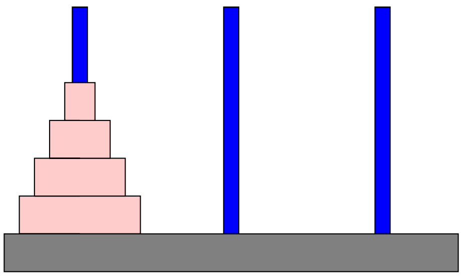
# Recurrence

Last updated: Monday 15th February, 2016 16:54

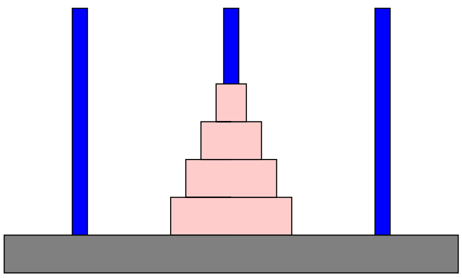
## Tower of Hanoi

Consider the following game of $n$ circular disks of different size and three poles. At the start of the game all the disks are all in one of the pole ordered by size. The bigest disk is at the bottom and the smallest disk is at the top. The goal is to move all the disk to another pole. However, you are only allowed to place the disk only on the bigger one.

Here is the starting position:



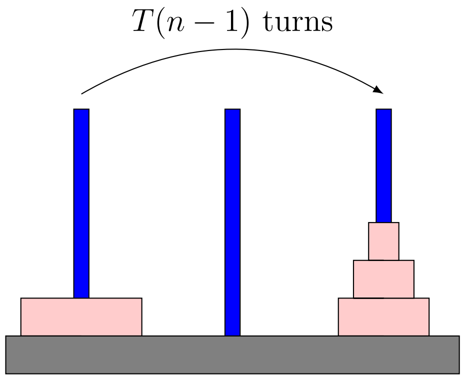
Here is the goal:



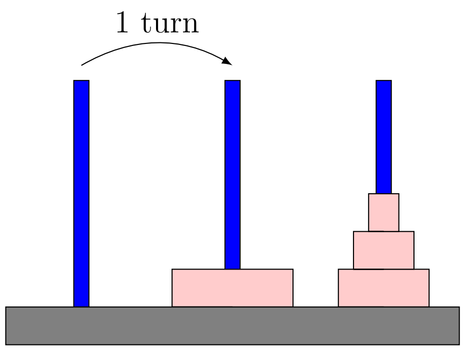
The question is how many move do we need to move $n$ disk($T(n)$).

After you have played it for a couple games we will find that the strategy for moving $n$ disks is

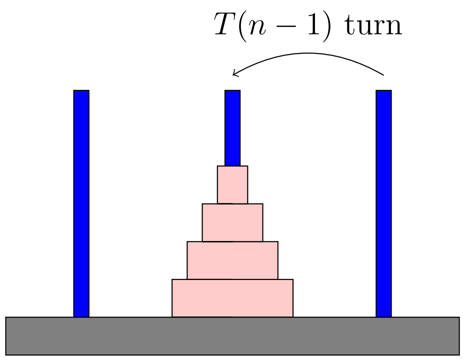
1) First, we need to move $n-1$ smallest disks to another pole first, this takes $T(n-1)$ turns.




2) Then we move the largest disk to the the desired pole. This takes 1 turn.




3) Lastly, we move the $n-1$ smallest disk to the desire pole.




So, the total number of turn needed to move $n$ turn is

$$T(n) = 2T(n-1) + 1$$

The relation above does not really love like we have solve the problem. But, actually we did. If we can figure out how many move to move 1 disk($T(1)$) then we can get the number of turn to move 2 disks($T(2)$). With $T(2)$ we can get $T(3)$ and so on.

Finding $T(1)$ is quite easy we need exactly one step to solve the hanoi tower problem of $n$ disk. So the complete description of $T(n)$ is

$$T(n) = 2T(n-1) + 1; T(1) = 1$$

This formula is not very convenient for calculating how many turns we need. We want a closed form formula.

# Plug and chug

This is the brute force way that I really want you to be able to do. The strategy is just write it out and see the pattern. The problem is

$$T(n) = 2T(n-1) + 1; T(1) = 1$$

Let us try to figure out the value of $T_5$. For this method is is very important that you *do not simplify*. The pattern is usually lost if you simply each step.

$$\begin{aligned}
T_5 &= 2T_4 + 1 \\
&= 2(2T_3 + 1) + 1 \\
&= 2^2T_3 + 2 + 1 \\
&= 2^2(2T_2 + 1) + 2 + 1 \\
&= 2^3T_2 + 2^2 + 2 + 1 \\
&= 2^3(2T_1 + 1) + 2^2 + 2 + 1 \\
&= 2^4T_1 + 2^3 + 2^2 + 2 + 1 \\
&= 2^4 + 2^3 + 2^2 + 2 + 1
\end{aligned}$$

After serveral number of step you can guess that

$$T_n = 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2^1 + 1$$

This sum is just the geometric sum we have learned before:

$$T_n = \frac{2^{n-1+1} - 1}{2 - 1} = 2^n - 1$$

This method is in no way proving that this is the solution. We just guess. To prove it you need to do induction. Watch carefully though the proof is a bit subtle. We have two definitions of hopefully the same function. We want to show that $T^*_n = 2^n + 1$ we found from plug and chuck and $T_n$ defined by the recurrence is the same function. So all we need to show is that $T_n$ and $T^*_n$ are equal for every single $n$.

**Theorem:** $T^*_n = 2^n - 1$ is equal to $T_n$ defined by

$$T_n = 2T_{n-1} + 1; \forall n \geq 1$$

with $T_1 = 1$

**Proof:** We will prove this by induction. **Inductive Predicate:**

P(s):= $T^*_s == T_s$

**Base Case:** $T^*_1 = 2^1 - 1 = 1$ and $T_1 = 1$ so $T_1 = T^*_1$ ✓

**Inductive Step:** Here we assume that $T^*_k = T_k$ for some k.

Now we want to show that $T^*_{k+1} = T_{k+1}$

The left hand side can be found from the definiton of $T^*_n$

$$LHS = T^*_{k+1} = 2^{k+1} - 1$$

The right hand side we use the definition of $T_n = 2T(n-1) + 1$. Then we use the IH that $T_k \underset{\substack{\uparrow \\ \text{IH}}}{=} T^*_k \underset{\substack{\uparrow \\ \text{*def}}}{=} 2^k - 1$

$$\begin{aligned}
RHS &= T_{k+1} \\
&= 2T_k + 1 && \text{Definition of } T_n \\
&= 2T^*_k + 1 && \text{By IH} \\
&= 2\left(2^k - 1\right) + 1 && \text{By Definition of } T^*_n \\
&= 2^{k+1} - 2 + 1 \\
&= 2^{k+1} - 1
\end{aligned}$$

So, for all these reason we know that with all these assumptions

$$RHS = T_{k+1} = 2^{k+1} - 1$$

which is the same thing as the left hand side. Thus,

$$T^*_{k+1} = T_{k+1}$$

So, by mathematical induction, $T_n = T_n^*$ for all $n \geq 1$. This means $T_n$ and $T_n^*$ is the same function. $\square$

## Some simple ones

As a practice try to write a recurrence for the following situations

a.) Today I have no money. On everyday, I'll have one Baht more than what I had the day before.

b.) I have 100 Baht today. At the end of every year I'll have the amount I had in the previous year plus the interest rate of 2% on the amount I had last year. Plus, I'll put in another 100 Baht.

## Linear Recurrence

Consider the following problem of counting the number of ways to walk up the $n$ step stairs. We have two choices

a.) We could walk 1 step up.

b.) Or, we could hop 2 steps up.

For example, for 4 step stairs. We could

1. step step step step.
2. hop hop.
3. step step hop
4. hop step step
5. step hop step

There are 5 in total.

We can break the problem of counting down to smaller problems. The number of ways to walk up $n$ step stairs can be computer by

$$\begin{aligned}\text{\# of ways for n step} =& \text{\# of way if we step first}\\ &+ \text{\# of way if we hop first}\end{aligned}$$

If we step first, then we need to solve for the number of ways to step up n-1 step. If we hop first, then we need to solve for the number of ways to step up n-2 step. At least we have smaller problem to solve. So, we can write this a a recurrence

$$T_n = T_{n-1} + T_{n-2}$$

We will also need a first few number to lock down all the subsequent numbers. We can just compute the number of wasy to step up 0 step and 1 step both are 1. So the complete description of the recurrence is

$$T_n = T_{n-1} + T_{n-2}; T_0 = 1, T_1 = 1$$

Of course this is the fibbonacci you have seen before many times in the homework.

This time we want to find the closed form formula for $T_n$. You can try plug and chug but, trust me, it won't get anywhere. I would be proud of you though if that is the first thing you try at least you are not afraid of brute forcing. We need a smarter way to solve this: GUESS for the solution.

Let us try to plug in $T^*(n) = x^n$ and hope that we can find $x$ that make $T^*$ at least satisfy the recurrence.

If $T^*(n) = x^n$ satisfy the recurrence that means

$$\begin{aligned}x^n &= x^{n-1} + x^{n-2}\\ x^2 &= x + 1\\ x^2 - x - 1 &= 0\end{aligned}$$

This equation is called the **characteristic equation**.

$$\begin{aligned}x &= \frac{1 \pm \sqrt{1^2 + 4}}{2}\\ &= \frac{1+\sqrt{5}}{2}, \frac{1-\sqrt{5}}{2}\end{aligned}$$

From this we know that

$$f(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n = a^n$$

and

$$g(n) = \left(\frac{1-\sqrt{5}}{2}\right)^n = b^n$$

solves the recurrence we want.

However, there is a bad news. Neither $f$ nor $g$ is the function we want. If we plugin, $n = 1$ we should get 1. But, none of the solution we had give us the value we want for $n = 1$.

# Linear Homogeneous Recurrence

But, we are not done yet. The recurrence we are solving is an example of a class of recurrence called linear homogeneous recurrence.

**Def:** A **linear homogeneous** recurrence is a recurrence of the form

$$f(n)+a_1f(n-1)+a_2f(n-2)+\ldots+a_df(n-d) = 0$$

For example, the recurrence we are trying solve can be written in this form as

$$T(n) - T(n-1) - T(n-2) = 0.$$

Not all recurrence is linear homogeneous recurrence. For instance, the recurrence we solve for the hanoi tower.

$$T(n) = 2T(n-1) + 1$$

is not a linear homogeneous recurrence since if we move the $2T(n-1)$ terms to the left hand side. We are left with 1 not 0.

Linear Homogeneous recurrence has a very nice property that if we have 2 solution then the linear combination of the two solution is another solution. If you have a chance to study ODE you will see the same trick again.

**Theorem:** Let $f_n$ and $g_n$ be solution to a linear homogeneous recurrence:

$$T(n) + a_1T(n-1)\ldots + a_dT(n-d) = 0.$$

Then, for all $r, s \in \mathbb{R}$, $rf_n + sg_n$ also solve the recurrence.

**Proof:** The proof is quite straight forward. All we need to do is to plug the linear combination into the recurrence and see if it is equal to 0.

Plugging $rf(n) + sg(n)$ in to the recurrence gives

$$\begin{aligned} LHS &= [rf_n + sg_n] + a_1[rf_{n-1} + sg_{n-1}] + \ldots a_d[rf_{n-d} + sg_{n-d}] \\ &= r[f_n + a_1f_{n-1} + \ldots a_df_{n-d}] + s[g_n + a_1g_{n-1} + \ldots a_dg_{n-d}] && \text{collecting terms} \\ &= r0 + s0 && \text{since } f_n, g_n \text{ solve the recurrence.} \\ &= 0 \end{aligned}$$

So, this means $rf_n + sg_n$ solves the recurrence $\square$

# Initial Conditions

The theorem we just prove is very useful for fitting the boundary conditions. So far for the stair problem we have 2 solutions but none of the fit the initial conditions. Any linear combination of the two is guaranteed to solve the recurrence. If we can find the linear combination that can satisfy the initial conditions then we are done.

The initial condition we need to satisfy are $T(0) = 1$, $T(1) = 1$. So, we hope that for some $a, b \in Real$

$$\begin{aligned} rf_0 + sg_0 &= 1 \\ rf_1 + sg_1 &= 1 \end{aligned}$$

The relation above implies

$$r + s = 1 \tag{1}$$

$$r\frac{1+\sqrt{5}}{2} + s\frac{1-\sqrt{5}}{2} = 1. \tag{2}$$

This is just two equation with two unknown we can solve for $r$ and $s$. This yields

$$r = \frac{1}{\sqrt{5}} \times \frac{1+\sqrt{5}}{2}$$

$$s = -\frac{1}{\sqrt{5}} \times \frac{1-\sqrt{5}}{2}$$

This means that the function that solve the recurrence and the initial condition is

$$h_n = \frac{1}{\sqrt{5}} \times \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}} \times \left(\frac{1-\sqrt{5}}{2}\right)^{n+1},$$

similar to what you prove in the homework(the initial condition is a bit different). This formula is called the Binet's formula.

Of course, this is no way a proof that is indeed a solution. We still need to show that this is true using induction. You have done that already in the homework.

# Putting it together

Since there were so many digression along the way of getting the solution. Let us do it again in one swoop to get the whole picture.

Let us try to solve the follwing recurrence

$$T(n) = -2T(n-1) + 15T(n-2)$$

with the initial conditions

$$T(0) = 1, T(1) = 2$$

First we guess that $T(n) = x^n$

$$x^n = -2x^{n-1} + 15x^{n-2}$$

This means the characteristic equation is

$$x^2 + 2x - 15 = 0$$

This can be factorized in to

$$(x-3)(x+5) = 0$$

Thus, $f(n) = 3^n$ and $g(n) = (-5)^n$ solves the recurrence.

Now we need to fit the initial conditions so we need to find $r, s \in \mathbb{R}$ such that

$$rf(0) + sg(0) = 1$$
$$rf(1) + sg(1) = 2$$

The equations that we need to solve is

$$r + s = 1$$
$$3r - 5s = 2$$

Solving this yields$r = \frac{7}{8}$ and $s = \frac{1}{8}$. So, the solution to the recurrence is

$$T(n) = \frac{7}{8}3^n + \frac{1}{8}(-5)^n$$

# Double Root and What not

Read the book if you are interested in what we should do if we got a double root.

# Maximum Buy/Sell Profit

After you graduate, you become CEO of an big investment firm. You have many many employee. You have list of stock price for $n$ days(really big $n$ $10^5$) and you want to find the best time to buy stock then the best day to sell it later. Each of your employee is required by law to compare no more than 1 pair of number a day. Otherwise, you firm have to face labor abuse lawsuit. You want to get this done in 1 day. You want to decide how many people you need.

So, you call your three managers and ask them what to do.

Your first manager said all you need to do is to compute the profit for ever pair of buy and sell date. You will need to compute $n(n-1)$ profit and compare all of them to find the maximum. This requires $n(n-1)$ comparisons. So you will need around $10^{10}$ people. You fired this manager right away since there are only 7billion people on the earth.

Your second manager said all you need to do is to give the first half to one of the manager then give the second half to another manager. Ask them for the best date of each half. Three things can happen

a.) The best pair is in the first half.

b.) The best pair is in the second half.

c.) The best buy date is in the first half and the best sell date is in the second half. To compute this one we will just need to find the miminum of the first half and the maximum of the second half. This takes $(n-1)$ comparisons.

Then, we need to compare the three this take 2 comparisons.

Of course, we do not know how many comparison the first two steps need. We know they are not going to do $n^2$ like the first manager since they will get fired right away. They will have to copy the way the second manger suggest by giving half of their work load to another group of people and hire another group to combine the result like we did. Then all the other manager will just copy your strategy recursively.[1]

This doesn't really help us count. The number of comparison need to solve the problem of size $n$ is

$$T(n) = \underset{\substack{\uparrow \\ \text{First half}}}{T(n/2)} + \overset{\substack{\text{second half} \\ \downarrow}}{T(n/2)} + \underbrace{n-1+2}_{\text{combine}}$$

Like the previous recurrence the description is not complete description if we do not have the initial condition. If the list is has only 2 points then, we will just return that point as the maximum profit(potentially loss). So we have $T(2) = 1$

So, we have a recurrence relation.

$$T(n) = 2T(n/2) + n + 1; T(2) = 1$$

This means you can find $T(2)$, $T(4)$, $T(8)$ ...and so on. Let us try to solve this

# Plug and Chug 2

Since we go by down by 2 all the times so let

$$n = 2^k$$

This means

$$k = \log_2 n$$

You should **always** try $2^k$. This will simplify the pattern a lot.

Let us compute $T(2^5)$. Again we do not want to simplify since it will break the pattern.

$$\begin{aligned}
T(2^5) &= 2T(2^4) + 2^5 + 1 \\
&= 2(2T(2^3) + 2^4 + 1) + 2^5 + 1 \\
&= 2^2T(2^3) + 2^5 + 2 + 2^5 + 1 \\
&= 2^2(2T(2^2) + 2^3 + 1) + 2^5 + 2 + 2^5 + 1 \\
&= 2^3T(2^2) + 2^5 + 2^2 + 2^5 + 2 + 2^5 + 1 \\
&= 2^3(2T(2^1) + 2^2 + 1) + 2^5 + 2^2 + 2^5 + 2 + 2^5 + 1 \\
&= 2^4T(2^1) + 2^5 + 2^3 + 2^5 + 2^2 + 2^5 + 2 + 2^5 + 1 \\
&= 2^4 + 2^5 + 2^3 + 2^5 + 2^2 + 2^5 + 2 + 2^5 + 1 \\
&= 2^4 + 2^3 + 2^2 + 2 + 1 + 2^5 + 2^5 + 2^5 + 2^5 \\
&= \sum_{i=0}^{i=5-1} 2^i + (5-1)2^5
\end{aligned}$$

So, we can guess make that

$$\begin{aligned}
T(n) = T(2^k) &= \sum_{i=0}^{i=k-1} 2^i + (k-1)2^k \\
&= \frac{2^k - 1}{2-1} + (k-1)2^k && \text{Geo sum.} \\
&= 2^k - 1 + k2^k - 2^k \\
&= k2^k - 1 \\
&= nk - 1 && n = 2^k \\
&= n\log_2 n - 1 && k = \log_2 n
\end{aligned}$$

Of course this is not a proof. It is just a really good way of finding what to prove. You will need to do induction to verify this.

[1]You can actually do much better than this by having the function returns max and min but we will not get there.

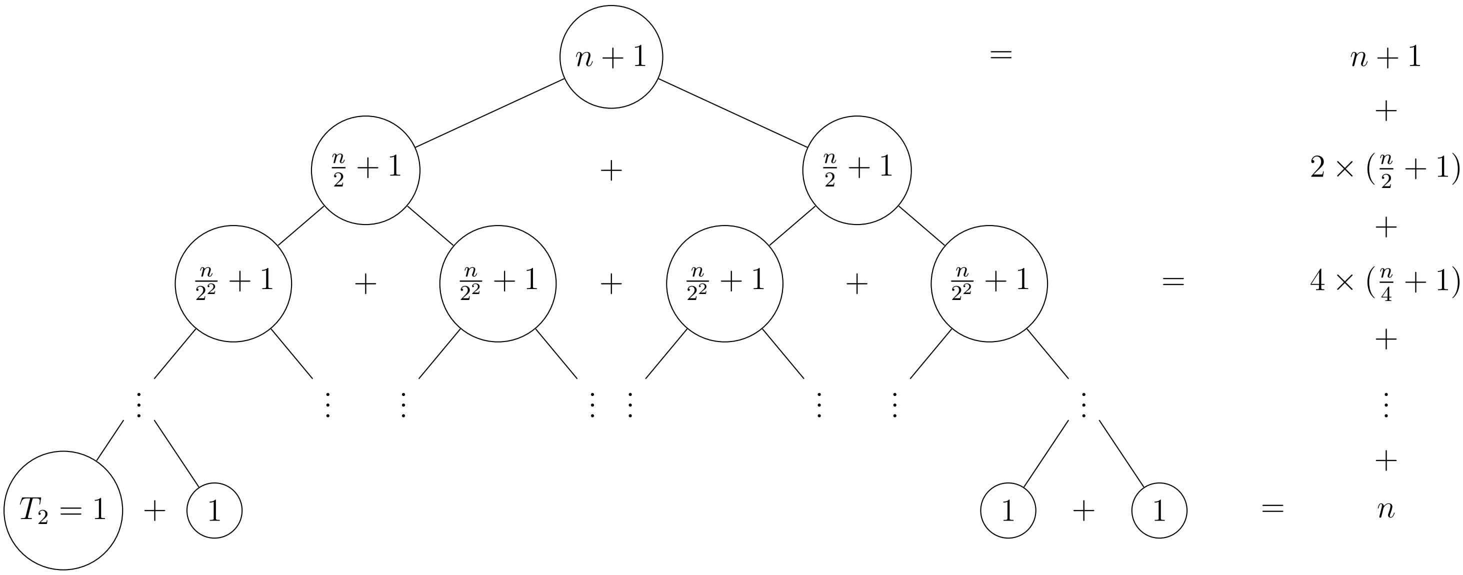
# Rounding

You may ask what happen to the number that is not $2^n$ for example $T(3)$. You can read the book.

The rounding doesn't do much to the divide and conquer recurrence. You can usually just ignore it if all you need is just the asymptotic behavior. Trust me on this.

# Drawing Tree

Another way to solve divide and conquer recurrence is to draw workload tree. Where you have the actual work done by each manager/node written on it. Then all we have to do is to sum it up systematically. You will need to be a bit careful on where to end whether it is $k$ or $k-1$. To figure that out you usually need to plug in a specific number.



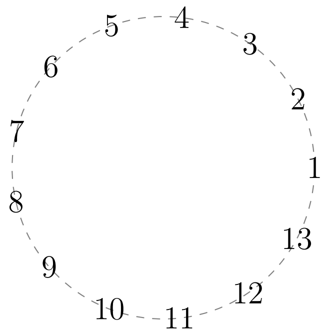

All we need to find is the num of all the number in the node

$$
\begin{aligned}
Sum &= \underset{\substack{\uparrow \\ \text{all the terms before the last level}}}{\sum_{i=0}^{i=k-2} \left(n + 2^i\right)} + \overset{\substack{\text{last level} \\ \downarrow}}{\frac{n}{2}} \\
&= (k-1)n + \sum_{i=0}^{i=k-2} 2^i + \frac{n}{2} \\
&= kn - n + 2^{k-1} - 1 + \frac{n}{2} \\
&= kn - \cancel{n} + \cancel{\frac{n}{2}} - 1 + \cancel{\frac{n}{2}} \\
&= kn - 1 \\
&= n\log_2 n - 1
\end{aligned}
$$

Which is the same thing as what we found before. This method requires you to be a bit careful in getting the bound of the sum correct. Pick whichever method you like.

# Josephus's Problem

You can read about Josephus from wikipedia he is quite a character and a writer. What we are considering here will be a variant of Josephus problem. Consider $n$ prisoner standing in a circle starting from position 1 to position $n$.




The executor start skip the one prisoner then kill one then skip one and keep repeating the process until only one survivor is left.

For example, given the 13 prisoners. The first round of execution will kill out all the even-numbered people.

$$1, \cancel{2}, 3, \cancel{4}, 5, \cancel{6}, 7, \cancel{8}, 9, \cancel{10}, 11, \cancel{12}, 13$$

Then the next after skipping 13 the excutor then proceed to kill 1 skip 3 and kill 5

$$\cancel{1}, 3, \cancel{5}, 7, \cancel{9}, 11, \cancel{13}$$

After killing 13 the executor proceed to skip 3 and kill 7

$$3, \cancel{7}, 11$$

Then he skips 11 and kill 3

$$\cancel{3}, 11$$

which means 11 survives.

The question is where do you want to stand if there are $n$ people? Let us consider this problem as a recurrence problem. First, we are going to consider the case where there are even number of people $n = 2k$. After the first round of executor. We will be left with $k = n/2$ people standing in a circle. Position 1 will have person number 1. Position 2 will have person number 3 on it and so on.

```
1, 2, 3, 4, 5 ,6 ,7, 8, 9, 10
1, 3, 5, 7, 9
```

If we happen to be able to figure out which position is safe for smaller circle we can infer about the safe position of the larger circle. Let the safe position for circle of $n$ people be $T_n$. If $n$ is even, we can ask our friend to find out which position is safe for circle of $n/2$ people, then all we need is translate that position to bigger circle.

$$T_n = 2T_{n/2} - 1 \text{ if } n \text{ is even.}$$

The case where $n = 2k + 1$ is odd is similar to the even case except that the first person will be killed so we get rid of the first person and realign them. This means that ask our friend about the survival position for $(n-1)/2$ circle then translate the position for smaller circle to position for circle with $n$ people.

```
1, 2, 3, 4, 5 ,6 ,7, 8, 9, 10, 11
3, 5, 7, 9, 11
```

This means

$$T_n = 2T_{(n-1)/2} + 1 \text{ if } n \text{ is odd.}$$

Combining the two gives us a funny recurrence.

$$T_n = \begin{cases} 2T_{(n-1)/2} + 1 & \text{if } n \text{ is odd.} \\ 2T_{n/2} - 1 & \text{if } n \text{ is even.} \end{cases}$$

Writing down a few first values for $T_n$

```
1, 3, 1, 3, 5, 1, 3, 5, 7, 1, 3, 5, 7, 9, ...
```

allows us to guess the solution. If $n = 2^m + l$ where $0 \leq l < 2^m$ then $T_n = 2l + 1$. This can be proven quite easily by induction.