Bucknell University
Computer Science

CSCI 311 - Data Structures

Recursion and
Dynamic Programming

Recursion

Recursion

Factorial:

$$fact(n) = n * fact(n-1)$$

$$fact(0) = 1$$

```
int fact(int n) {
      if (n == 0) return 1;
      else return n * fact(n-1);
}
```

Recursion

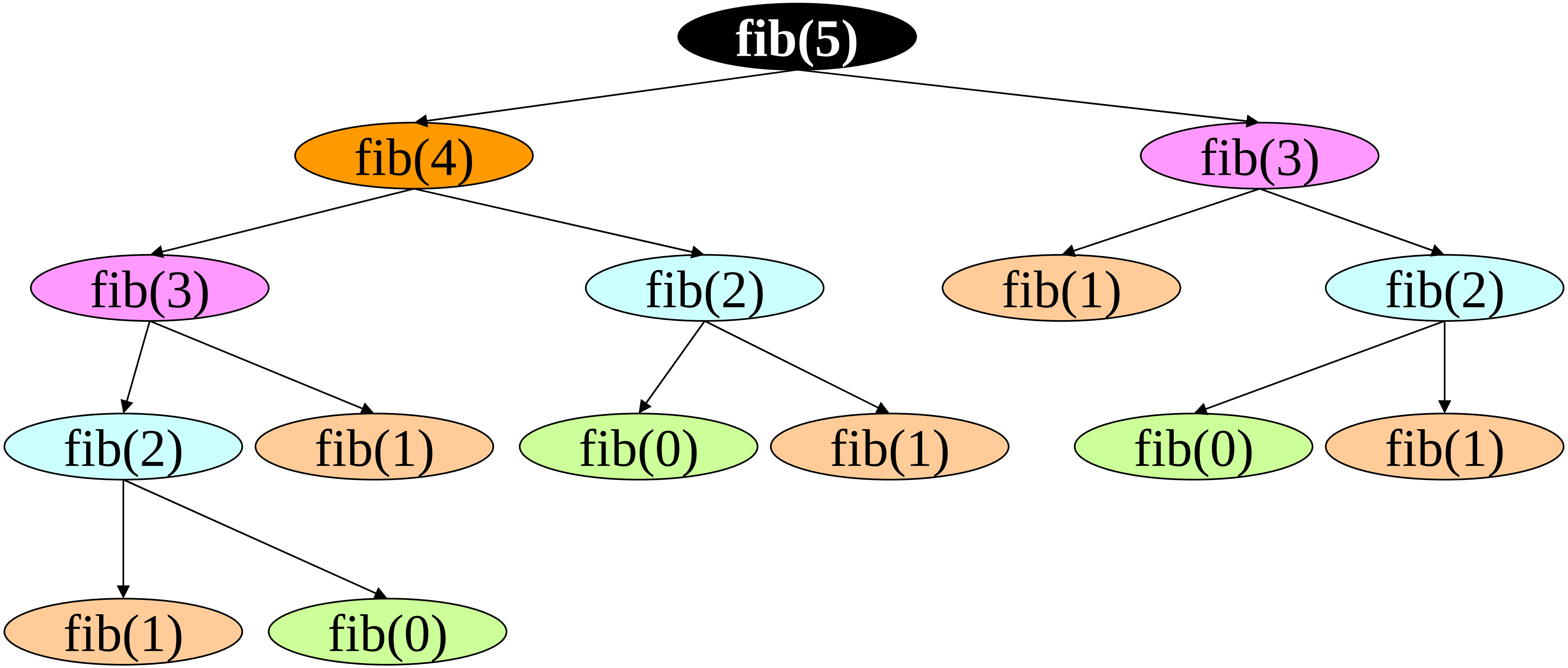
Fibonacci:

$$fib(n) = fib(n-1) + fib(n-2)$$

$$fib(0) = 1, \quad fib(1) = 1$$

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

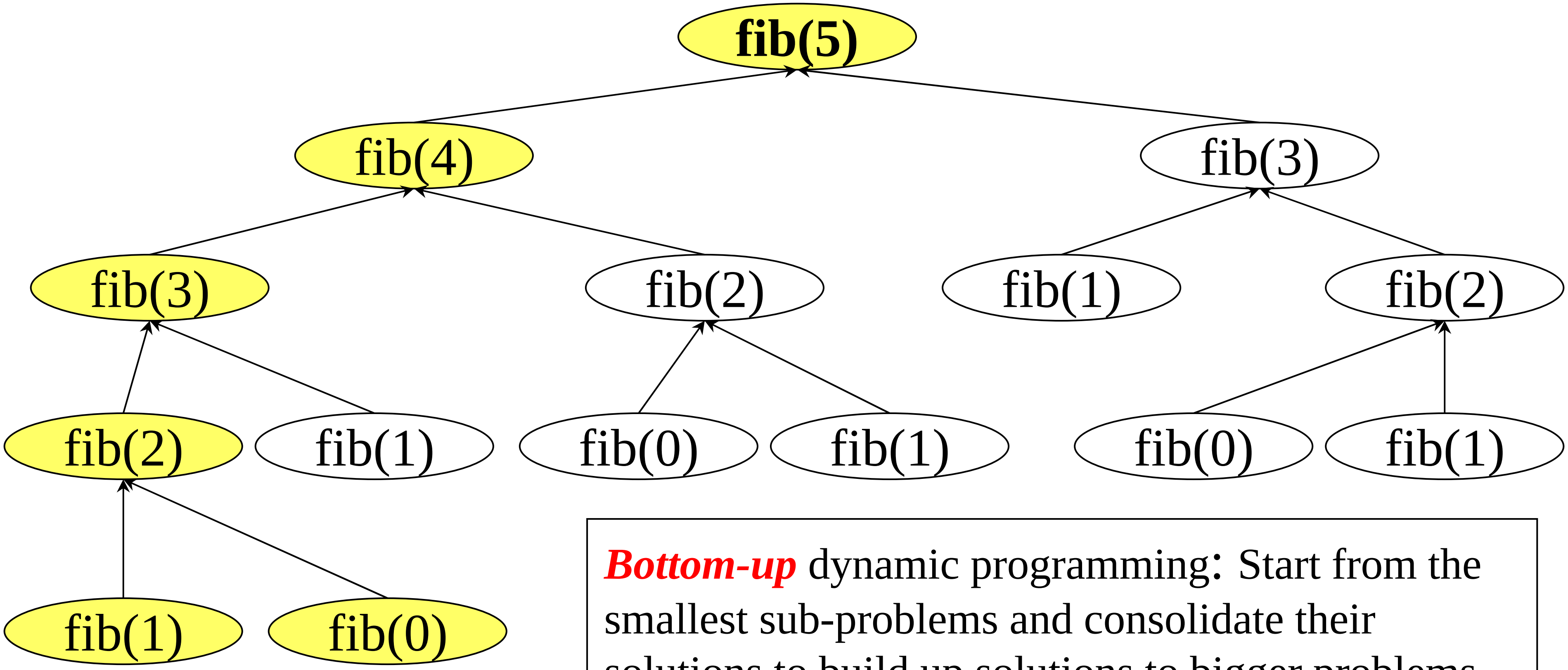
Fibonacci Numbers



Question: How can this be accomplished without repeating the same work over and over?

Dynamic Programming

Fibonacci Numbers



Bottom-up dynamic programming: Start from the smallest sub-problems and consolidate their solutions to build up solutions to bigger problems.

0	1	2	3	4	5	6	7	8	9
1	1	2	3	5	8	0	0	0	0

Fibonacci Numbers

$F_N = \phi^N$ (exponential - time)

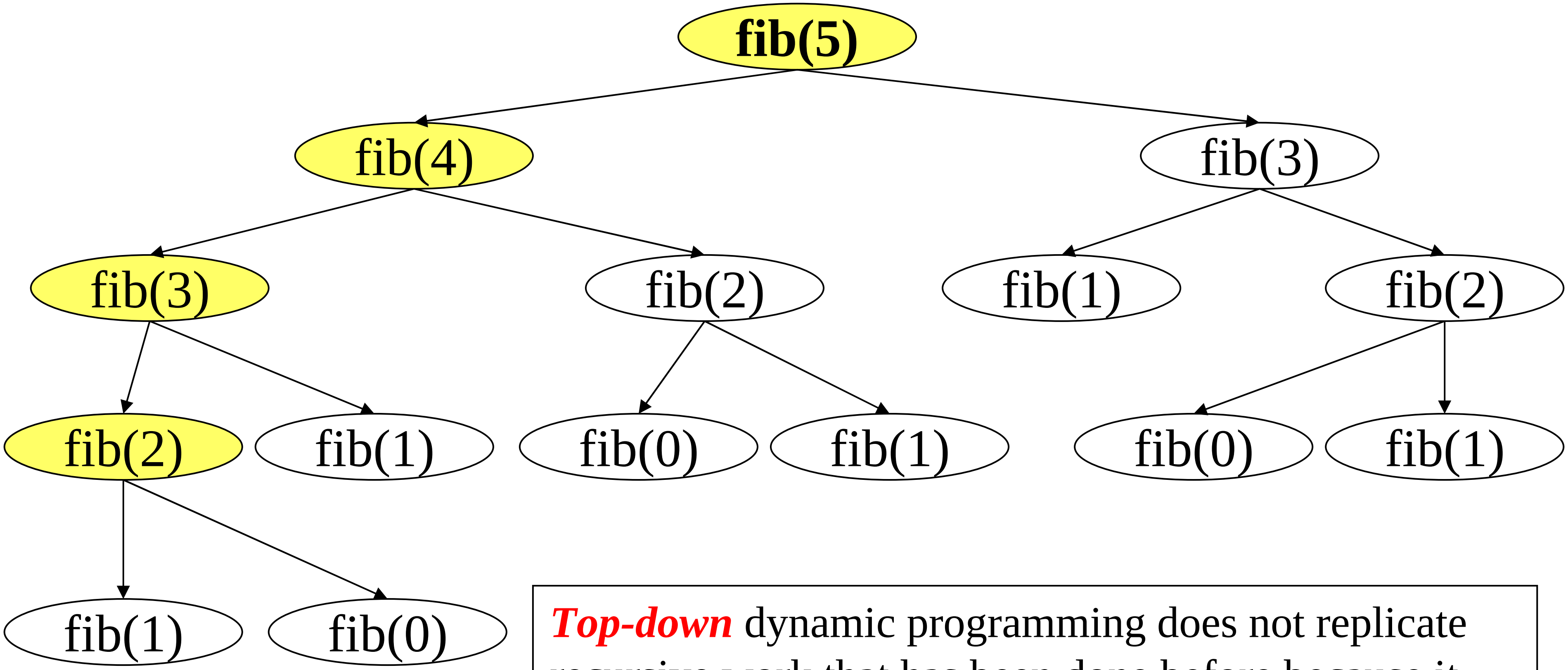
```
int F(int i) {
   if (i <= 1) return 1;
   return F(i-1) + F(i-1);
}
```

```
int F(int i) {
   static int knownF[maxN];

   if (knownF[i] != 0)
      return knownF[i];

   int t = i;
   if (i <= 1) return 1;
   // i > 1 and F(i) not known
   t = F(i-1) + F(i-2);
   knownF[i] = t;
   return knownF[i];
}
```

Top-down Dynamic Programming
or
Memoization

Question: What is the run time complexity of this alternative?

Fibonacci Numbers



Top-down dynamic programming does not replicate recursive work that has been done before because it can remember the results generated at each step.

0	1	2	3	4	5	6	7	8	9
1	1	2	3	5	8	0	0	0	0