

From the last lecture we know, in the random walk on a line, the expected time to reach  $n$  starting at 0 is  $n^2$ . In today's lecture we use random walks on line to devise algorithms for 2SAT and 3SAT.

## A Randomized algorithm for 2-SAT

Let  $\phi$  be a 2CNF formula. Consider the following algorithm.

1. Input  $\phi(y_1, \dots, y_n)$
2. Pick an arbitrary assignment  $x$ 
  - if  $\phi(x) = 1$  output  $x$  and Accept
  - else pick a clause that  $x$  does not satisfy  
(this clause has 2 literals and  $x$  sets both to be false)
  - Randomly pick one of the literals and update  $x$  by making that literal true
3. Repeat Step 2 for  $m$  times

If the input formula is not satisfiable, the the algorithm does not accept. From now, we assume that the input formula is satisfiable. Let  $a = a_1 \dots a_n$  be a satisfying assignment.

Let  $x_t$  denote the assignment at the beginning of  $t^{\text{th}}$  iteration of the loop. Let  $X_t$  be a random variable that represents the number of bits at which  $x_t$  and  $a$  match. Observe that the algorithm accepts when  $X_t$  reaches  $n$  or when  $x$  becomes another satisfying assignment of  $\phi$ .

Observe that If  $x_t$  and  $a$  matches at  $j$  places, then  $x_{t+1}$  matches with  $a$  at either  $j + 1$  or  $j - 1$  places. Thus

$$\begin{aligned} Pr[X_{t+1} = j + 1 | X_t = j] &\geq \frac{1}{2} \\ Pr[X_{t+1} = j - 1 | X_t = j] &\leq \frac{1}{2} \end{aligned}$$

However, these probabilities do not exactly correspond to a random walk on a line. However, the worst case scenario is when  $Pr[X_t = j + 1 | X_t = j] = \frac{1}{2}$  and  $Pr[X_t = j - 1 | X_t = j] = \frac{1}{2}$ . Thus the worst-case behavior of the algorithm corresponds to a random walk on a line.

From last lecture, we know that the expected number of steps to reach  $n$  from 0 is  $n^2$ . Thus the expected number of steps to for the algorithm to accept is  $n^2$ .

Let  $X$  denote the number of steps to reach  $n$ , then  $E(X) = n^2$   
According to Markov's inequality,  $Pr[X \geq a] \leq \frac{E(X)}{a}$ ,  
Thus the probability that , after  $2n^2$  steps, we have not reached  $n$  is

$$Pr[X \geq 2n^2] \leq \frac{E(X)}{2n^2} = \frac{n^2}{2n^2} = \frac{1}{2}$$

If we set  $m = 2bn$ , then the probability of error  $\leq \frac{1}{2^b}$   
 The expected time of running the algorithm  $= O(n^2b)$

## A randomized algorithm for 3-SAT

What happens when we use similar ideas to arrive at a randomized algorithm for 3SAT? The main difference is in Step 2, when we randomly pick a literal that make it true, the the probability that we picked correct literal is at least  $1/3$ , and the probability that we picked a wrong literal is at most  $2/3$ . Now, this corresponds to the following random walk on a line.

$$\begin{aligned} Pr[X_{t+1} = j + 1 | X_j = j] &= \frac{1}{3} \\ Pr[X_{t+1} = j - 1 | X_j = j] &= \frac{2}{3} \end{aligned}$$

Let  $h_j$  denote  $E(X_j)$ .

$$\begin{aligned} h_j &= \frac{1}{3}[1 + h_{j+1}] + \frac{2}{3}[1 + h_{j-1}] \\ &= \frac{h_{j+1}}{3} + \frac{2h_{j-1}}{3} + 1 \end{aligned}$$

Now,  $2h_j - 2h_{j-1} = h_{j+1} - h_j + 3$ . Let  $f_j = h_j - h_{j-1}$ . Then,

$$\begin{aligned} 2f_j &= f_{j+1} + 3 \\ f_{j+1} &= O(2^j) \\ h_j &= h_{j+1} + O(2^j) \\ h_0 &= O(2^n) \end{aligned}$$

So a similar strategy will give a  $2^n$  time algorithm. However, we know that there is a deterministic algorithm that runs in time  $2^n$ . Now we will see how to bring down the running time.

First consider the following scenario. Suppose we are at position  $n - 1$ , we assume that at any point of time, we have the ability to come back  $n - 1$  if we wish. Since  $h_{n-1} = O(2^n)$ , the expected number of steps to reach  $n$  is roughly  $2^n$ . So, if our goal is to reach  $n$ , we have the following strategy. Walk for  $2^{n+1}$  steps, then with probability at least  $1/2$ , we will reach  $n$ . However, here is a better strategy: Take one step, if we have not reached  $n$ , then go back to  $n - 1$ . Now repeat this process  $t$  times. Within each iteration, the probability of reaching  $n$  is  $1/3$ . Thus the probability that this strategy does not take us to  $n$  within  $t$  steps is  $(2/3)^t$ . This strategy is obviously better than the previous one.

Intuitively, if we have not reached  $n$  after a certain number steps, then we must have taken a large number of left moves and so after we are far away from  $n$ . Reaching  $n$  from this position takes exponential steps. So we are better off starting all over again rather than trying to reach  $n$  from this position.

Using these ideas, we arrive at the following algorithm for 3SAT.

1. Input  $\phi(y_1 \cdots y_n)$
2. Randomly pick an assignment  $x$ 
  - (a) If  $\phi(x) = 1$  then Accept  
Else ( $x$  does not satisfy a clause)  
Randomly pick a literal in the clause and make it true  
That is our new  $x$
  - (b) Repeat 2(a) for  $3n$  times
3. Repeat 2 for  $m$  times

Clearly, the algorithm does not accept if  $\phi$  is not satisfiable. Assume  $\phi$  is satisfiable, and let  $a$  be a satisfying assignment.

$x_t \rightarrow$  Assignment at  $t^{\text{th}}$  iteration of inner loop

$X_t \rightarrow$  # of places  $x_t$  and a match

Suppose  $X_0 = n - j$ . This scenario corresponds to random walk starting at position  $n - j$ .

Let's denote  $q_j$  as the probability of reaching  $n$  from  $n - j$  with  $3n$  steps. We can reach  $n$  by making  $j + k$  right moves and  $k$  left moves. Thus

$$q_j \geq \text{Max}_{j+2k \leq 3n} \binom{j+2k}{k} \left(\frac{1}{3}\right)^{j+k} \left(\frac{2}{3}\right)^k$$

Pick  $k = j$

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

By using Stirling's approximation, we obtain

$$q_j \geq \frac{c}{\sqrt{j}} \times \frac{1}{2^j},$$

where  $c$  is constant close to 1.

If  $x_0$  matches with  $a$  at  $n - j$  places, then the inner loop finds an assignment within  $3n$  steps with probability  $\geq q_j$ . So the probability that the inner loop finds an assignment within  $3n$  steps is: (let's denote this prob as  $q$ )

$$q \geq \sum_{j=0}^n \Pr[X_0 = n - j] q_j$$

Thus

$$q \geq \sum_{j=0}^n \binom{n}{j} \frac{1}{2^n} \times \frac{c}{\sqrt{j}} \times \frac{1}{2^j} \geq \frac{c}{\sqrt{n}} \times \left(\frac{3}{4}\right)^n$$

So the expected number of times that the inner loop need to be repeated till it finds a satisfied assignment is  $\frac{1}{q}$ . By Markov's inequality, if we repeat inner loop  $\frac{2}{q}$  times, we obtain a satisfied assignment with probability  $\geq \frac{1}{2}$ . Thus we repeat the entire algorithm  $n$  times, then the error probability is  $1/2^n$ . The running time of the algorithm is  $O\left(\left(\frac{4}{3}\right)^n n^2\right)$ . This is a huge improvement over  $2^n$ , moreover this algorithm is very easy to implement.