Classic Application

Suppose you need to multiply **really** big numbers.

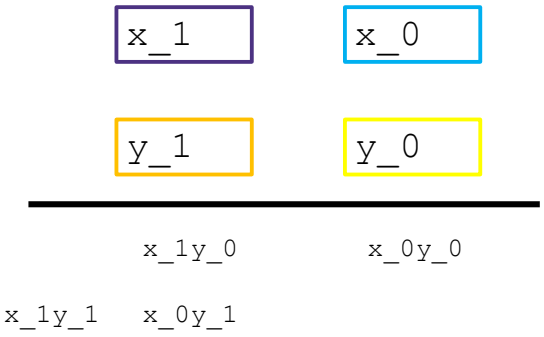
Much bigger than `ints`

Split the n bit numbers in half

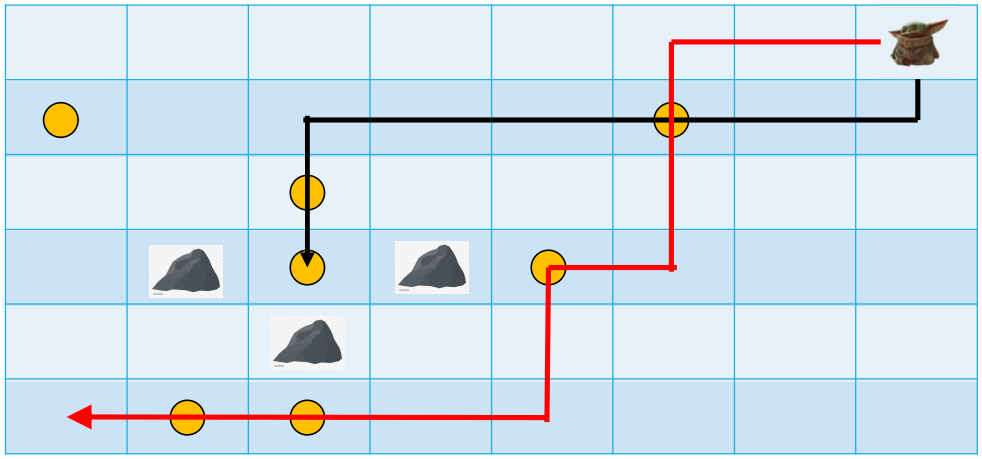
Think of them as written in base $2^{n/2}$

What would the "normal" multiplication algorithm do?

4 multiplications, i.e. 4 recursive calls.



Baby Yoda Searching



Black path: get stuck. Invalid.

Red path: valid! And optimal (no path collects more than 4 eggs.)

Let `OPT(i,j)` be the maximum number of eggs we can get on a legal path from `(i,j)` to `(0,0)` (including the egg in `(i,j)` if there is one)

What recursive calls do we need?

Don't try to divide & conquer, think closer to home…

We have to decide whether to go down or left…

Activity

Fill out the question at pollev.com/robbie

Figure out how to take advantage of the repeated calculation.
What do you think the running time will be of your new algorithm?

```
FindOPT(int i,int j, bool[][] rocks, bool[][] eggs)
    if(i<0 || j < 0) return -∞
    if(rocks[i][j]) return −∞
    if(i==0 && j==0) return eggs[0][0]
    int left = FindOPT(i-1,j,rocks,eggs)
    int down = FindOPT(i,j-1,rocks,eggs)
    return Max(left,down) + eggs[i][j]
```