Recursion

Thomas Schwarz, SJ

“The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit reference.”

ALGORITHMS +DATASTRUCTURES = PROGRAMS

N.E. WIRTH, 1976

Recursion as a Universal Tool

- Recursion possible:
  - Solution depends partially on solution(s) to (a) smaller problem(s)
- Recursion function consists of
  - Base Case
  - Call to function with smaller arguments

Examples for Recursion

- Euclidean Algorithm
  - Base case: one number is zero
  - Recursion: express the problem using smaller numbers
  - $a > b \Rightarrow \ \text{gcd}(a, b) = \text{gcd}(b, a \,\%\, b)$

Examples of Recursion:

Efficient Calculations of Powers

- Naive power calculation:
  -
    ```
    def power(x,n):
        acc = 1
        for _ in range(n):
            acc *= x
    ```
  - This uses *n* multiplications.
  - Can do better by setting `acc = x`, but that still uses $n - 1$ multiplications

Examples of Recursion:

Efficient Calculations of Powers

- There is a better way with recursion
  - If n is even, $n = 2m$: x^n is the product of x^m with itself.
  - If n is odd, $n = 2m + 1$: x^n is $x^m \cdot x^m \cdot x$
- This leads to very simple Python code

Examples of Recursion:

Efficient Calculations of Powers

- Direct Python Implementation:

```
def power(x, n):
    if n == 0:
        return 1
    if n == 1:
        return x
    if n%2 == 0:
        return power(x,n//2)*power(x,n//2)
    return power(x,n//2)*power(x,n//2)*x
```

Examples of Recursion:

Efficient Calculations of Powers

- Why does this work?
  - A formal proof can assure that we did not make an implementation mistake
  - Proof is by induction
    - Base Cases: n=0 and n=1 are directly in the code
    - Induction Step: Assume it works for all inputs up to, but not including n
      - Need to show that it also works for n

Examples of Recursion:

Efficient Calculations of Powers

- Case distinction:
  - If n is even:
    - Then $x^n = x^{n//2} \cdot x^{n//2}$ and the code works
  - If n is odd
    - Then $x^n = x \cdot x^{n//2} \cdot x^{n//2}$ and the code works
  - Here, we are using the induction hypothesis

Examples of Recursion:

Efficient Calculations of Powers

- As you can see, recursion and induction match each other closely

Examples of Recursion:

Efficient Calculations of Powers

- Performance:
  - Best case: n is a power of 2, i.e. $n = 2^m$
    - By induction: show that the algorithm takes *m* steps.
    - Each step uses one multiplication.
    - Total of $m = \log_2(n)$ multiplications

Examples of Recursion:

Efficient Calculations of Powers

- Performance:
  - Worst case: n is always odd
    - $n = 1,3,7,\ldots = a_1, a_2, a_3, \ldots$
    - Next element in this sequence is calculated from the previous: $a_j = 2 \cdot a_{j-1} + 1$
  - Can prove by induction:
    - $a_j = 2^j - 1$

Examples of Recursion:

Efficient Calculations of Powers

- Performance:
  - Worst case: $n = 2^m - 1$
  - Two multiplications per step
  - There are $m - 1$ steps
  - Total is $2m - 2$ multiplications
  - Which is $2\log_2(n) - 2$ multiplications

Examples of Recursion:

Efficient Calculations of Powers

- Performance: $O(\log(n))$

Examples of Recursion:

Efficient Calculations of Powers

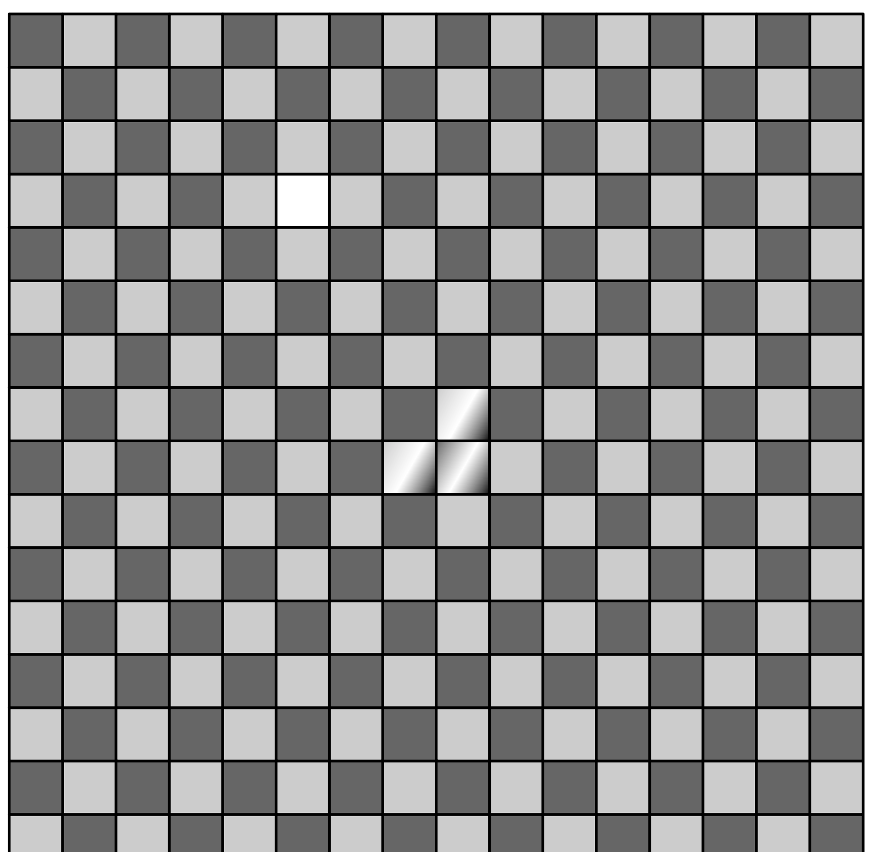
- Implementation using binary operations

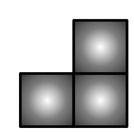
```
def power(x, n):
    if n == 0:
        return 1
    if n == 1:
        return x
    m = n >> 1
    r = n&0x01
    p = power(x,m)
    if r:
        return p*p*x
    return p*p
```

Examples of Recursion:

Triominos

- We are given a chess board of size $2^m \times 2^m$ with one field removed.
- Write a program that shows how to tessellate the chess board with triominos





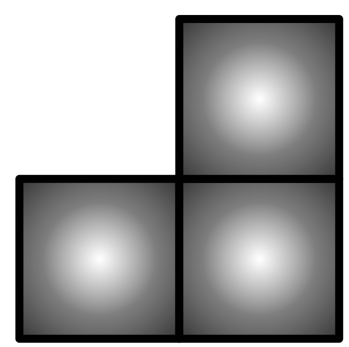
Examples of Recursion: Triominos

- Notice:
  - The chess board has $2^m \times 2^m \equiv (-1)^m \times (-1)^m \equiv 1 \pmod 3$ fields.
  - A single triomino has three fields
  - So, maybe it is possible

Examples of Recursion:

Triominos

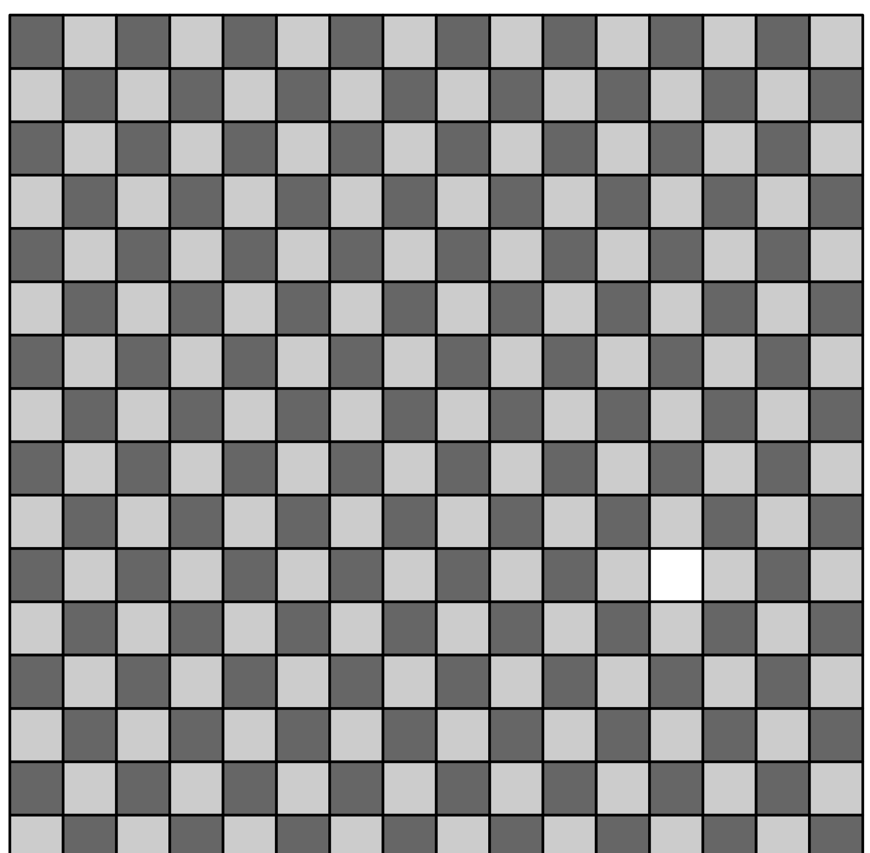
- Base Case:
    - $m = 1$.
        - A two-by-two chess board has four fields.
        - Remove one, and you have a triomino



Examples of Recursion:

Triominos

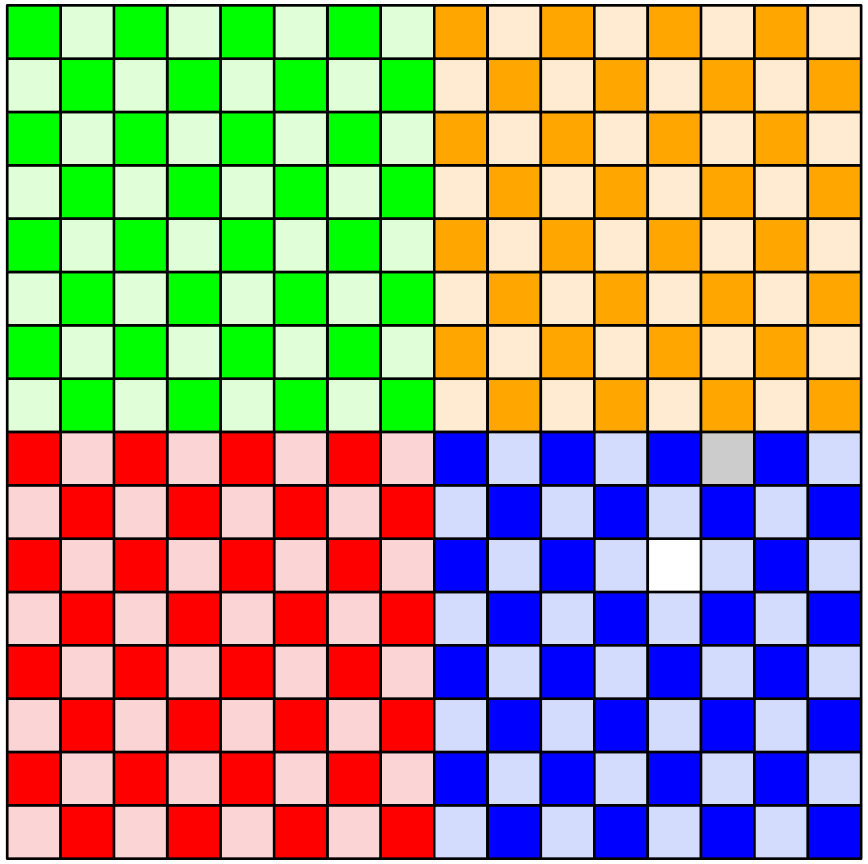
- Recursion:
  - A $2^m \times 2^m$ chessboard consists of four $2^{m-1} \times 2^{m-1}$ chessboards.
  - Take such a board with one field removed.



Examples of Recursion:

Triominos

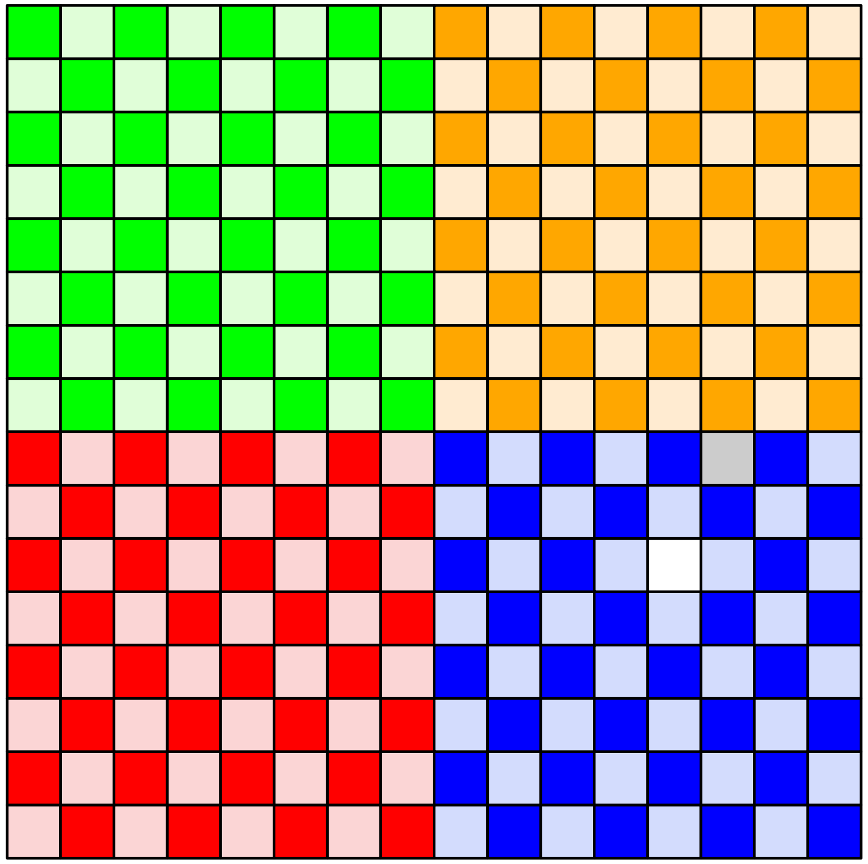
- Divide the board into four equal parts



Examples of Recursion:

Triominos

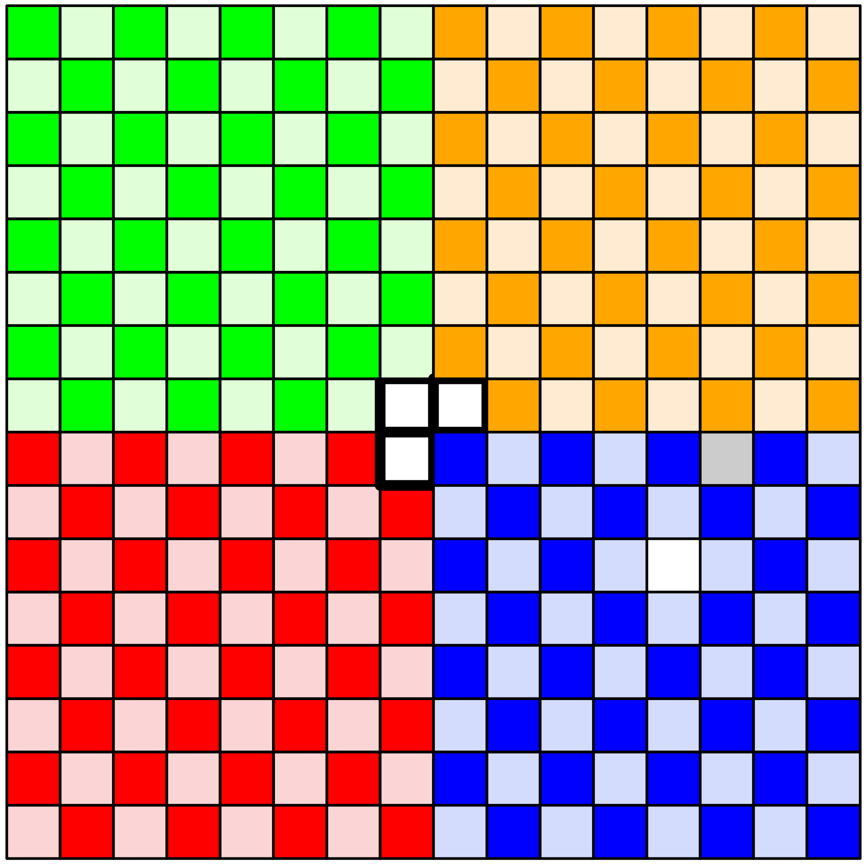
- One of them (here the blue one) has the missing field.
- Create a list of triominos that fill this up



Examples of Recursion:

Triominos

- Place a Triomino in the middle, cutting out one field from each of the other sub-boards.



Examples of Recursion:

Triominos

- Add this triomino to the list.
- The three remaining boards (green, orange, red) can now be covered with triominos