pi$.

We can easily check that $\pi_i = \frac{\deg(i)}{2m}$ is a solution.

The return time of u, defined as h_u, is actually related to the stationary distribution π_u and satisfies

Lemma 3.9. *G connected. $\forall i, j \in V, h_{ij}$ is finite.*

Proof: $\exists t, P^t_{ij} > 0 \Rightarrow h_{ij} \leq \dfrac{t}{P^t_{ij}}$ □

Lemma 3.10. $\pi_i h_i = 1$ *(and so π is unique)*

Proof: We define $h_{ii} = 0$

For $i \neq j$, we have $h_{ij} = 1 + \sum_k P_{ik} h_{kj} \Rightarrow h_j = 1 + \sum_k P_{jk} h_{kj}$

Furthermore, $\forall i, j$, we have

$$\begin{aligned}
h_{ij} + \delta_{ij} h_j &= 1 + \sum_k P_{ik} h_{kj} \\
\Rightarrow \sum_i \pi_i h_{ij} + \pi_j h_j &= \underbrace{\sum_i \pi_i}_{=1} + \sum_k \underbrace{\left(\sum_i P_{ik} \pi_i\right)}_{\pi_k} h_{kj} \\
\Rightarrow \sum_i \pi_i h_{ij} + \pi_j h_j &= 1 + \sum_k \pi_k h_{kj} \\
\Rightarrow \pi_j h_j &= 1
\end{aligned}$$

□

Thus :

$$h_u = \frac{1}{\pi_u} = \frac{2m}{\deg(u)}$$

The intuition behind this lies in the probability of $X_t = u$ being constant, and hence resembling a series of Bernoulli trials (biased coin tosses). The expected number of steps needed to get one success is then given by the expected value of the geometric distribution, which is $1/p$.

We can establish an upper bound on the hitting time h_{uv} where $(u, v) \in E$.

$$\begin{aligned}
h_v &= 1 + \frac{1}{\deg(v)} \sum_{k \in \text{neighbors}(v)} h_{kv} \\
&\geq 1 + \frac{1}{\deg(v)} h_{uv} \\
h_{uv} &\leq \left(\frac{2m}{\deg(v)} - 1\right) \deg(v) \\
&\leq 2m
\end{aligned}$$

□

Coming back to the theorem proof...

Now fix $u \in V$. By performing a depth-first search of graph G starting a u we obtain a spanning tree of G which has n vertices and $n-1$ edges. We can construct a tour $T = (u_1 = u, u_2, \ldots, u_N = u)$ of the tree such that $(u_i, u_{i+1}) \in E$ and so that it covers all the vertices of G. This tour passes each edge of the spanning tree twice, therefore $N \leq 2(n-1)$.

Note that $C(G)$ is less than or equal to the average time it takes to travel from u_1 to u_2, then from u_2 to u_3, etc. So $C(G) \leq h_{u_1,u_2} + \cdots + h_{u_{N-1},u_N}$. For all i we have $h_{u_i,u_{i+1}} \leq 2m$, and thus

$$C(G) \leq (N-1)(2m) < 4nm$$

□

The upper-bound on the expected time to go from a node to another can actually be divided by two, since there exists a path of length at most n between two connected nodes.

Generalization : we have proved the theorem for a simple graph (with no multiple-edges and no edges from a vertex to itself). We can model graphs with loops in a node by adding some edges like $(i, i) \in E$. We can also model graphs with weight integer of each edge by adding many edges between the same two vertices. Thus, the theorem is easily generalized to these cases.

We can use this upper bound of the expected run time of algorithm 1 to convert it into a Monte Carlo algorithm.

Algorithm 2 USTCON Monte Carlo Algorithm

```
Require: $s, t \in V$, $T_{max} \in \mathbb{Z}^+$
Ensure: boolean indicating whether there exists a path from $s$ to $t$
  $u \leftarrow s$
  while $u \neq t$ and number of iterations $\leq T_{max}$ do
      $v \leftarrow$ random element from $\{v \mid (u, v) \in E\}$
      $u \leftarrow v$
  end while
  if $u = t$ then
      return true
  else
      return false
  end if
```

If there is no path between s and t, the algorithm will always return false. As a corollary to theorem 3.7 we have

Corollary 3.11. *If $T \geq 8knm$, then algorithm 2 has a one-sided error less than or equal to 2^{-k}*

Proof: Using Markov's inequality and

$$C(G) = \mathbb{E}\left(\max_{u \in V} \min_{t \in T} \left\{ t \in T \mid \bigcup_{s=0}^{t} X_s = V, X_0 = u \right\}\right)$$

we can see that the probability of the algorithm returning false because it has not found t yet, even though there is a path between s and t at step $T \geq 8nm$ is

$$\mathbb{P}\left(\max_{u \in V} \min_{t \in T} \left\{ t \in T \mid \bigcup_{s=0}^{t} X_s = V, X_0 = u \right\} \geq 8nm\right) \leq \frac{C\left(G\right)}{8nm} < \frac{1}{2}$$

We can run our algorithm for a longer time $T \geq 8knm$, giving us a one-sided error of 2^{-k}, since

$$\begin{aligned}\mathbb{P}\left(t \text{ not reached at run } k\right) &= \mathbb{P}\left(t \text{ not reached at run } k \mid t \text{ not reached at run } k-1\right) \cdot \ldots \\ &\quad \mathbb{P}\left(t \text{ not reached at run } 2 \mid t \text{ not reached at run } 1\right) \cdot \\ &\quad \mathbb{P}\left(t \text{ not reached at run } 1\right) \\ &\leq \left(\frac{1}{2}\right)^k\end{aligned}$$

□

3.2.1 Examples

Linear Graph

Let $V = \{1, \ldots, n\}$ and $E = \{(i, i+1) \mid i \in \{1, \ldots, n-1\}\}$. By theorem 3.7 we have $C(G) \leq 4(n-1)n = O\left(n^2\right)$.

Complete Graph

In a complete graph, we have $C(G) \leq 4n \cdot n(n-1)/2 = O\left(()\, n^3\right)$. But in fact, we can prove a tighter bound of $C(G) \sim n \log n$.

Proof: In a complete graph the cover time is closely related to the coupon collector's problem. Let τ_i denote the first step at which i vertices have been visited. The number of steps it takes to reach a new vertex is

$$\tau_{i+1} - \tau_i = \frac{n-i}{n-1}$$

Since these events are independent we have

$$\mathbb{E}\left(\tau_{i+1} - \tau_i\right) = \frac{n-1}{n-i}$$

and we can use the approximation of the harmonic series by the natural logarithm to show that

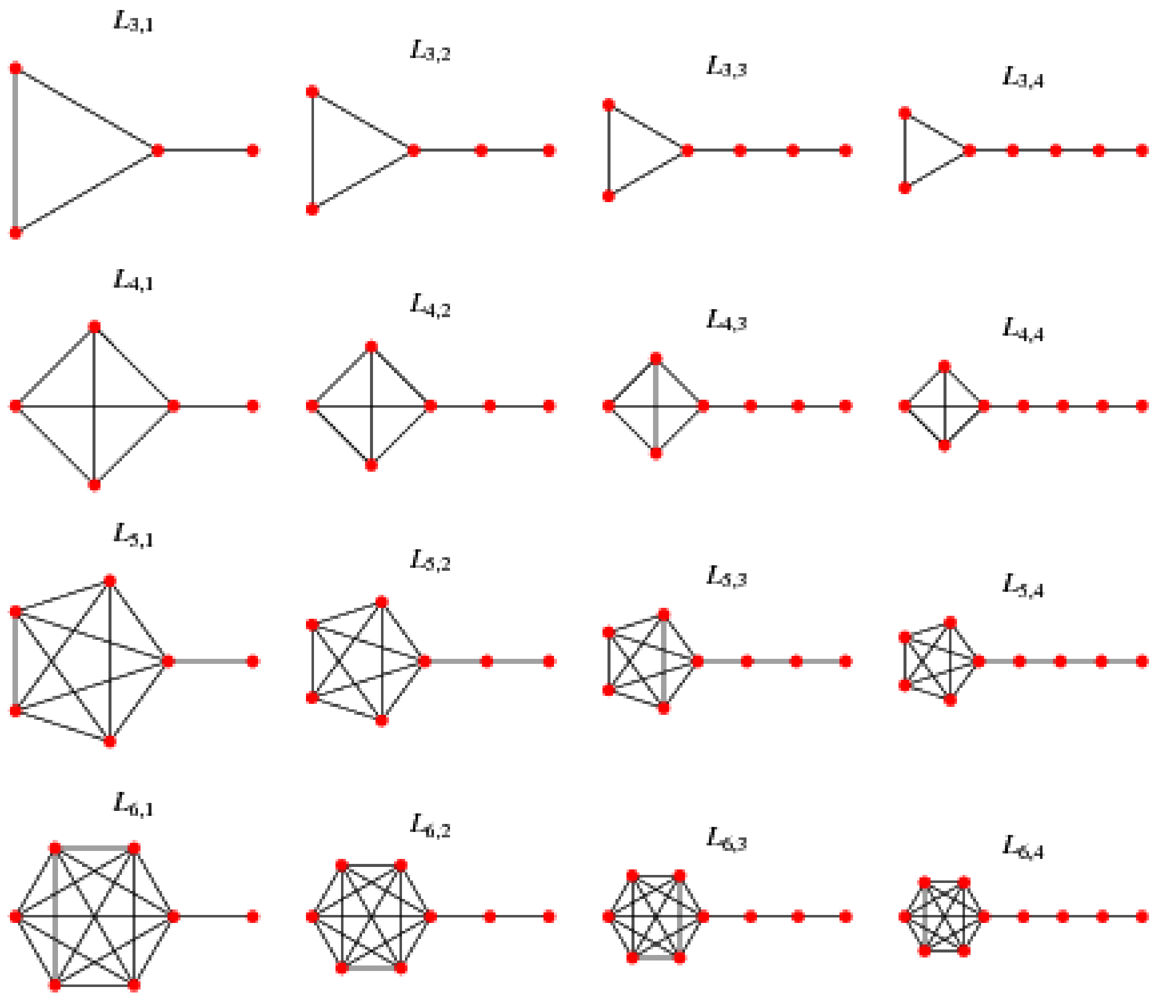
$$\mathbb{E}\left(\tau_n\right) = \mathbb{E}\left(\tau_1\right) + \sum_{i=1}^{n-1} \mathbb{E}\left(\tau_{i+1} - \tau_i\right) = 1 + \sum_{i=1}^{n-1} \frac{n-1}{n-i} = 1 + (n-1) \sum_{i=1}^{n-1} \frac{1}{i} \approx n \log n \text{ as } n \to \infty$$

□

Lollipop Graph

Lollipop graph is a graph as the conjunction of a linear graph (with $n/2$ vertices) and a complete graph (also with $n/2$ vertices). We deduce from the last section that : $C(G) \leq O\left(n^3\right)$.

The following graph is examples of Lollipop graphs.



Let s and t be left-most and right-most vertices of the linear graph. We can show that $h(s,t) = O(n^3)$ and $h(t,s) = O(n^2)$, which shows an interesting property of asymmetry in this graph.

3.3 Randomized algorithm for satisfiability : SAT

Definition 3.12. *Let $X_1, X_2, \ldots, X_n$ be $n \geq 1$ logic variables. A* literal *l is of the form X_i or $\overline{X_i}$ i.e. a literal is either a variable or the negation of a variable. A* clause *C is a disjunction of literals, for example $C = X_1 \vee X_2 \vee \overline{X_3}$. A clause C containing at most k variables is also called a* k-clause. *A* SAT formula *ϕ is of the form $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where C_i are clauses. A SAT formula ϕ is called a* k-SAT formula *if ϕ contains only k-clauses.*

The k-SAT is a decision problem to decide if for a given k-SAT formula there exist values of the boolean variables for which the formula is true.

K-SAT

Input formula ϕ k-sat

Output $a \in \{0,1\}^n : \phi(a) = 1$. Reject if there is no solution.

We will pose the following theorems without proving them.

Theorem 3.13. *2-SAT $\in P$ and k-SAT is NP-complete for all $k \geq 3$*

Theorem 3.14. *2-SAT can be solved by a deterministic algorithm with complexity $O(n+m)$ where n is the number of variables and m is the number of clauses.*

The algorithm of the above theorem is first to construct a graph with literals of all variables, and then to check whether a variable x_i and $\overline{x_i}$ are contained in the same strongly connected component. Note that the second step can be completed within linear time using Tarjan's algorithm.

We are interested in designing a probabilistic algorithm to solve k-SAT which works for all k.

Algorithm 3 Random k-SAT algorithm

```
Require: a k-SAT formula $\phi$, clauses $C_1, \ldots, C_m$, and literals $(\neg)X_1, \ldots, (\neg)X_n$   ▷ $n = km$
Ensure: a boolean indicating whether there exists an interpretation that satisfies $\phi$
  $a \leftarrow$ any value in $\{0,1\}^n$                                   ▷ $a = (X_1, \ldots, X_n)$
  while $\phi(a) = 0$ do
      $j \leftarrow$ any integer such that $C_j(a) = 0$
      $i \leftarrow$ random integer from $\{k \mid X_k$ is a variable of $C_j\}$
      $X_k \leftarrow 1 - X_k$                              ▷ Flip the bit of this variable in $a$
  end while
  return true
```

This algorithm will never terminate if there is no interpretation that satisfies ϕ i.e. $\phi(a) = 0$ for all a. If there is an interpretation satisfying ϕ, the algorithm will always find it, but there is no upper bound to its running time. Note that each iteration of the while-loop has a complexity of $O(mk)$ since it requires the evaluation of the entire formula, which has km literals.

Theorem 3.15. *If ϕ is 2-SAT and there exists an interpretation a such that $\phi(a) = 1$, then the average number of iterations needed to find a is $\leq 4n^2$.*

Corollary 3.16. *There exists an algorithm for 2-SAT with one-sided error 2^{-k} and running time $8kn^2$.*

Proof: For a 2-SAT problem let $s \in \{0,1\}^n$ such that $\phi(s) = 1$.

We define $d(a,s) = |\{a_i \neq s_i \mid 1 \leq i \leq n\}|$. If $a = s$ then $d(a,s) = 0$ and in general $d(a,s) \in \{0,1,...,n\}$. Let $X_i = d(a,s)$ after i iterations.

If the algorithm has not stopped, we have

$$\mathbb{P}(X_{i+1} = n-1 \mid X_i = n) = 1$$

$$\mathbb{P}(X_{i+1} = j-1 \mid X_i = j) \geq \frac{1}{2} \quad \text{for} \quad 1 \leq j < n$$

The first statement is obvious because if all literals have the wrong value, changing one will always decrease the distance. For the case where $1 \leq j < n$ we can consider two cases :

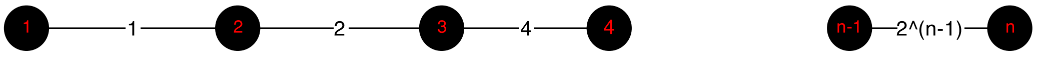
– If C is a single literal, e.g. $C = \overline{X}_5 \vee \overline{X}_5$, the distance decreases with probability 1.
– If C has two literals, e.g. $C = X_2 \vee \overline{X}_7$, the distance decreases with probability $\geq 1/2$, because at least 1 of the 2 bits is wrong, and there is a probability of $1/2$ that we flip the right one.

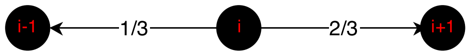
□

Note that the algorithm we have constructed is similar to a random walk on a line, where the upper bound of probability $1/2$ is the case where either direction is equally likely. We know that $h_{n,0}$ is upper bounded by $C(G) \leq 4n^2$.

As we did with the USTCON algorithms, we can bound the run time of the 2-SAT algorithm by a time $8kn^2$ to construct an algorithm with a one-sided error of 2^{-k}.

Unfortunately, for 3-SAT, the distance does not necessarily decrease at each step and the same algorithm is not enough. We can easily see that the average time to the algorithm terminate is not always polynomial by making again the comparison with a random walk on a line where the probability to go right is twice the probability to go left :





The number of edges in the graph (concentrated in its right) is exponential, and then, its cover time is also exponential.

Thereby, the state space is finite and if there is a solution it is found almost surely in a finite time. Still the algorithm can be modified to have a better convergence speed. The idea is to do the same algorithm, but only for $3n$ steps, and then to restart. It is called Walk & Restart.

Let us study the speed of this algorithm. Let p be the odd that the walk terminates in less than $3n$ steps. The expected number of restarts is obviously $1/p$.

For a 3-SAT problem let $s \in \{0,1\}^n$ such that $\phi(s) = 1$.

We define $d(a,s) = |\{a_i \neq s_i \mid 1 \leq i \leq n\}|$. If $a = s$ then $d(a,s) = 0$ and in general $d(a,s) \in \{0, 1, ..., n\}$. Let $X_i = d(a,s)$ after i iterations.

Let p_i be the probability that a $3n$-walk terminates knowing that $X_0 = i$.

$$p = \sum_{i=1}^{n} p_i \, \mathbb{P}(X_0 = i) = \sum_{i=1}^{n} p_i \, \frac{\binom{n}{i}}{2^n}$$

When a clause a selected, one literal is different from that of s since $\phi(s) = 1$, therefore $\mathbb{P}\left(X_{i+1} = X_i - 1\right) \geq 1/3$

$$p_j = \mathbb{P}\left(\min\{i,\ X_i = 0\} \leq 3n \mid X_0 = j\right) \geq \mathbb{P}\left(\min\{i,\ X_i = 0\} \leq 3j \mid X_0 = j\right) \geq$$

Algorithm 4 Random k-SAT algorithm : Walk & Restart

```
Require: a $k$-SAT formula $\phi$, clauses $C_1, \dots, C_m$, and literals $(\neg)X_1, \dots, (\neg)X_n$   ▷ $n = km$
Ensure: a boolean indicating that there exists an interpretation that satisfies $\phi$, if there
  isn't, the algorithm does not terminate.
  $t \leftarrow 0$, $a \leftarrow$ any value in $\{0,1\}^n$   ▷ $a = (X_1, \dots, X_n)$
  while $\phi(a) = 0$ do
      while $t \leq 3n$ and $\phi(a) = 0$ do
          $j \leftarrow$ any integer such that $C_j(a) = 0$
          $i \leftarrow$ random integer from $\{k \mid X_k \text{ is a variable of } C_j\}$
          $X_k \leftarrow 1 - X_k$   ▷ Flip the bit of this variable in $a$
      end while
      $t \leftarrow 0$   ▷ Restart
  end while
  return true
```

$$q_j = \mathbb{P}\left(\#\left\{i \leq 3j,\ X_{i+1} = X_i - 1\right\} = 2j,\ \#\left\{i \leq 3j,\ X_{i+1} = X_i + 1\right\} = j \mid X_0 = j\right)$$

$$q_j = \binom{3j}{j}\left(\frac{1}{3}\right)^{2j}\left(\frac{2}{3}\right)^{j}$$

$$q_j \sim \sqrt{\frac{3}{4\pi}}\frac{1}{\sqrt{j}}\left(\frac{1}{2}\right)^{j}$$

$$\sum_{i=1}^{n} p_i\,\frac{\binom{n}{i}}{2^n} \geq \sum_{i=1}^{n} q_i\,\frac{\binom{n}{i}}{2^n} \sim O(1)\sqrt{n}\left(\frac{3}{4}\right)^{n}$$

Therefore, the time complexity is $n^{O(1)}\left(\frac{4}{3}\right)^n$

As of year 2011, there is a deterministic algorithm with the same time complexity, and another randomized algorithm with time complexity $O^*(1.308^n)$

There is a version of this algorithm for k-SAT with $k \geq 4$ with time complexity $O^*\left(\left(1 + \frac{k-2}{k}\right)^n\right)$