Recursion

http://bit.ly/1PjFovW

Recursion

- “the repeated application of a recursive procedure or definition”

Examples

Loops

- Loops and recursion are equivalent
- Anything that can be written with recursion can be written with a loop (albeit a complicated one in some cases)
- So why ever use recursion?

Loops

```
long factorial(int n) {
    long result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

Loops

- Advantages:
  - Very clear, obviously correct
  - No wasted memory usage on runtime stack
  - Not recursion
- Disadvantages:
  - Not recursion?

Needless recursion

```
long factorial2(int n) {
    if (n <= 1) {
        return 1;
    }
    return n * factorial2(n - 1);
}
```

Recursion

- Advantages:
  - Translates directly from math!
  - Value of `long` overflows before stack space is relevant
- Disadvantages:
  - Stack space used

Loops

```
int sum(int[] a) {
    int sum = 0;
    for (int i = 0; i < a.length; i++) {
        sum += a[i];
    }
    return sum;
}
```

Useless recursion

```
int sum(int[] a) {
    return sum(a, 0);
}
int sum(int[] a, int i) {
    if (i == a.length) {
        return 0;
    }
    return a[i] + sum(a, i + 1);
}
```

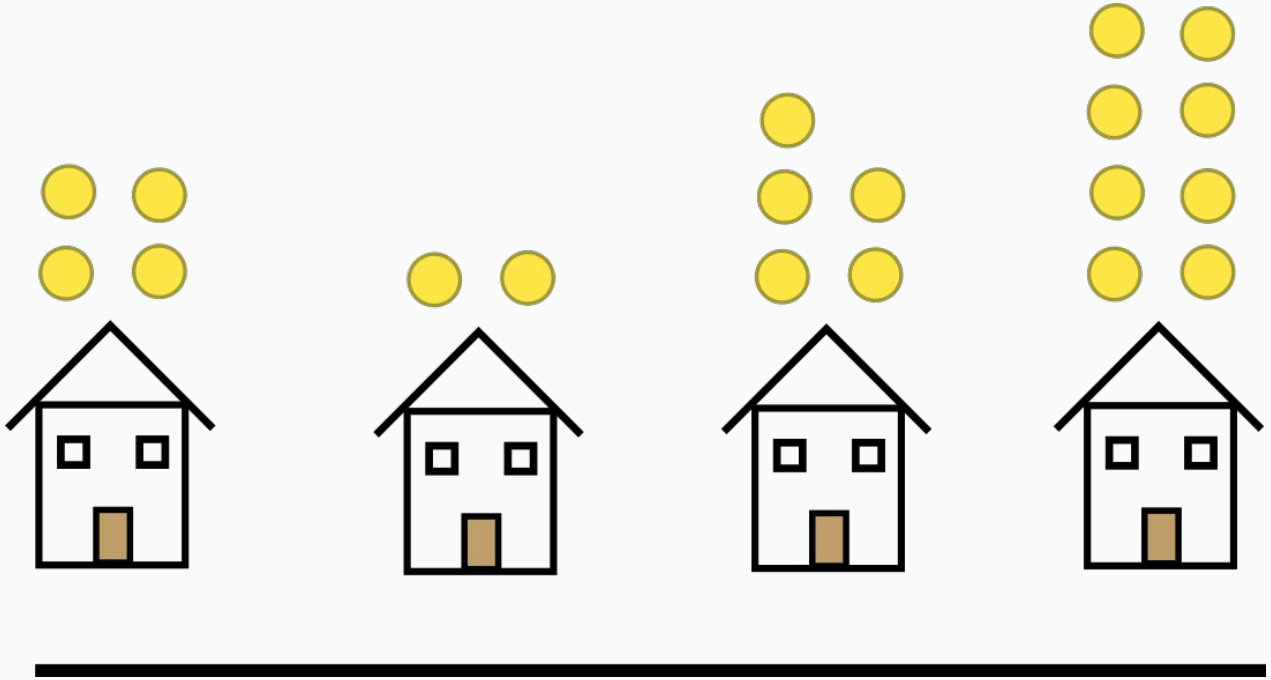
Why?

- Sometimes recursion is more clear and concise

Applications for competitive programming

- Sometimes just implementing a recursive definition
  - eg: Catalan numbers, Collatz sequence
- DFS
  - Or many different types of tree traversals
- Backtracking, or other combinatorial exploration
  - eg: All words that a phone number maps to

Example problem



Example problem

[4, 2, 5, 8]

Example problem

- What’s the best we can do?

Example problem

- 12, by taking 4, skipping 2 and 5, then taking 8

Example problem

- Can we model this recursively?

Modeling recursively

- Consider a single item in the array (a single house with money)
- What are our choices?

Modeling recursively

- We have two options:
  - Take their money
  - Leave their money
- If we take their money, we can’t rob their neighbors
- If we leave their money, we can rob their neighbors

Modeling recursively

- Think about “sub-problems”
- If we’re looking at house number *i*, and we rob them, then we can start our process over at house *i + 2*
- If we don’t rob house *i*, we can rob house number *i + 1* no problem
- If only we had a procedure for calculating the best result starting at house *i*

Modeling recursively

- How do you know if you should rob a house?
- Why not just try both options?
- We want the best outcome, so take the maximum

Example solution (math)

$$M(i) = \mathbf{max} \begin{cases} a_i + M(i+2) \\ M(i+1) \end{cases}$$

```
static int A = [4,2,5,8];
static int M(int i) {
    if (i >= A.length) {
        return 0;
    }
    return Math.max(
        A[i] + M(i + 2),
        M(i + 1),
    );
}
```

Example solution (https://spruett.me/blog/static/code/robber.java.html)

Indexing

- Equivalently, you can work backwards
  - Start at A[n - 1], and work with M(i - 2) and M(i - 1)

Anyone guess the runtime?

- $T(N) = T(N - 1) + T(N - 2) + O(1)$

Runtime

- Grows like the Fibonacci sequence
- O(pretty bad)
- Next lecture we'll magically transform this into O(N)
  - You might see the solution already
  - I apologize for leaving you with a terribly inefficient algorithm, but the hardest part of this problem is the recurrence

Recursion

- Sometimes the hardest part is thinking recursively, even if your final answer doesn’t involve recursion

Places to practice

- https://www.hackerrank.com/feed
  - Interview style questions
- https://open.kattis.com/problems
  - Competitive programming style problems (some from past regionals)
- http://codeforces.com/
  - Weekly contests, archive of past problems
- https://projecteuler.net/
  - Combination of math and programming

Problems for today

- Trying something new, we'll just do some project Euler examples
  - https://projecteuler.net/problem=14
  - https://projecteuler.net/problem=18

Input for Problem 18

- You can save the file, or paste in to terminal and use Scanner if you wish
- Hard-coded array:
  - https://spruett.me/blog/static/code/pyramid-input.java.html

Recursion