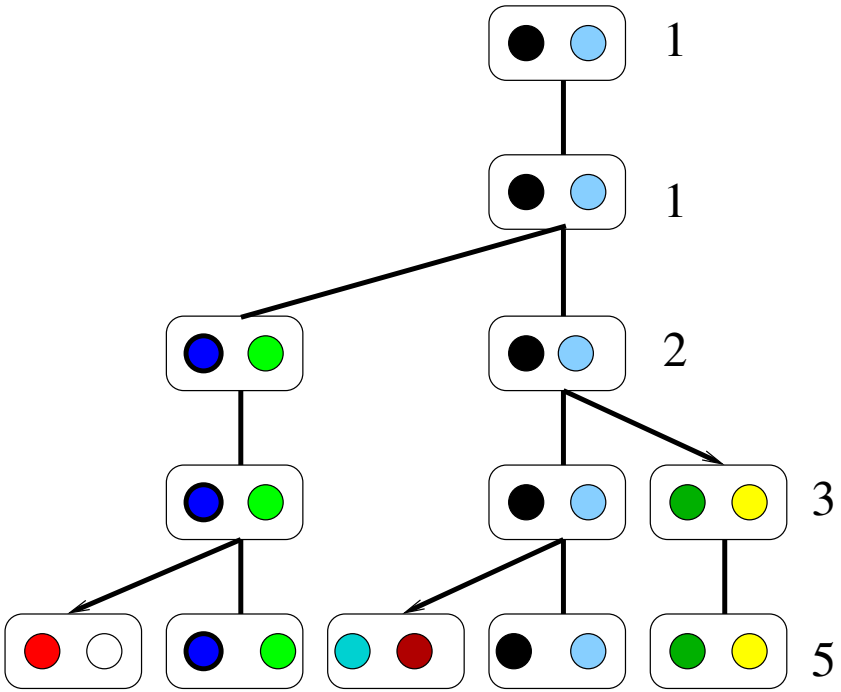
Recursive Functions

Fibonacci Numbers



- Growth is exponential: possible to find $r > 1$ st. $f_n \geq r^{n-2}$.
- Let $r = \frac{1+\sqrt{5}}{2} = 1.62$, so that $r^2 = r + 1$
- We need to prove that $f_n \geq r^{n-2}$.

Recursive Functions

Fibonacci Numbers

- **Basis**: $P(1)$ is true, $f_1 = 1$, since $r^{1-2} = r^{-1} \leq 1$.
  Note, also that as $r^{2-2} = 1 = f_2$, $P(2)$ holds.
- **Induction hypothesis**: for a fixed $n > 2$, $P(1), P(2) \ldots P(n)$ are all true.
- **Induction step**: consider $f_{n+1} = f_{n-1} + f_{n-2}$.
  - Using hypothesis $f_n \leq r^{n-2} + r^{n-3}$
  - So, $f_n \leq r^{n-3}(r+1) = r^{n-3}.r^2 = r^{n-1}$.
- Hence $P(n+1)$ is true.

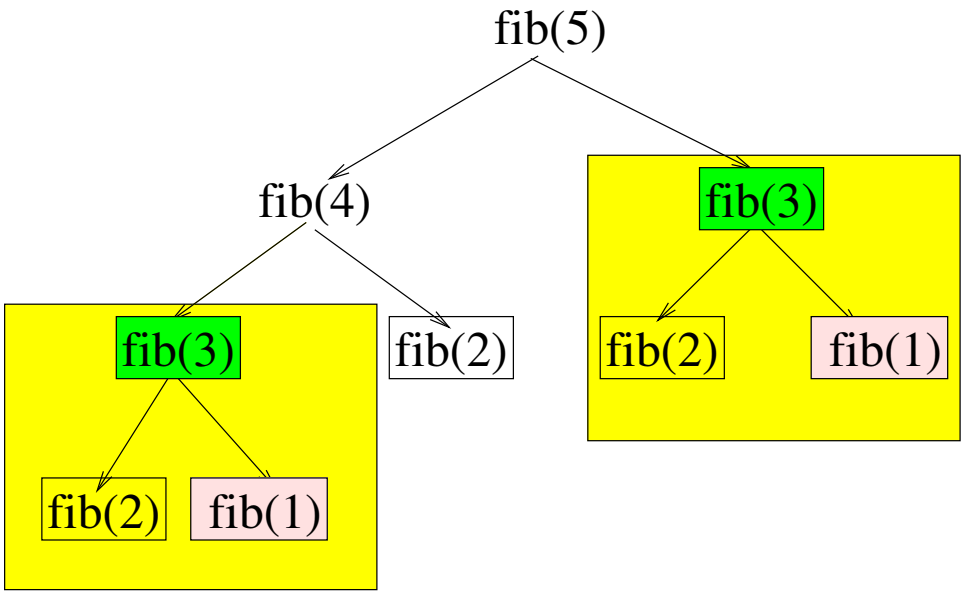
Examples of Recursive Functions

Fibonacci Function

```
#include <stdio.h>
int fib(int n) {

    if (n == 1) return 1;
    if (n == 2) return 1;

    return fib(n-1) + fib(n-2);
}

int main() {
    int n;
    printf("Enter n : ");
    scanf("%d", &n);
    printf("%dth fibonacci number = %d\", n, fib(n));
}
```

Examples of Recursive Functions

Fibonacci Function



Examples of Recursive Functions

Efficient Computation of Fibonacci

- To make it more efficient the strategy would be
  - Keep track of both current and previous fibonacci numbers
  - How many are to be computed?
  - Initially $f(1)$ and $f(2)$ are known and $n-2$ other numbers to be computed.

$$\text{fibo(cur, prev, } n) = \begin{cases} 1, \text{ if } n <= 2 \\ \text{fibo(cur+prev, cur, } n\text{-1)} \end{cases}$$

- So if $n = 6$ then 4 values are to be computed.

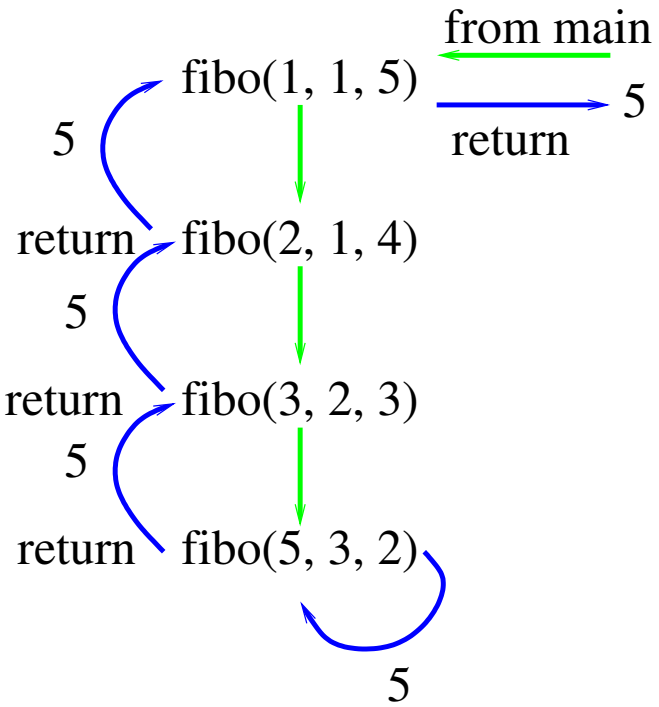
Examples of Recursive Functions

Efficient Computation of Fibonacci

```
#include <stdio.h>
int fibo(int cur, int prev, int n) {
    if (n-2 <= 0)
      return cur;
    return fibo(cur + prev, cur, n-1);
}

int main() {
   int n;

   printf("Enter a +ve n : ");
   scanf("%d", &n);
   printf("fib(%d) = %d\n", n, fibo(1, 1,n));
}
```

Examples of Recursive Functions

Fibonacci Function



Examples of Recursive Functions

Power Function

- Recursive specification of x^n is as follow:

$$pow(x,n) = \begin{cases} 1, \text{ if } n = 0 \\ x * \text{square}(pow(x, n/2)), \text{ if odd}(n) \\ \text{square}(pow(x, n/2)), \text{ otherwise} \end{cases}$$

Examples of Recursive Functions

Power Function

```
#include <stdio.h>
int power(int x, int n) {
    int t;

    if (n == 0)
        return 1;

    t = power(x, n/2);
    if (n%2 != 0)
        return x * t * t;
    return t * t;
}

int main() {
    int n, x;
    printf("Enter x and n: ");
    scanf("%d %d", &x, &n);
    printf("power(%d, %d) = %d\n", x, n, power(x,n));
}
```