Basic for loop

- Print 0 through N

```
1
2
```

```
for i = 0 to N
    print i
```

- Runtime?
  - O(N)

Nested loops

- Print pairs

```
1   for i = 0 to N
2       for j = 0 to N:
3           print i + ", " + j
```

- Runtime?
  - $O(N^2)$

Drop constants, 2N is the same as N (does it scale linearly?)

More loops

- Print evens

```
1   for i = 0 to N
2       if i % 2 == 0:
3           print i
```

- Runtime?
  - O(N)

Two loops

- Print evens, then odds

```
1  for i = 0 to N
2      if i % 2 == 0:
3          print i
4
5  for i = 0 to N
6      if i % 2 != 0:
7          print i
```

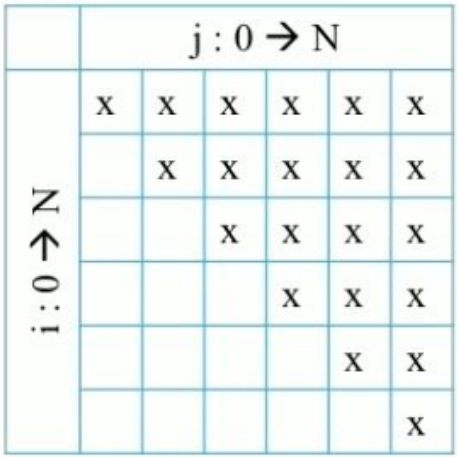
- Runtime?
  - O(N)

Basic for loop

- Print ordered pairs

```
1   for i = 0 to N
2       for j = i to N:
3           print i + ", " + j
```

- Runtime?
  - $O(N^2)$



Basic for loop

- Print ordered pairs

```
1   for i = 0 to A.length
2       for j = 0 to B.length
3           print A[i] + ", " + B[j]
```

- Runtime?
  - O(A * B)

Okay now things are getting tougher!

```
1  int last_death = Integer.Min
2
3  /* step 1: get last death */
4  for (Person person : people) {
5      last_death = max(last_death, person.death)
6  }
7
8  /* step 2: increment counter for each year som
9  int[] counter = new int[last_death]
10 for (Person person : people) {
11     for (int year = person.birth; year < perso
12         counter[year]++;
13     }
14 }
15
16 /* step 3: find population peak */
17 int highest_population = 0
18 for (int year = 0; year < counter.length; year+
19     highest_population = max(highest_population
20 }
```

- Step 1: O(P)
  - P = number of people
- Step 2: O(P * L)
  - L = max life span
- Step 3: O(Y)
  - Y = total # years

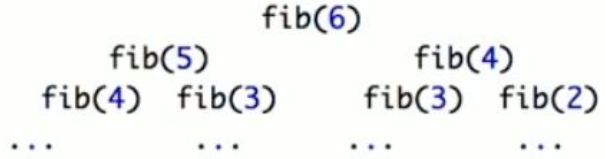
- O(P + P * L + Y) → O(P * L + Y)

Eeek. Recursion?

- Fibonacci

```
1 int fib(int n) {
2     if (n == 0 || n == 1) {
3         return 1;
4     } else {
5         return fib(n - 1) + fib(n - 2);
6     }
7 }
```

- Runtime?



- Height of K
- Each level doubles # nodes
- → $O(2^k)$ ***

Actually slightly less for complex math reasons.

Eeek. Recursion?

- Fibonacci, part 2

```
1 int fib(int n, int[] memo) {
2     if (n == 0 || n == 1) {
3         return 1;
4     } else if (memo[n] == 0)
5         memo[n] = fib(n - 1) + fib(n - 2);
6     }
7     return memo[n];
8 }
```

- Runtime?

```
                fib(6)
          fib(5)          x
      fib(4)   x
  fib(3)
...
```

- Height of K
- Each level has ONE node
- → O(k)