Sep 04

2^n sets of dispatcher locations to check

- assume your computer checks $500,000 = 5 \cdot 10^5$ sets per second (this is unrealistic, since the number should depend on n...) (the length of time needed for a check would realistically increase with the size of the network.)
- assume a solution is found after checking $\frac{1}{8}$ of all sets ($= 2^{n-3}$ sets)

$\Rightarrow$ How long does your computer need, for various values of n?

$n=10 \longrightarrow 2^{n-3} = 2^7 = 128 \rightsquigarrow$ fraction of a millisecond

$n=20 \longrightarrow 2^{n-3} = 2^{17} \approx 130,000 \rightsquigarrow \approx \frac{1}{4}$ second

$n=50 \longrightarrow 2^{n-3} = 2^{47} \approx 1.41 \cdot 10^{14} \approx 5 \cdot 10^5 \cdot 10^8 \cdot 2 \rightsquigarrow \approx 7$ years

$n=100 \longrightarrow 2^{n-3} = 2^{97} \approx 1.58 \cdot 10^{29} \approx 5 \cdot 10^5 \; 10^{23} \cdot 2 \rightsquigarrow \approx$ 7 quadrillion years

EXAMPLE 2.

computing the Fibonacci numbers

$F(n)$ n-th Fib. number

1, 1, 2, 3, 5, 8, 13, ...

$F(1) = 1$
$F(2) = 1$
$F(n) = F(n-1) + F(n-2)$ for $n > 2$.

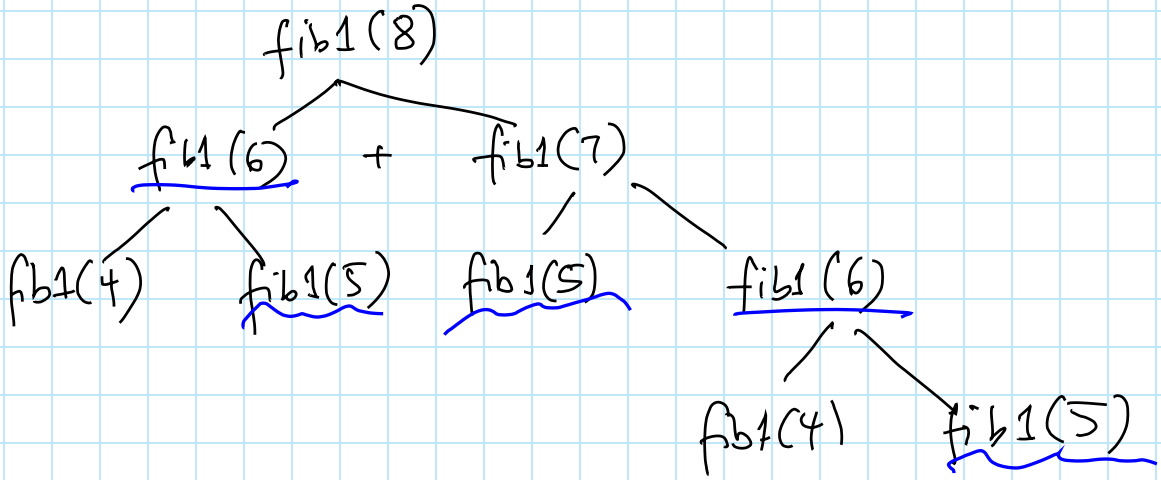
$F(10) = 55$
$F(50) = 12,586,269,025$
$F(92) > 7$ quintillion

Program 1

```
long long fib1 (int n)
{
    if (n==1 || n==2)
        return 1;
    else
        return fib1(n-1) + fib1(n-2);
}
```

fib1(50) takes a couple of minutes

fib1(92) got hung up



wasteful !

easier / less wasteful solution?

program 2.

```
long long fib2 (int n)
{
    long long value = 1;
    long long first = 1;
    long long second = 1;
    for (int i = 3; i <= n, i++)
    {
        value = first + second;
        first = second;
        second = value;
    }
    return value;
}
```

fib2(50)
fib2(92)
computed in virtually no time.

⇒ efficiency of algorithms is a central concern!

space / time trade-off:

often a more time-efficient alg is less memory-efficient

⇒ we analyze algorithms w.r.t.
- the time their computations consume
- the memory (space) their computations consume

tentative course outline

1. Algorithm analysis [book Ch. 2]
2. Priority queues [Ch. 6]
3. Sorting [Ch. 7]

5. Complexity and unsolvability [Ch. 9.7 + add-ons]

6. Algorithm design techniques [Ch. 10]

1. ALGORITHM ANALYSIS

goal: analyze performance of algorithms/programs wrt

- running time
- space

running time of a program depends on

- compiler
- OS
- machine instructions
- computer
- programming language

} ignore

- underlying algorithm
- the input to the program

⇝ analyze algorithms rather than programs!