re·cur·sion (ri'kərZHən)

(noun) See recursion.

Atul Vyas Memorial Lecture

How to roll a five-sided die

Mark Huber
Fletcher Jones Foundation Associate Professor of Mathematics and Statistics and George R. Roberts Fellow
Claremont McKenna College
1 Nov, 2016

Work supported by NSF grant DMS 1418495

What
is
Recursion?

Breaking up the problem

Recursion is an essential mathematical tool that breaks a problem into versions of the same problem.

A classic problem

How many ways are there to arrange n objects?
Example: $n = 3$

123	231
132	312
213	321

Each arrangement is called a permutation

Factorials

Definition

The number of ways to arrange n objects in a line is called n factorial, and written $n!$.

So for example
$$3! = 6.$$

A classic problem

Think about which object gets put into position 1

$\boxed{1}\boxed{}\boxed{} \quad \boxed{2}\boxed{}\boxed{} \quad \boxed{3}\boxed{}\boxed{}$

- There are 3 choices for first position
- The remaining two boxes arrange 2 objects
- So $3! = 3 \cdot 2!$

More generally

Same idea with n objects gives a recursive formula for factorials

$$n! = n \cdot (n-1)!$$

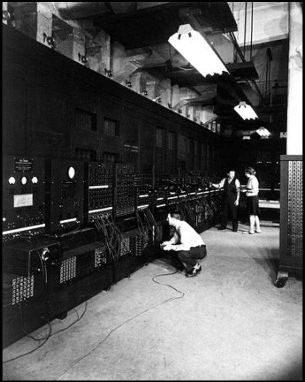
Need also a base case:

$$1! = 1$$

Can use this recursive formula for calculation

$$5! = 5 \cdot 4! = 5 \cdot 4 \cdot 3! = 5 \cdot 4 \cdot 3 \cdot 2! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

For a computer...



`Factorial` *Input:* n *Output:* $n!$
1) If $n = 1$ then output 1 and quit 2) Else output $n \cdot \texttt{Factorial}(n-1)$

Simulation via recursion

Problem

Suppose I want to generate uniformly from the set of permutations of n objects. So

$$\mathbb{P}(X = (x_1, \ldots, x_n)) = \frac{1}{n!}.$$

Notation

- $\mathbb{P}$ means probability of
- $(x_1, \ldots, x_n)$ is an arbitrary permutation (ex: $(1, 2, 3)$)
- X is a random variable that is the random permutation

Using recursion and symmetry

Using recursion to generate a random permutation:

- The first position is equally likely to be $1, 2, \ldots, 1/n$.
- Then generate the rest of permutation from items that are left.

For example:

$$\boxed{}\,\boxed{}\,\boxed{} \longrightarrow \boxed{2}\,\boxed{}\,\boxed{} \longrightarrow \boxed{2}\,\boxed{1}\,\boxed{} \longrightarrow \boxed{2}\,\boxed{1}\,\boxed{3}$$

Equally likely to be $\{1, 3\}$

Equally likely to be $\{1, 2, 3\}$

Equally likely to be $\{3\}$

In pseudocode

`Uniform_Permutation`

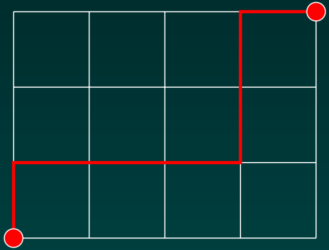
Input: item set $I = \{i_1, \ldots, i_n\}$

Output: permutation $(x_1, \ldots, x_n)$

1) If $n = 0$ then output $\emptyset$ and quit
2) Else
3) Choose i uniformly at random from $\{i_1, \ldots, i_n\}$
4) Remove i from the set I
5) Output $(i, \texttt{Uniform\_Permutation}(I))$

Path counting

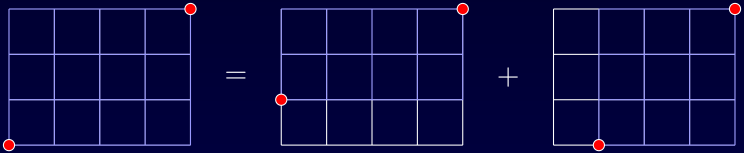
How many paths are there from (0,0) to (4,3) that use only right moves or up moves?



Answer: use recursion!

Each path to (x, y) either begins with...

- ...an up move. # of paths from $(1, 0)$ to (x, y) same as paths from $(0, 0)$ to $(x - 1, y)$, or...
- ...a right move. # of paths from $(0, 1)$ to (x, y) same as paths from $(0, 0)$ to $(x, y - 1)$.



$$P(x, y) = P(x - 1, y) + P(x, y - 1), \quad P(0, 1) = P(1, 0) = 1.$$

Table of paths

Using the recursive formula gives:

				y		
		0	1	2	3	4
	0	1	1	1	1	1
	1	1	2	3	4	5
x	2	1	3	6	10	15
	3	1	4	10	20	35
	4	1	5	15	$\boxed{35}$	70

$$P(4,3) = 35$$

Pascal's Triangle

Turned 45 degrees, gives Pascal's Triangle:

$$
\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1
\end{array}
$$

Uses: # of subsets of a set, Binomial coefficients

Simulation question

Sampling uniformly from grid paths

Need 4 right moves and 3 up moves

Randomly permuted

First move has a $4/(4+3)$ chance of being to the right

(Otherwise move up)

Recursively draw the rest of the path

The pseudocode

Up_Right_Grid_Path
Input: x, y
Output: path consisting of up and right moves

If $x = y = 0$ return $\emptyset$
With probability $x/(x+y)$
    $P \leftarrow \texttt{Up\_Right\_Grid\_Path}(x-1, y)$
    Output (right, P)
Otherwise
    $P \leftarrow \texttt{Up\_Right\_Grid\_Path}(x, y-1)$
    Output (up, P)

Structure

So far, our recursions have two elements:

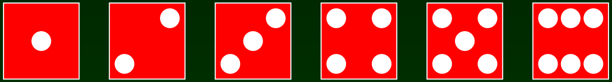
- A recursive call to the same algorithm
- A base case

What if we don't have a base case!

Infinite Recursion

A fair six sided die

Suppose I have a die with six sides



I am equally likely on a roll to get one of $\{1, 2, 3, 4, 5, 6\}$

I can roll the die as many times as I want

How can I use this to roll a five sided die?

Use infinite recursion

Intuitive answer:

1. Roll a six sided die. If the answer is in $\{1, 2, 3, 4, 5\}$, output answer and stop.
2. Otherwise, recursively roll a five sided die.

In pseudocode

`Fair_five_sided_die`

Output: X uniform draw from $\{1, 2, 3, 4, 5\}$

1) Let X the roll of a fair six sided die
2) If $X \in \{1, 2, 3, 4, 5\}$, output X and quit
3) Else
4) $\quad X \leftarrow$ `Fair_five_sided_die`
5) $\quad$ Output X and quit

Remember our earlier code for permutations

`Uniform_Permutation`
Input: item set $I = \{i_1, \ldots, i_n\}$
Output: permutation $(x_1, \ldots, x_n)$

---

1) If $n = 0$ then output $\emptyset$ and quit
2) Else
3)   Choose i uniformly at random from $\{i_1, \ldots, i_n\}$
4)   Remove i from the set I
5)   Output $(i, \texttt{Uniform\_Permutation}(I))$

The first line is the base case

Infinite recursion

`Fair_five_sided_die` does not have a base case!

- Could call recursives twice...
- ...or a million times...
- ...or a billion times.
- Very unlikely to do so, but it could happen!
- There is no upper bound on the number of recursions

Our intuition is that this works

Theorem (H. 2015)

The Fundamental Theorem of Perfect Simulation [FTPS] says essentially that when proving a recursive probabilistic algorithm works, as long as the algorithm terminates with probability 1, you can assume that the recursive call generates from the correct distribution.

Applying the FTPS to `Fair_five_sided_die`

Fact

The output of `Fair_five_sided_die` *is equally likely to be any of* $\{1, 2, 3, 4, 5, 6\}$

`Fair_five_sided_die`

Output: X uniform draw from $\{1, 2, 3, 4, 5\}$

Let X the roll of a fair six sided die
If $X \in \{1, 2, 3, 4, 5\}$, output X and quit
Else
    $X \leftarrow$ `Fair_five_sided_die`
    Output X and quit

Applying the FTPS to `Fair_five_sided_die`

Fact

The output of `Fair_five_sided_die` *is equally likely to be any of* $\{1, 2, 3, 4, 5\}$

Proof

The only way the algorithm never terminates is if we roll an infinite number of 6's in a row:

$$6, 6, 6, 6, 6, \ldots.$$

The chance of this happening is

$$\frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} \cdots = \lim_{n \to \infty} \left(\frac{1}{6}\right)^n = 0.$$

So the FTPS can be applied!

Applying the FTPS to `Fair_five_sided_die`

Proof (continued)

Now consider, what is $\mathbb{P}(X = 3)$? Well, X could equal three in line 1, which happens with probability $1/6$. Or $X = 6$ (which happens with probability 1/6), and we roll X recursively at line 4. The FTPS says that we can assume the line 4 call has the correct probability, so

$$\mathbb{P}(X = 3) = \frac{1}{6} + \frac{1}{6} \cdot \frac{1}{5} = \frac{5+1}{30} = \frac{1}{5}$$

The same argument gives $\mathbb{P}(X = i) = 1/5$ for $i \in \{1, 2, 3, 4, 5\}$, so we're done! $\square$

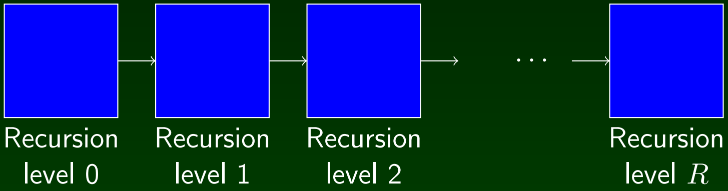
More about the FTPS

The Fundamental Theorem of Perfect Simulation

- Very useful in proofs for probabilistic recursive algorithms
- Another term for probabilistic recursive algorithm is perfect simulation
- Hence the name
- Why does it work?

Fundamental Theorem of Perfect Simulation proof

Original algorithm uses recursion a random number of times R:

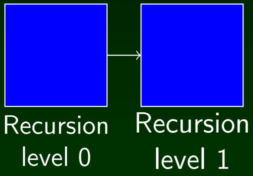


Call the output of this algorithm X

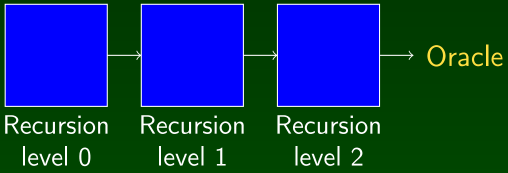
Fundamental Theorem of Perfect Simulation proof

Set n, and if $R > n$, use magic (oracle) instead of recursion.

Example: $n = 2$ and $R = 1$



Example: $n = 2$ and $R = 4$



Call the output of this algorithm Y_n

Fundamental Theorem of Perfect Simulation proof

Outline of proof of FTPS

- Because we used the oracle, Y_n has the target distribution
- Now consider as n becomes larger and larger...
- Because R is finite with probability 1,

$$\lim_{n \to \infty} Y_n = X \text{ (with probability 1)}$$

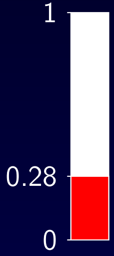
- So X also must have the correct distribution!

Flipping a biased coin

Biased coin

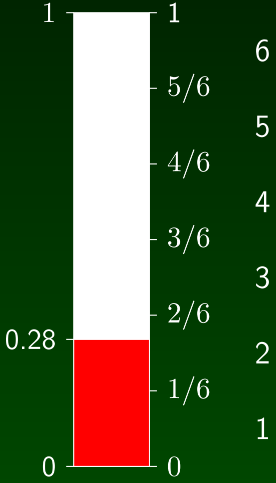
Suppose that I want to simulate an event

- The event happens 28% of the time.
- Represent by a picture:



- All I have is my trusty six sided die

Divide probability using the die



Let X be a roll of the die

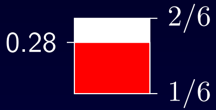
If $X = 1$ definitely in red region

If $X \geq 3$ definitely in white region

If $X = 2$ could be either red or white

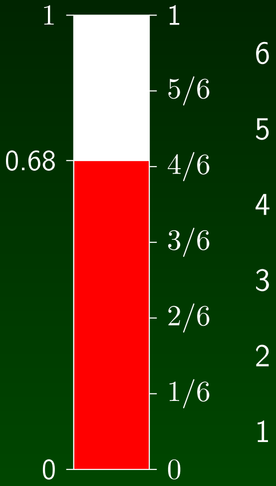
What to do when $X = 2$? Recursion!

When $X = 2$, the bar looks like:



- The percentage of red area is $(0.28 - 1/6)/(2/6 - 1/6) = 0.68$
- Flip a new coin recursively with probability 68%

Divide probability using the die



Let X be a roll of the die

If $X \leq 4$ definitely in red region

If $X = 6$ definitely in white region

If $X = 5$ could be either red or white

How to find regions

Floor function rounds down to nearest integer:

$$\lfloor 4.2 \rfloor = 4, \quad \lfloor 3 \rfloor = 3$$

Ceiling function rounds up to nearest integer:

$$\lceil 4.2 \rceil = 5, \quad \lceil 3 \rceil = 3.$$

For $p = 0.68$,

- $X \leq \lfloor 6 \cdot 0.68 \rfloor = \lfloor 4.08 \rfloor = 4$ in red region
- $X \geq \lceil 6 \cdot 0.68 \rceil + 1 = 6$ in white region

Pseudocode

```
Event_decision
Input: p (probability event occurs)
Output: S or F (Success or Failure)
```

1) Randomly draw X uniformly from $\{1, 2, 3, 4, 5, 6\}$
2) If $X \leq \lfloor 6p \rfloor$
3) Output S and quit
4) Elseif $X \geq \lceil 6p \rceil + 1$
5) Output F and quit
6) Output `Event_decision`$(6p - \lfloor 6p \rfloor)$

Notes on `Event_decision`

The input p can change at every level of recursion:

$$0.28 \to 0.68 \to 0.08 \to 0.48 \to 0.88 \to 0.28 \to$$

Sequence cycles if and only if original p is a rational number.

Can use FTPS to prove that algorithm is correct

Independent Sets of a Graph

Graphs

Definition

A graph consists of nodes and edges which connect pairs of nodes

A line graph with 5 nodes

Independent sets

Definition

An independent set of a graph is a subset of nodes, no two of which are adjacent.

Not an independent set

An indpendent set

Question: how many independent sets are there in a line graph with n nodes?

The node on one end can be out of the ind. set, leaving $n-1$ nodes

The node on the end is in the ind. set, so the node next is out, leaving $n-2$ nodes

The recursive formula is

$$F(n) = F(n-1) + F(n-2).$$

Fibonacci

Number of independent sets in a line graph with n nodes:

$$F(n) = F(n-1) + F(n-2), \quad F(0) = F(-1) = 1.$$

This gives rise to the famous Fibonacci sequence

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Leonardo of Pisa aka Fibonacci

1170-1250

In the Liber Abaci (Book of Calculation), he introduced Hindu-Arabic numerals to Europe, and introduced the Fibonacci sequence as an example.

Simulating uniformly over the independent sets

Can try acceptance rejection:

- For each node, flip a fair coin
- If heads, try to put the node in the independent set
- If result is actually an independent set accept, otherwise reject and start again

Draw a sample

reject and try again

accept as independent set!

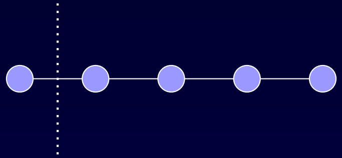
One tiny problem...

This is very, very, slow!

Speeding things up....

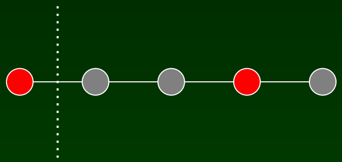
Suppose that I break the line graph into two pieces

This is called a cut of the graph

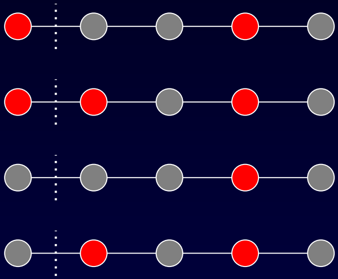


Piece by piece AR

Draw independent sets for the two sides of the cut



Bringing the two halves together



Accept!

Reject!

Accept!

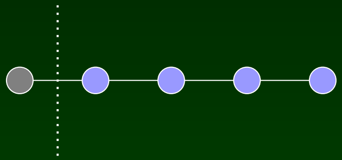
Accept!

If together they form an independent set of original graph accept

Otherwise start over

Something to note

If the left side is not in the independent set, accept for any right side!



We don't even need to draw the right side in order to accept!

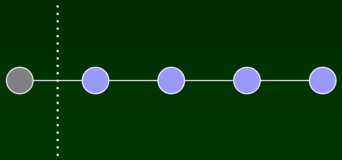
New algorithm!

Generate first node of independent set of a line graph with n nodes

1. Uniformly choose node 1 in or out of the ind set. If it is out of ind set, accept and return
2. Else recursively draw the value for node 2 of ind set on nodes $\{2, \ldots, n\}$
3. If node 2 is in the ind set, reject both sides and start over.
4. Otherwise, accept and return

New algorithm in action

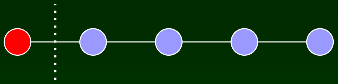
First random choice puts node 1 out of independent set...



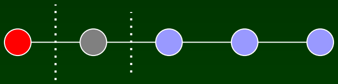
Accept and return

New algorithm in action

First random choice puts node 1 in independent set...



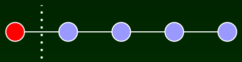
Recursively draw value for node 2. Say it is out of ind set



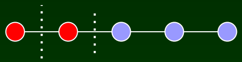
Then accept node 2, which means also accept node 1!

New algorithm in action

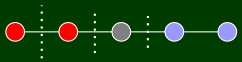
First random choice puts node 1 in independent set...



Recursively draw value for node 2. Say node 2 is in the ind set



Recursively draw value for node 3. Say it is out of the ind set



Then node 2 gets accepted as in the ind set, which means node 1 gets rejected because it was next to a node already in the independent set! Reset and start over!

What is the chance that left most node is in the independent set

Suppose we have a line graph with n nodes...

...let p_n be chance that left most node in independent set

- To be in the independent set, we have to try to put the node in the independent set. This happens with probability $1/2$.
- Then either 1) the node next to it is not in the independent set or 2) the node next to it is in the independent set and we reject, start over, and eventually put the first node in.

$$p_n = \frac{1}{2}\left[(1 - p_{n-1}) + p_{n-1}p_n\right]$$

The sequence of probabilities

So we know:

$$p_n = \frac{1}{2}\left[(1 - p_{n-1}) + p_{n-1}p_n\right]$$

Doing some algebra:

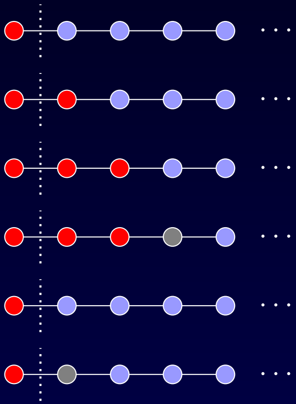
$$p_n = \frac{1 - p_{n-1}}{2 - p_{n-1}}$$

Use this an $p_1 = 1/2$ to find the first few values:

$$p_1 = \frac{1}{2},\ p_2 = \frac{1}{3},\ p_3 = \frac{2}{5},\ p_4 = \frac{3}{8},\ p_5 = \frac{5}{13}, \ldots$$

Infinite line graph

This algorithm works even if you have an infinite number of nodes!



Accept!

What is the chance that the first node is in the set

Instead of

$$p_n = \frac{1 - p_{n-1}}{2 - p_{n-1}}$$

you get

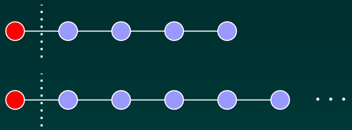
$$p = \frac{1 - p}{2 - p}$$

Unique solution:

$$p = (1/2)(3 - \sqrt{5}) = 0.3857\ldots$$

Moving from finite graph to infinite graph

As the graph gets larger, $p_n \to p$:



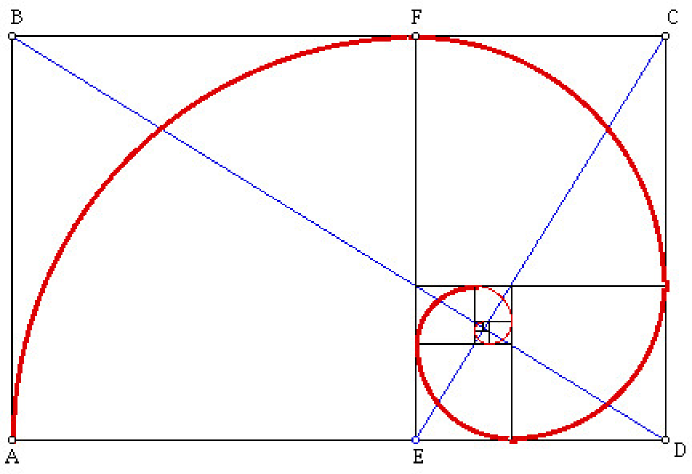
Related to a well known constant

The constant can be rewritten as

$$p = \frac{1}{2}(3 - \sqrt{5}) = 1 - \frac{1}{\phi},$$

where ϕ is the Golden Ratio

Geometric view of Golden Ratio



The Fundamental Theorem of Perfect Simulation

Gives us multiple ways to simulate from tough examples:

- Acceptance/rejection
- Coupling from the past
- Randomness Recycler
- Popping algorithms
- Bernoulli factory

The wonderful idea of infinity

How can we satisfy ourselves without going on in infinitum? And, after all, what satisfaction is there in that infinite progression?

David Hume, 1779

Thank you!