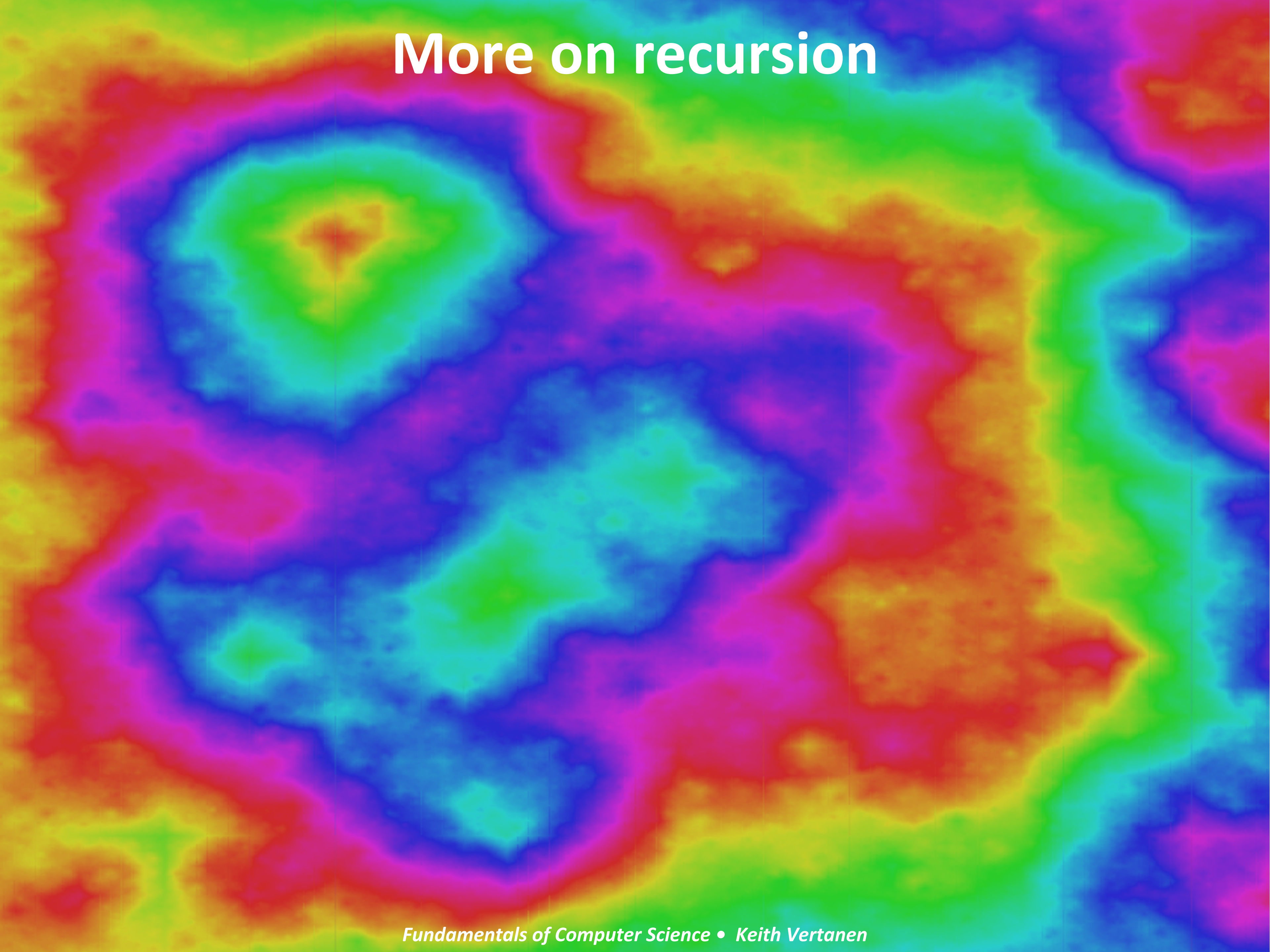
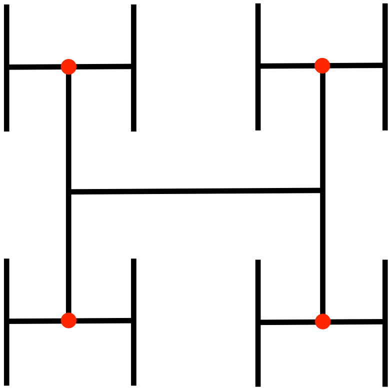
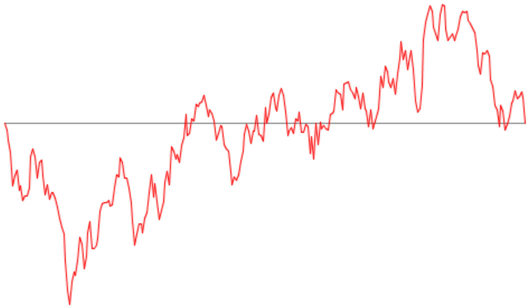
More on recursion



Overview

- Recursion
  - A method calling itself
    - A new way of thinking about a problem
    - A powerful programming paradigm
- Examples:
  - Last time:
    - Factorial, binary search, H-tree, Fibonacci
  - Today:
    - Greatest Common Divisor (GCD)
    - Brownian Motion
    - Sorting things





Greatest Common Divisor

- GCD
  - Find largest integer d that evenly divides p and q
  - e.g. gcd(4032, 1272) = 24
    - $4032 = 2^6 \text{ x } 3^2 \text{ x } 7^1$
    - $1272 = 2^3 \text{ x } 3^1 \text{ x } 53^1$
    - $\text{gcd} = 2^3 \text{ x } 3^1 = 24$
- Applications
  - Simplify fractions:

$$1272/4032 = 53/168$$

  - RSA cryptography

Simple GCD algorithm

- GCD
  - Find largest integer d that evenly divides p and q
    - Assume p > q, p and q are positive integers
- Simple algorithm:
  - Set i = q
  - See if i evenly divides both p and q
    - If yes, i is the GCD
  - Decrement i
  - Repeat until i = 1

```
public static long gcd(long p, long q)
{
   for (long i = q; i > 1; i--)
   {
      if ((p % i == 0) && (q % i == 0))
         return i;
   }
   return 1;
}
```

Euclid's GCD algorithm

- GCD
  - Find largest integer d that evenly divides p and q
    - Assume p > q, p and q are positive integers
- Euclid's algorithm (300 BC)

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q,\ p \% q) & \text{otherwise} \end{cases}$$

base case

reduction step, converges to base case

```
gcd(4032, 1272) =
               = gcd(1272, 216)
               = gcd(216, 192)
               = gcd(192, 24)
               = gcd(24, 0)
               =  24
```

```
4032 = 3 x 1272 + 216
1272 = 5 x  216 + 192
 216 = 1 x  192 +  24
 192 = 8 x   24 +   0
```

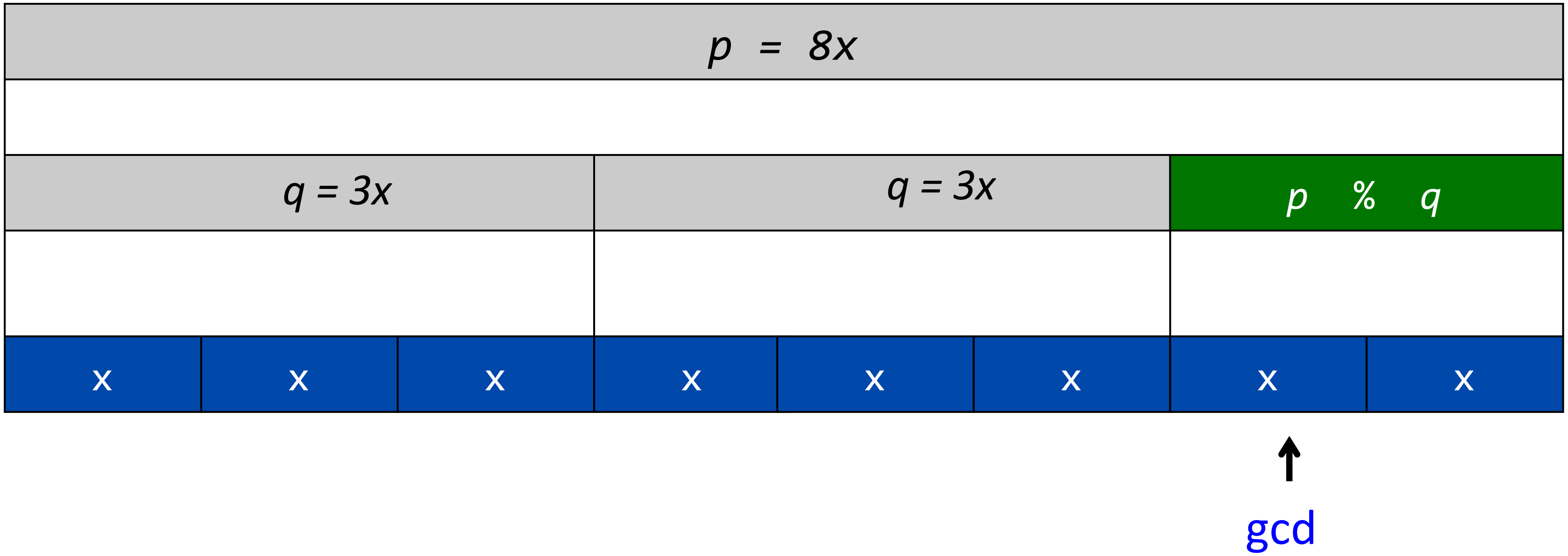
Greatest Common Divisor

- GCD
  - Find largest integer d that evenly divides p and q
    - Assume p > q, p and q are positive integers

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q, p \% q) & \text{otherwise} \end{cases}$$

base case

reduction step, converges to base case



Greatest Common Divisor

- GCD
  - Find largest integer d that evenly divides p and q
    - Assume p > q, p and q are positive integers

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q,\ p \ \%\ q) & \text{otherwise} \end{cases}$$

← base case

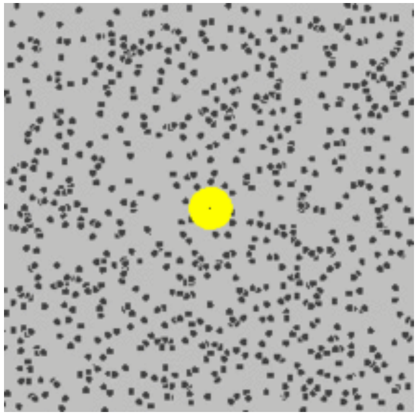
← reduction step, converges to base case

```
public static long gcd(long p, long q)
{
   if (q == 0)
      return p;
   else
      return gcd(q, p % q);
}
```

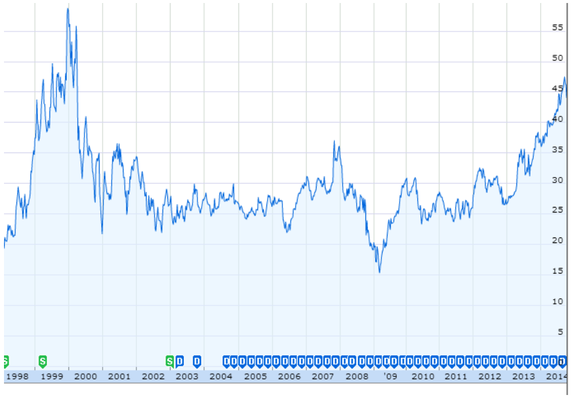
base case

reduction step

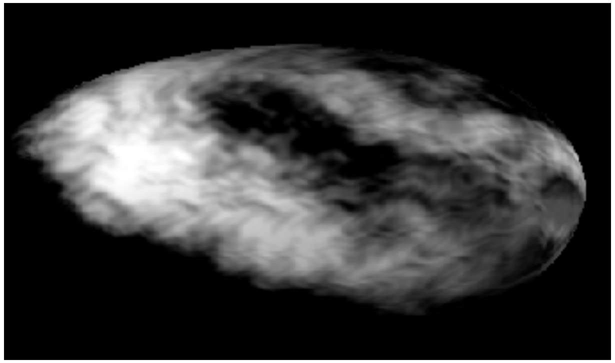
Brownian motion

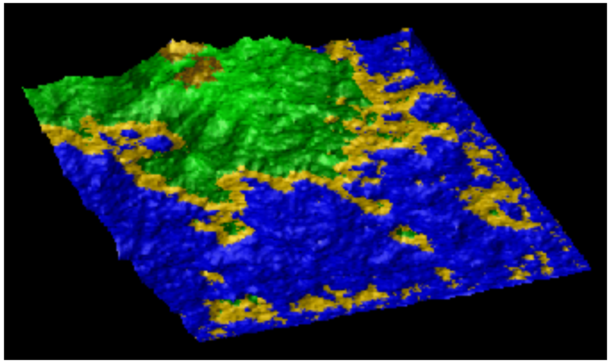


- Models many natural and artificial phenomenon
  - Motion of pollen grains in water
  - Price of stocks



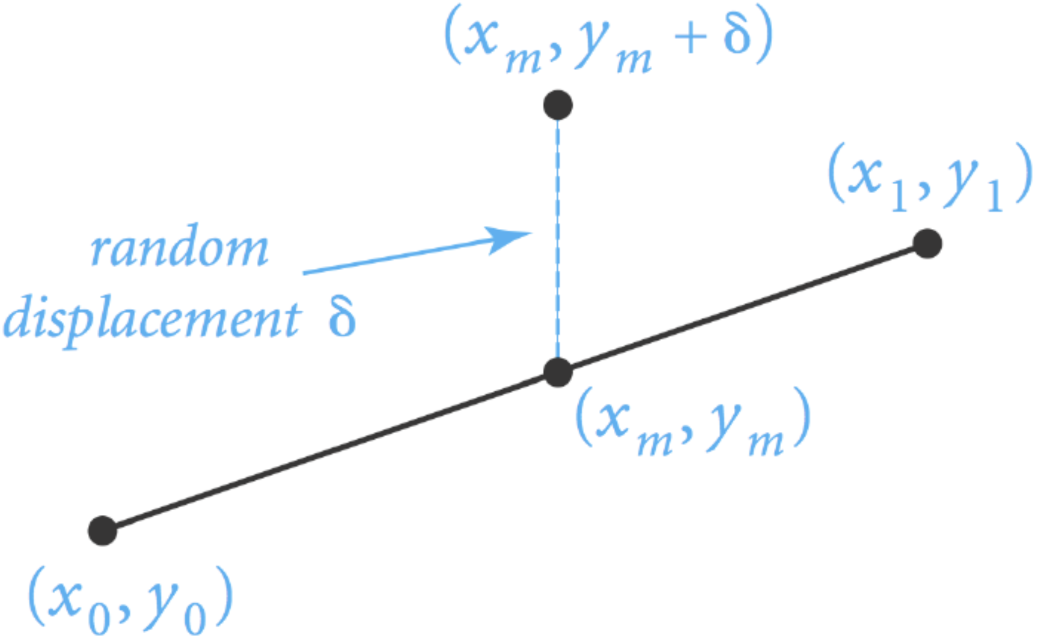
  - Rugged shapes of mountains and clouds



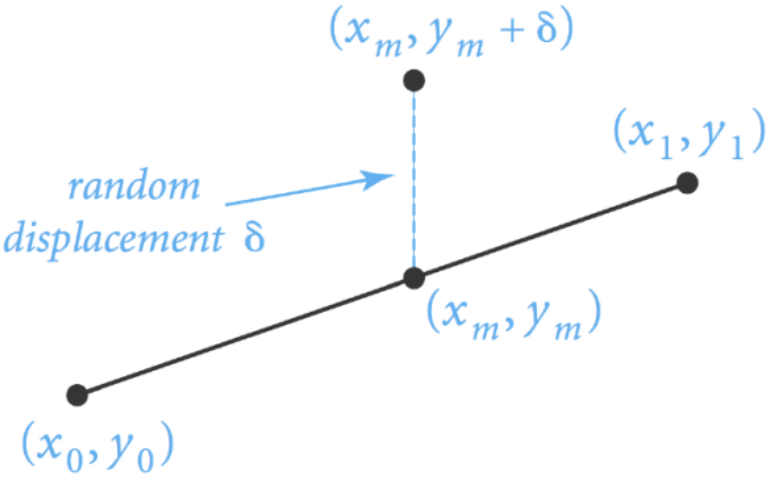


Simulating Brownian Motion

- Midpoint displacement method:
  - Track interval (x_0, y_0) to (x_1, y_1)
  - Choose δ displacement randomly from Gaussian
  - Divide in half, $x_m = (x_0+x_1)/2$ and $y_m = (y_0+y_1)/2 + \delta$
  - Recur on the left and right intervals



Recursive midpoint displacement algorithm



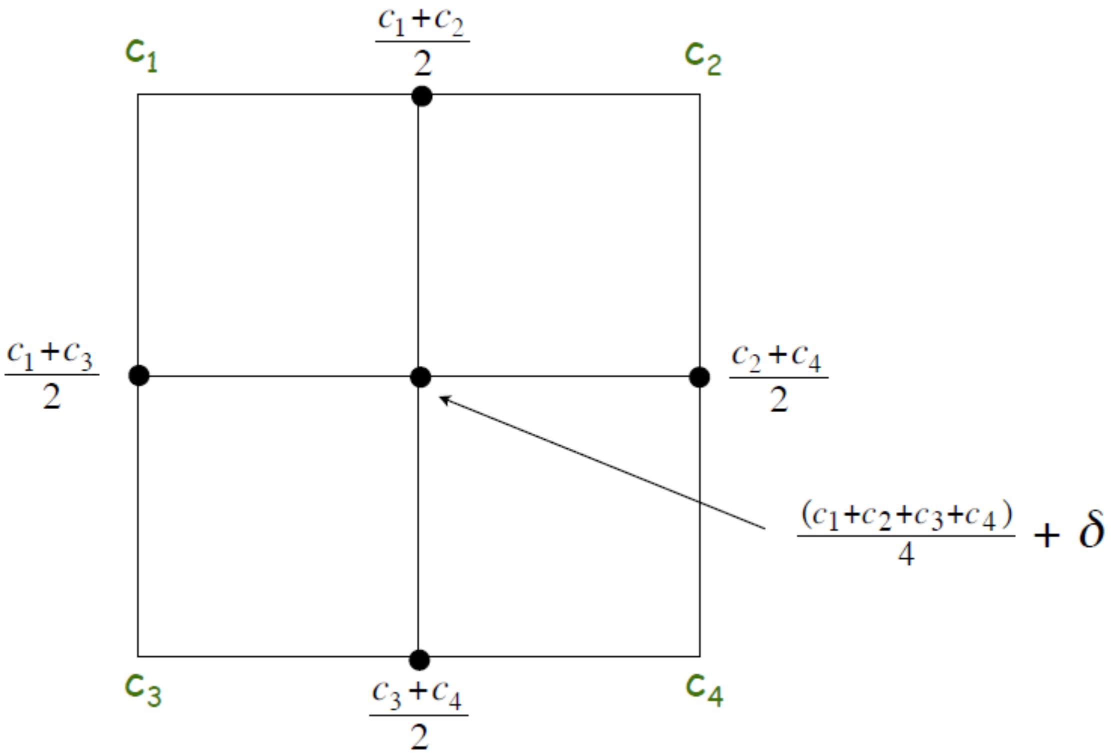
```java
void curve(double x0, double y0, double x1, double y1, double var)
{
   if (x1 - x0 < .005)
   {
      StdDraw.line(x0, y0, x1, y1);
      return;
   }

   double xm = (x0 + x1) / 2.0;
   double ym = (y0 + y1) / 2.0;

   ym = ym + StdRandom.gaussian(0, Math.sqrt(var));

   curve(x0, y0, xm, ym, var / 2.0);
   curve(xm, ym, x1, y1, var / 2.0);
}
```

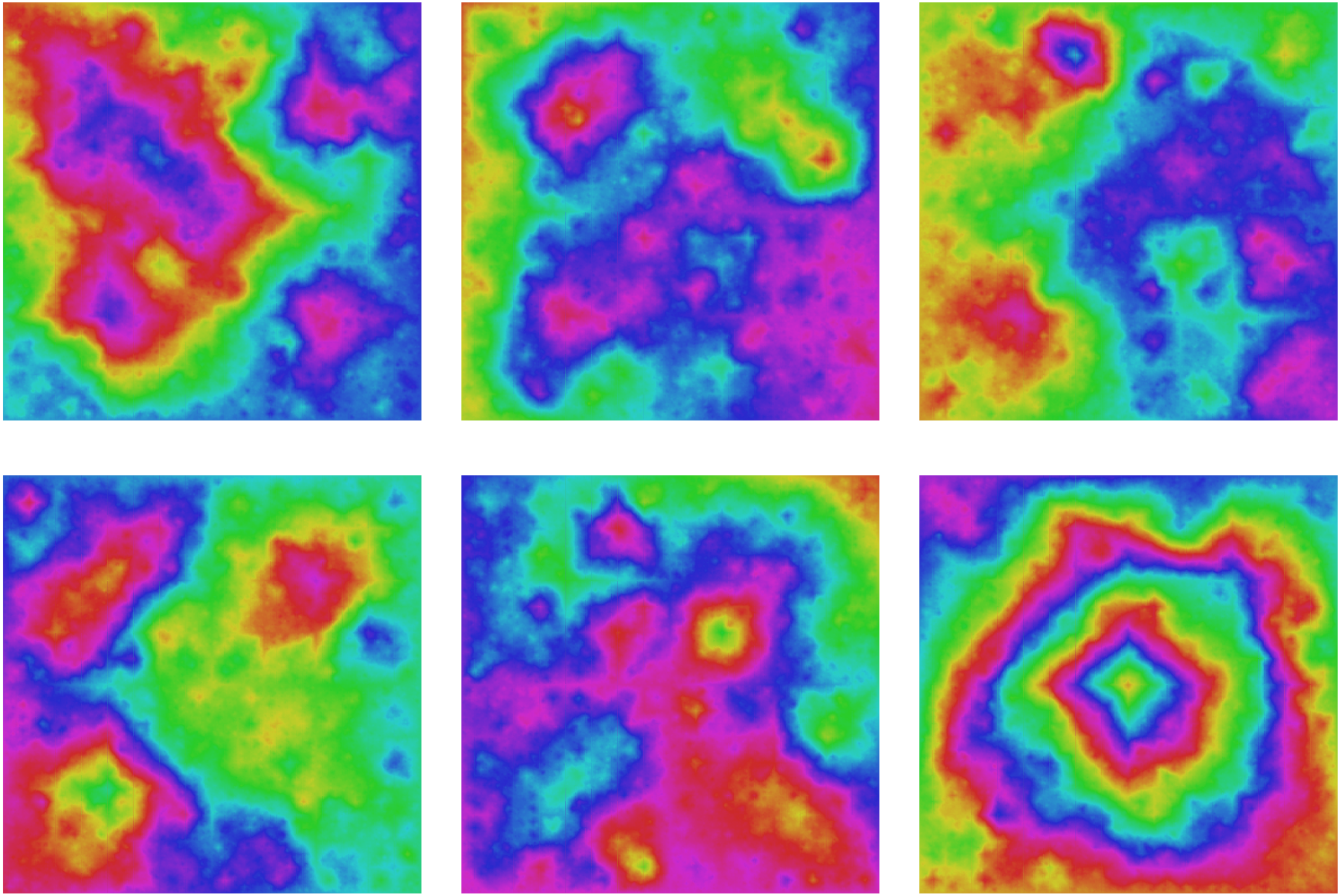
base case

reduction step

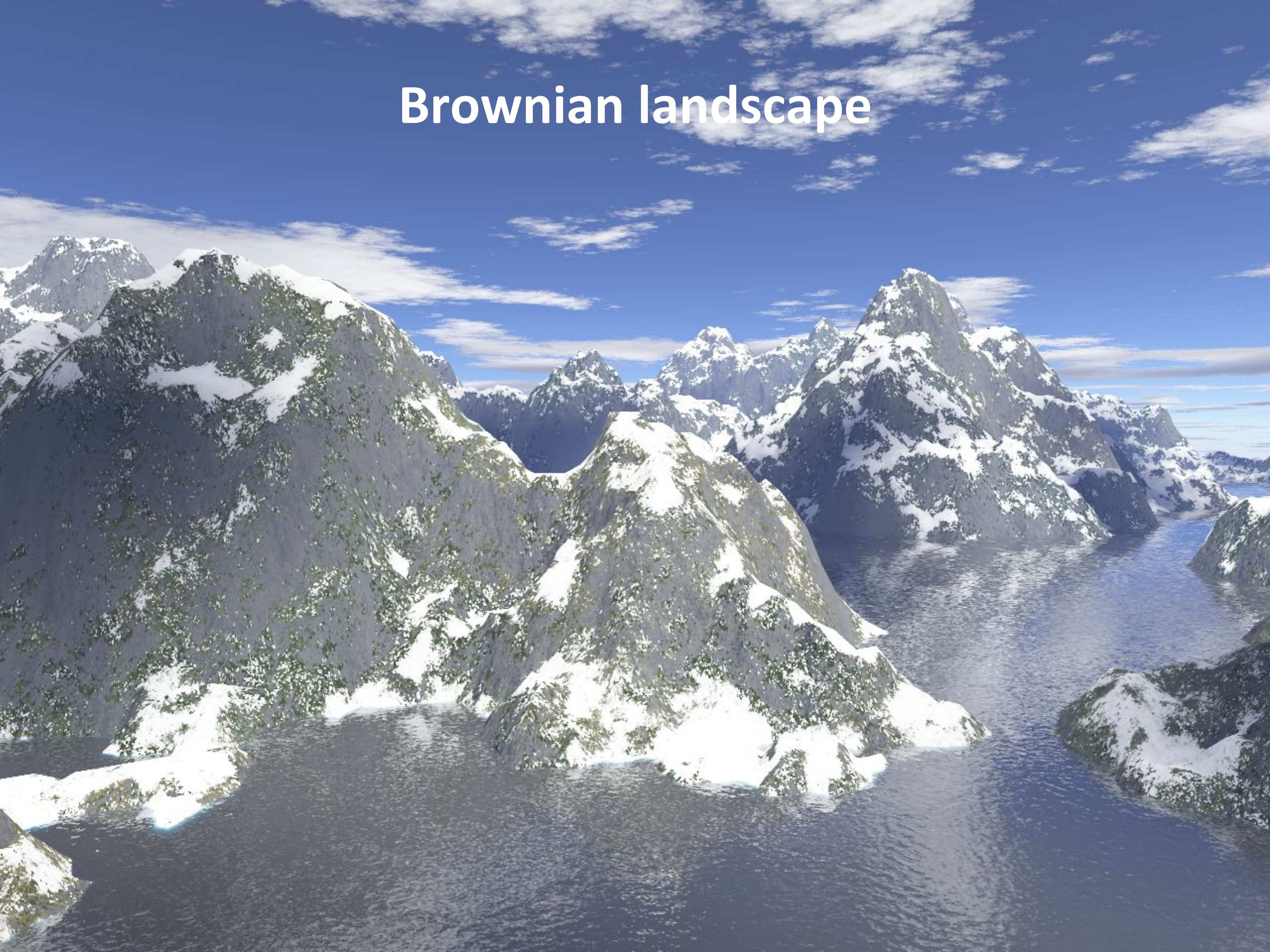
Plasma cloud

- Same idea, but in 2D
  - Each corner of square has some greyscale value
  - Divide into four sub-squares
  - New corners: avg of original corners, or all 4 + random
  - Recur on four sub-squares





Brownian landscape



Divide and conquer

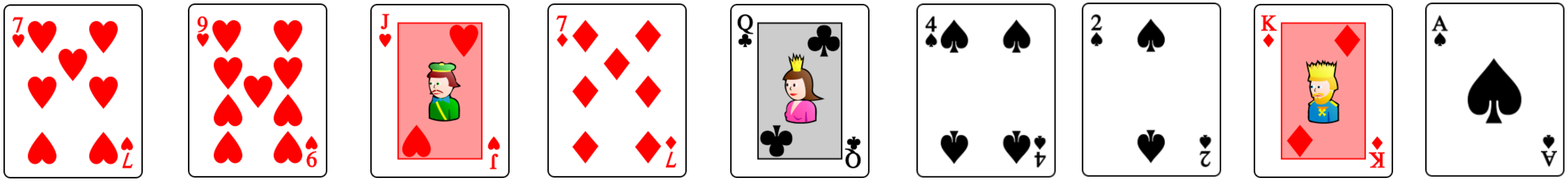
- Divide and conquer paradigm
  - Break big problem into small sub-problems
  - Solve sub-problems recursively
  - Combine results

> "Divide et impera. Vendi, vidi, vici."
> *-Julius Caesar*

- Used to solve many important problems
  - Sorting things, mergesort: O(N log N)
  - Parsing programming languages
  - Discrete FFT, signal processing
  - Multiplying large numbers
  - Traversing multiply linked structures (stay tuned)

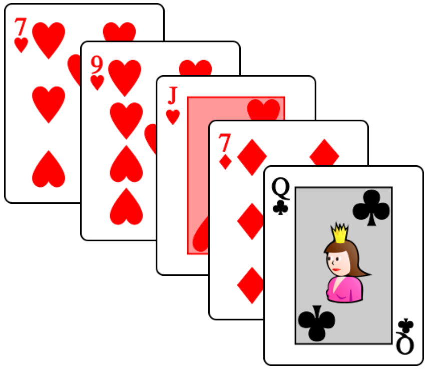
Divide and conquer: sorting

- Goal: Sort by number, ignore suit, aces high

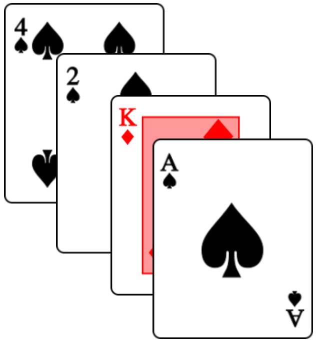


Approach

1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



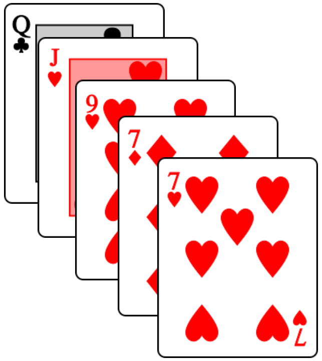
Unsorted pile #1



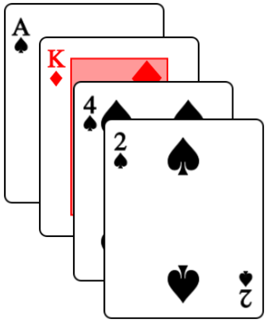
Unsorted pile #2

Approach

1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1



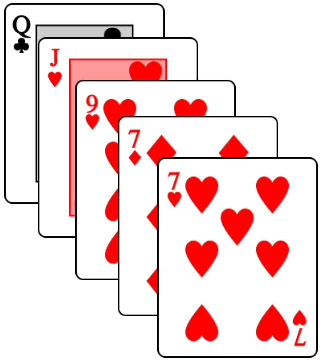
Sorted pile #2

Merging

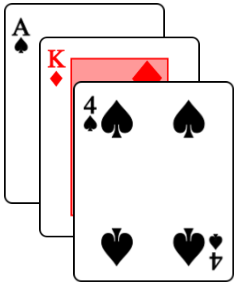
Take card from whichever pile has lowest card

Approach

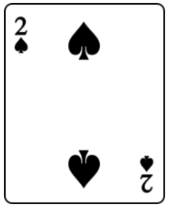
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1

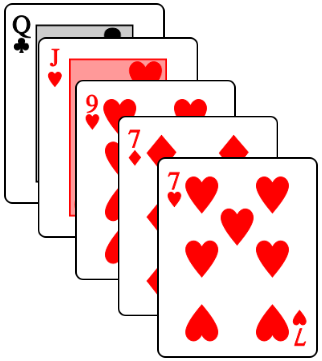


Sorted pile #2

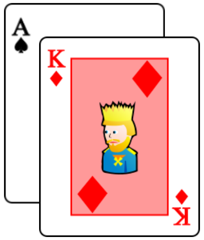


Approach

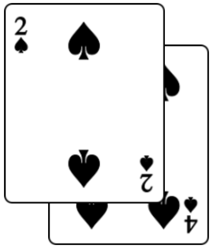
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1

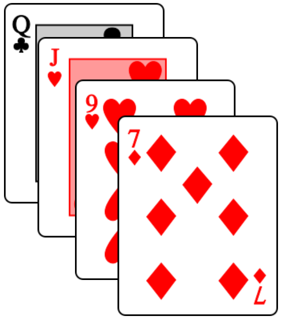


Sorted pile #2

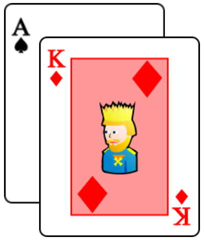


Approach

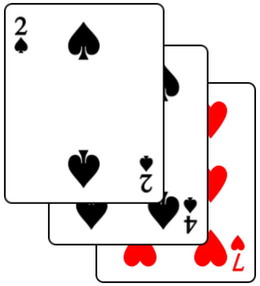
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1

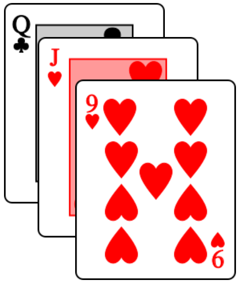


Sorted pile #2

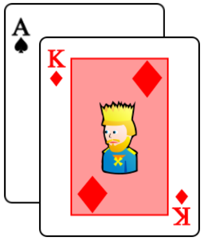


Approach

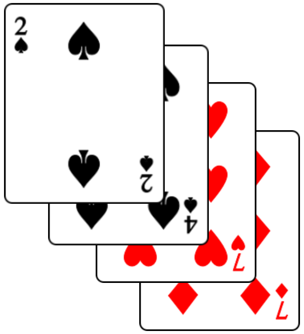
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1

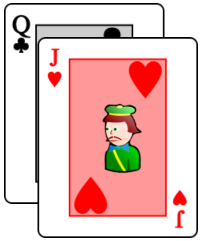


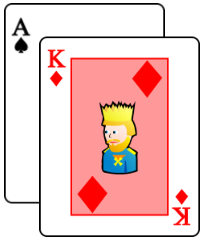
Sorted pile #2



Approach

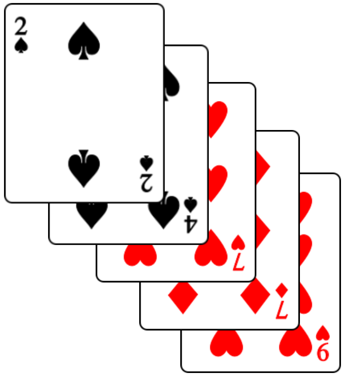
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together





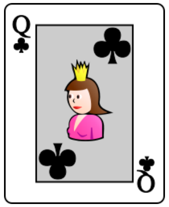
Sorted pile #1

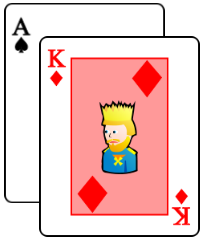
Sorted pile #2



Approach

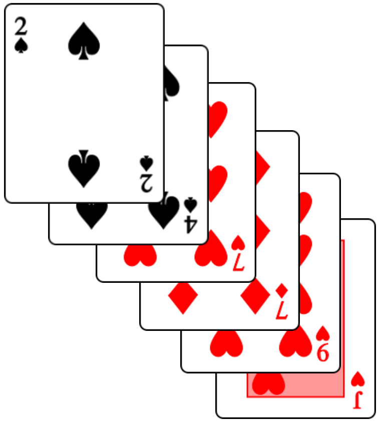
1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together





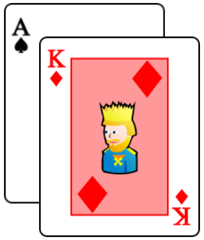
Sorted pile #1

Sorted pile #2



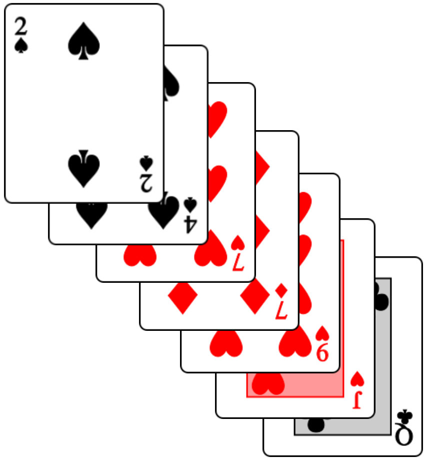
Approach

1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



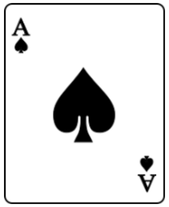
Sorted pile #1

Sorted pile #2



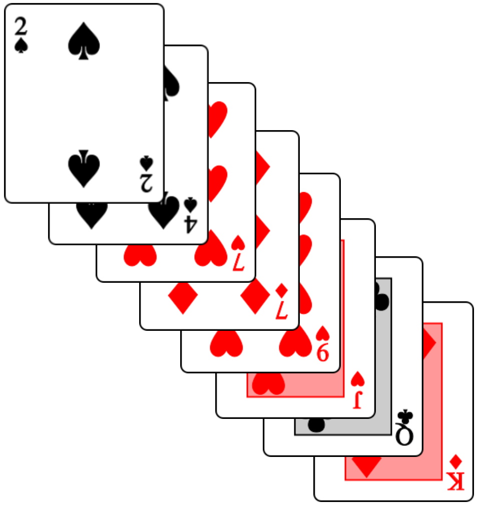
Approach

1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together



Sorted pile #1

Sorted pile #2

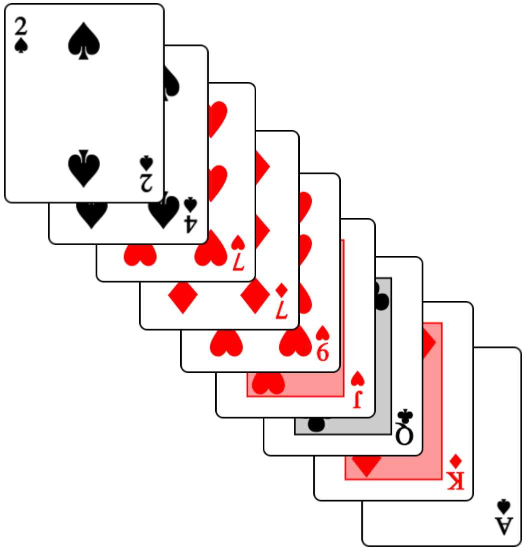


Approach

1) Split in half (or as close as possible)
2) Give each half to somebody to sort
3) Take two halves and merge together

Sorted pile #1

Sorted pile #2



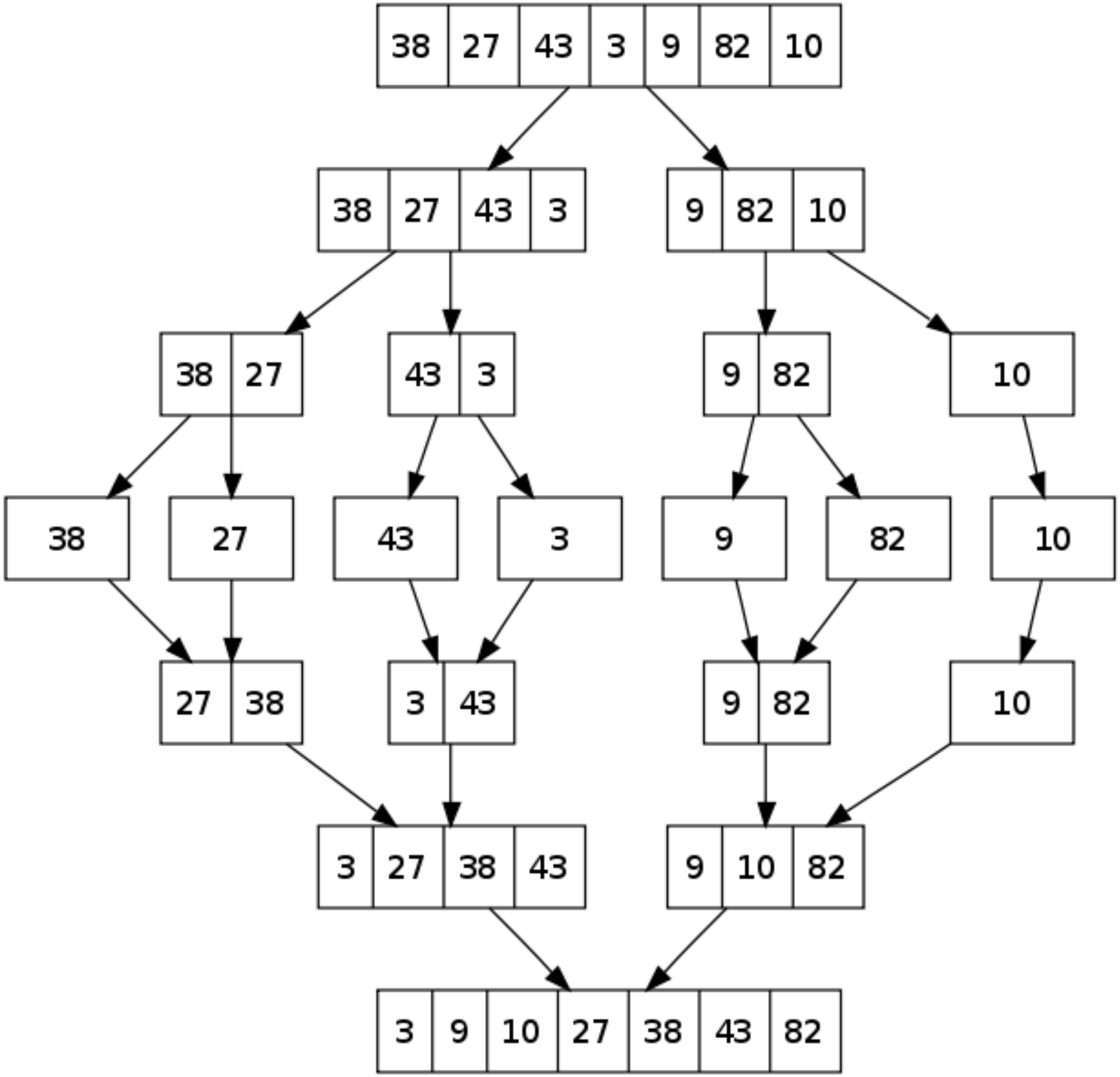
How many operations to do the merge?

Linear in the number of cards, O(N)

But how did pile 1 and 2 get sorted?

Recursively of course!

Split each pile into two halves, give to different people to sort.



How many split levels?

$O(\log_2 N)$

How many merge levels?

$O(\log_2 N)$

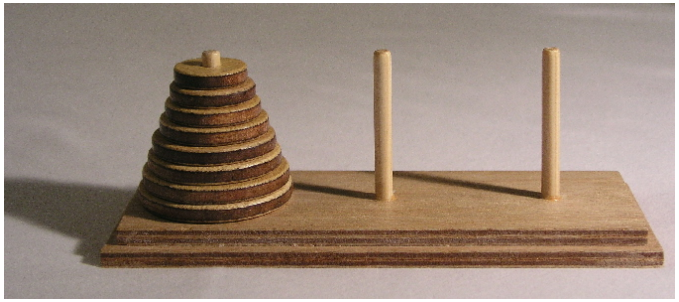
Operations per level?

$O(N)$

Total operations?

$O(N\log_2 N)$

Summary



- Recursion
  - A method calling itself:
    - Sometimes just once, e.g. binary search
    - Sometimes twice, e.g. mergesort
    - Sometimes multiple times, e.g. H-tree
  - All good recursion must come to an end:
    - Base case that does NOT call itself recursively
  - A powerful tool in computer science:
    - Allows elegant and easy to understand algorithms
    - (Once you get your head around it)