
Fibonacci

Pure recursion, with time $O(2^n)$:

```
int fib(int n) {
    if (n==0) return 1;
    if (n==1) return 1;
    return fib(n-1) + fib(n-2);
}
```

Two ways follow to make fib of $O(n)$ asymptotic time complexity, the first still using recursion and the second one using iteration:

Fibonacci with memoizing

```
/** Called dynamic programming because top-down breaking of
larger problem into smaller problems that fit together nicely.
Storing results for future use in a data structure is called
memoization.
*/

public Hashtable<Integer,Integer> t =
    new Hashtable<Integer, Integer>;
t.put(0,1); t.put(1,1);           // autoboxing of int's

int fib(int n){
    if (t.get(n) == null)        // null doesn't need .equals
        t.put(n, fib(n-1) + fib(n-2));
    return t.get(n);
}
```

Fibonacci iteratively

```
public int fib(int n) {
    int [] a = new int[n+1];      //n+1 is size of array
    a[0] = 1; a[1] = 1;
    for(i=2; i<=n; i++) {        //Is n tested before
        a[i] = a[i-1] + a[i-2];  // first execution? Yes!
    }
    return a[n];
}
```