Αναδρομή – Ανάλυση Αλγορίθμων

Παράδειγμα: Υπολογισμός του παραγοντικού

- Ορισμός του n!

  n! = n x (n - 1) x … x 2 x 1
- Ο παραπάνω ορισμός μπορεί να γραφεί ως

$$n! = \begin{cases} 1 & \text{αν } n = 0 \\ n \times (n-1)! & \text{αλλιώς} \end{cases}$$

Παράδειγμα (συνέχ).

- Στον εναλλακτικό ορισμό, ο υπολογισμός του n! (σύνθετο πρόβλημα) ανάγεται στον υπολογισμό του (n-1)! (απλούστερο πρόβλημα).

Αναδρομή

- Στρατηγική επίλυσης προβλημάτων
- Τεχνική προγραμματισμού σύμφωνα με την οποία ένα σύνθετο πρόβλημα ανάγεται σε ένα απλούστερο **της ίδιας μορφής**.
- Είναι μια *παράξενη*, μη *διαισθητική τεχνική* η οποία στην αρχή δυσκολεύει τους προγραμματιστές
- Σε κάθε περίπτωση όπου χρησιμοποιείται αναδρομή, είναι δυνατόν, εναλλακτικά, να χρησιμοποιηθεί επανάληψη
- Η χρήση της αναδρομής προσφέρει, σε κάποιες περιπτώσεις, απλούστερες λύσεις.

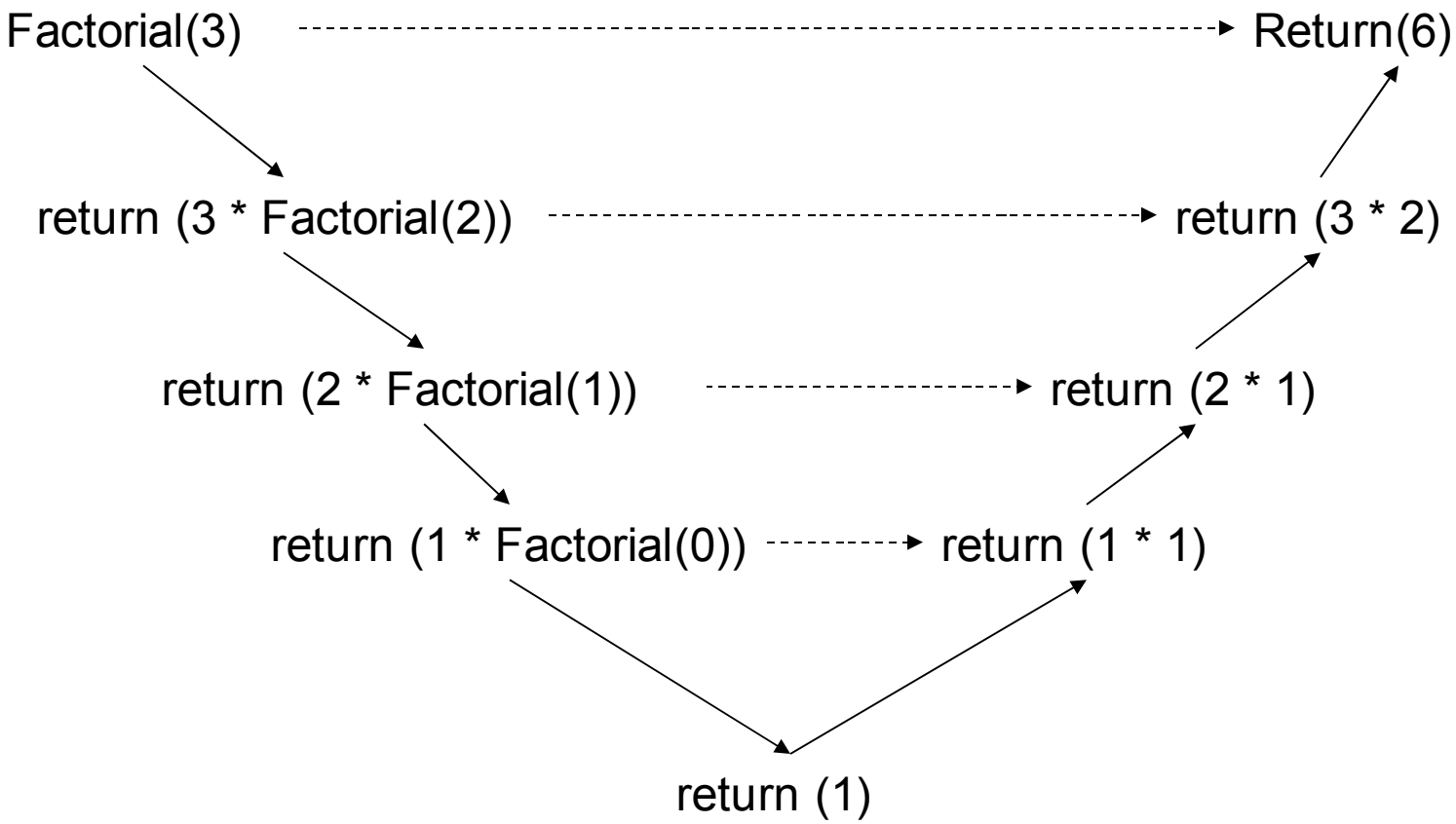
Η αναδρομή στη C

- Η αναδρομή στη C πραγματοποιείται ορίζοντας συναρτήσεις που καλούν **τον εαυτό τους** (άμεση αναδρομή).
- Υποστηρίζεται και η **έμμεση** αναδρομή

Παράδειγμα: Η συνάρτηση Factorial

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

Παράδειγμα κλήσης της Factorial



Επαναληπτική υλοποίηση

```
int Factorial(int n)
{
   int i, result;
   result = 1;
   for (i = 2; i<= n; i++)
   {
     result = result * i;
   }
   return result;
}
```

Βασική δομή αναδρομικής συνάρτησης

```
συνάρτηση()
{

    if (έλεγχος απλής περίπτωσης) {
        return (απλή λύση χωρίς τη χρήση αναδρομής);
    } else {
        return (αναδρομική κλήση που απαιτεί την κλήση της
        ίδιας συνάρτησης);
}
```

Ένα παράδειγμα: Συνάρτηση ύψωσης στη δύναμη

```
static int RaiseIntToPower(int n, int k)
{
   /* Έλεγχος απλής περίπτωσης */
   if (k ==0) {
      return (1);
   } else {
   /* Αναδρομική κλήση της ίδιας συνάρτησης */
      return (n * RaiseIntToPower(n, k-1));
   }
}
```

Κατά τη χρήση της αναδρομής

- Βρίσκουμε τη λύση για την απλή περίπτωση.
- «Φανταζόμαστε» ότι έχουμε βρει τη λύση για μια (ή περισσότερες) απλούστερες περιπτώσεις
- Με βάση τις παραπάνω απλούστερες περιπτώσεις, συνθέτουμε τη λύση για τη γενικότερη περίπτωση

Τι θα συμβεί αν λείπει ο έλεγχος της απλής περίπτωσης;

- Τι κάνει το παρακάτω πρόγραμμα;

```
main() { inc(1);}

int inc(int k)
{
   k++;
   inc(k);
}
```

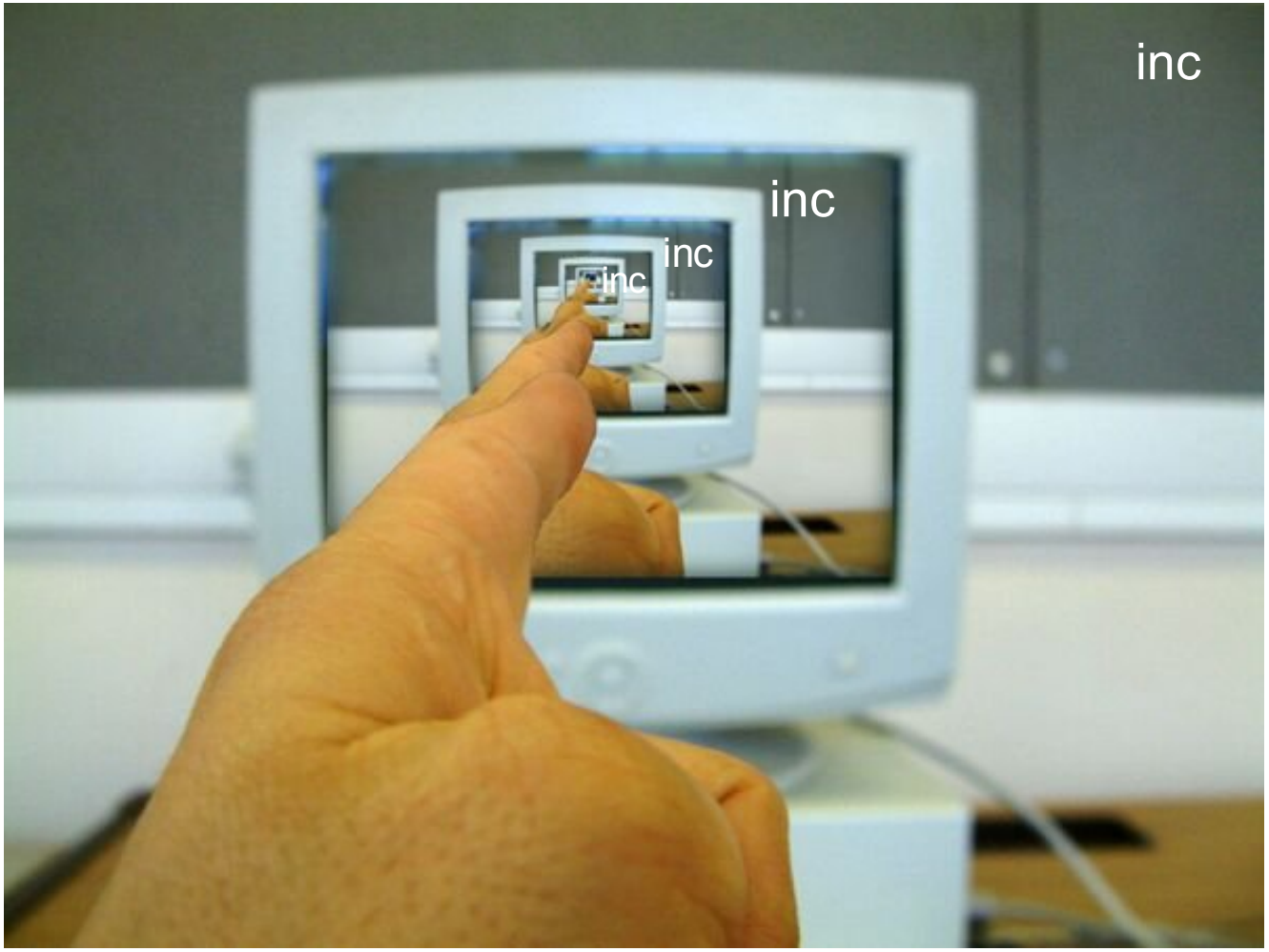
Περίπτωση «ατέρμονος» αναδρομής

- Θεωρητικά, η προηγούμενη συνάρτηση, inc, όταν κληθεί, καλεί επ' άπειρον τον εαυτό της
  - Το πρόγραμμα μπαίνει σε μια ατέρμονα επανάληψη («κολλάει»)
- Πρακτικά, μετά από κάποιο χρόνο το πρόγραμμα τερματίζει εμφανίζοντας ένα μήνυμα σφάλματος.
  - π.χ. **`Stack overflow`** ή
  - **`Segmentation Fault`**

Γιατί συμβαίνει αυτό;

- Σε κάθε κλήση συνάρτησης (αναδρομική ή όχι) δεδομένα εγγράφονται σε μια περιοχή της μνήμης που ονομάζεται Στοίβα (Stack).
- Τέτοια δεδομένα είναι:
  - Οι τιμές των πραγματικών παραμέτρων
  - Η διεύθυνση επιστροφής
  - Τοπικές μεταβλητές
- Μετά από ένα μεγάλο αριθμό αναδρομικών κλήσεων η στοίβα υπερχειλίζει (overflow) και το πρόγραμμα τερματίζει με την εμφάνιση κατάλληλου μηνύματος.

Περίπτωση άπειρων αναφορών μιας οντότητας στον εαυτό της



Μαθηματικά

- Η **μαθηματική επαγωγή**, ως μέθοδος απόδειξης, είναι μια περίπτωση αναδρομής

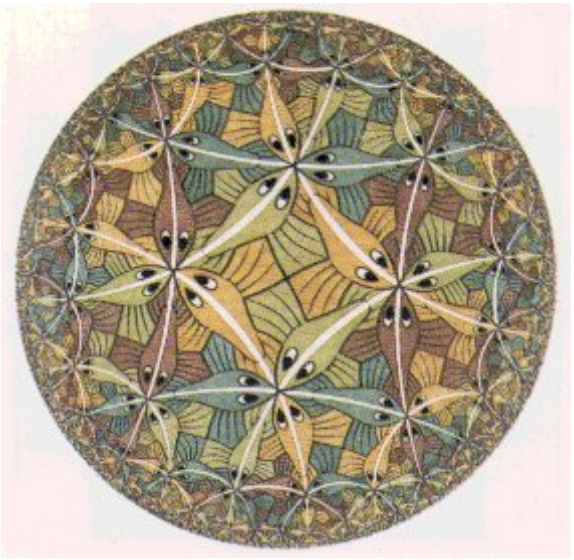
Η αναδρομή στην Τέχνη

M.C. Escher, Circle limit IV

Regular Division of the Plane VI

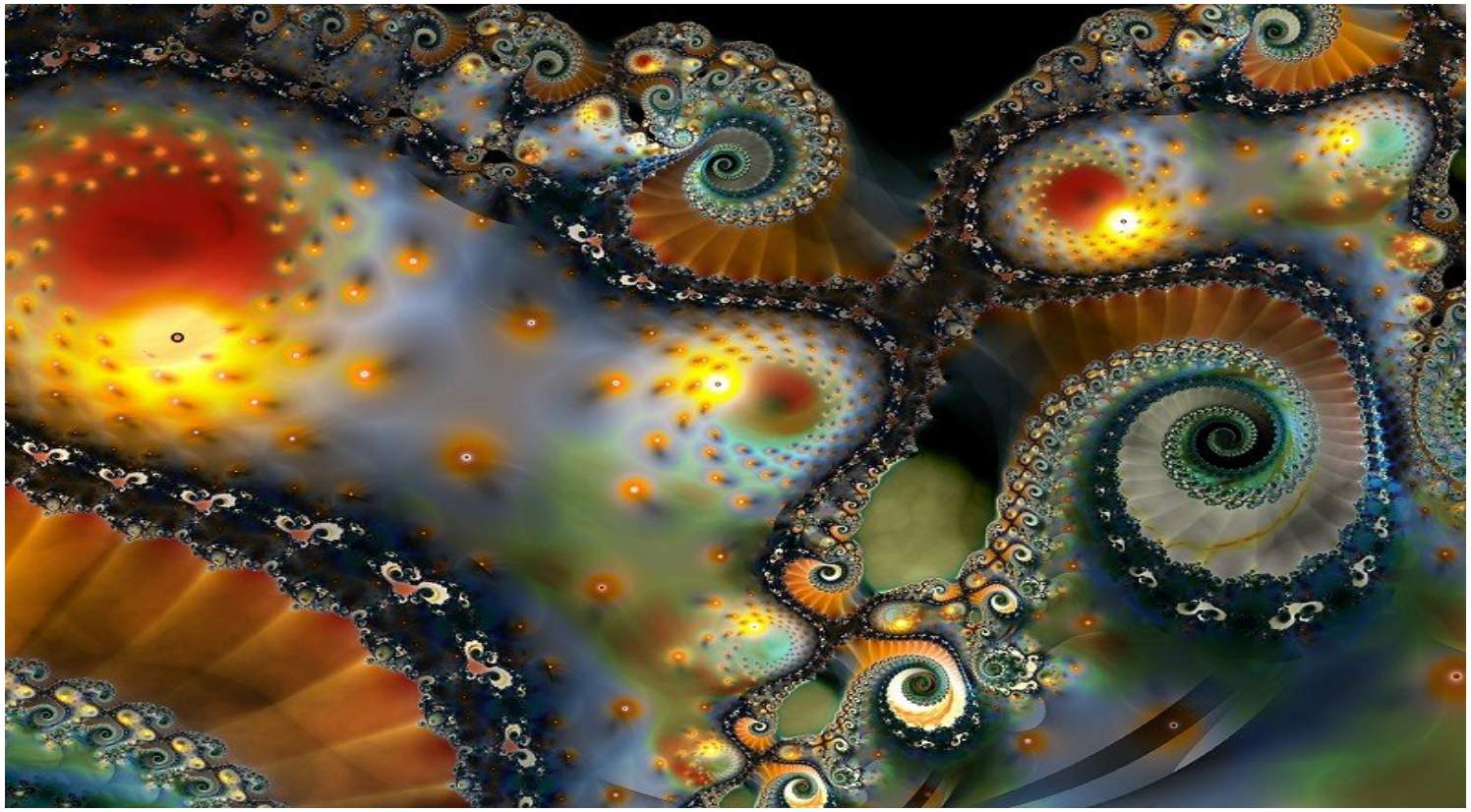
Η αναδρομή στην Τέχνη

M. C. Escher, Square Limit

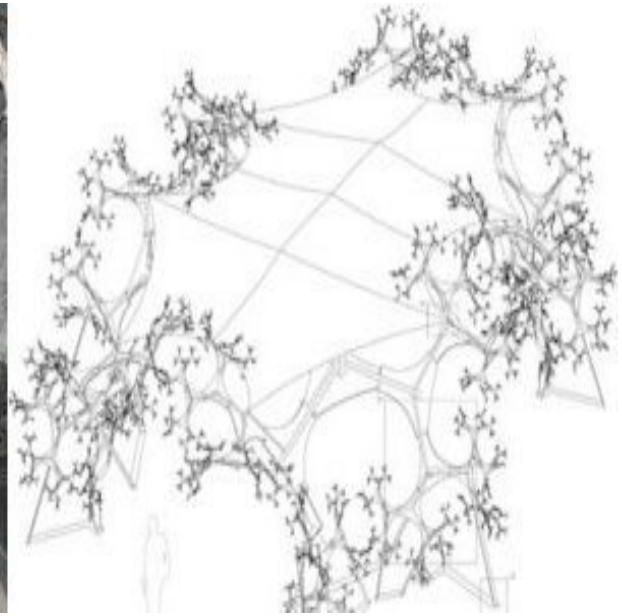


Circle Limit III

Fractals

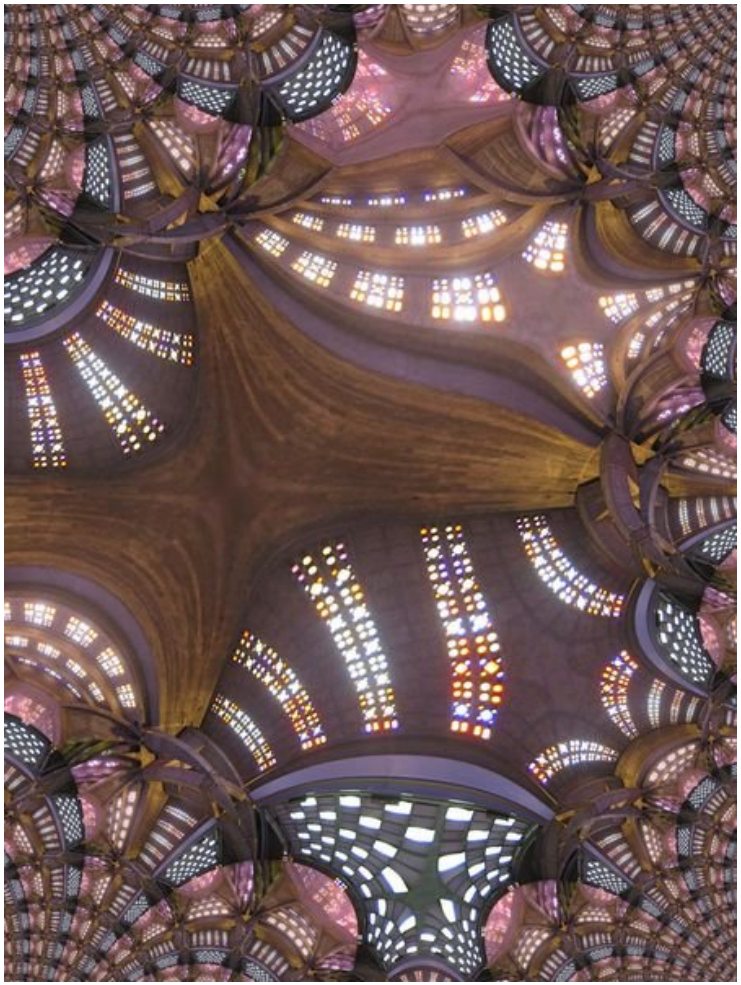


Αρχιτεκτονική/Γλυπτική



http://www.theverymany.net/2006_05_01_archive.html

Αρχιτεκτονική

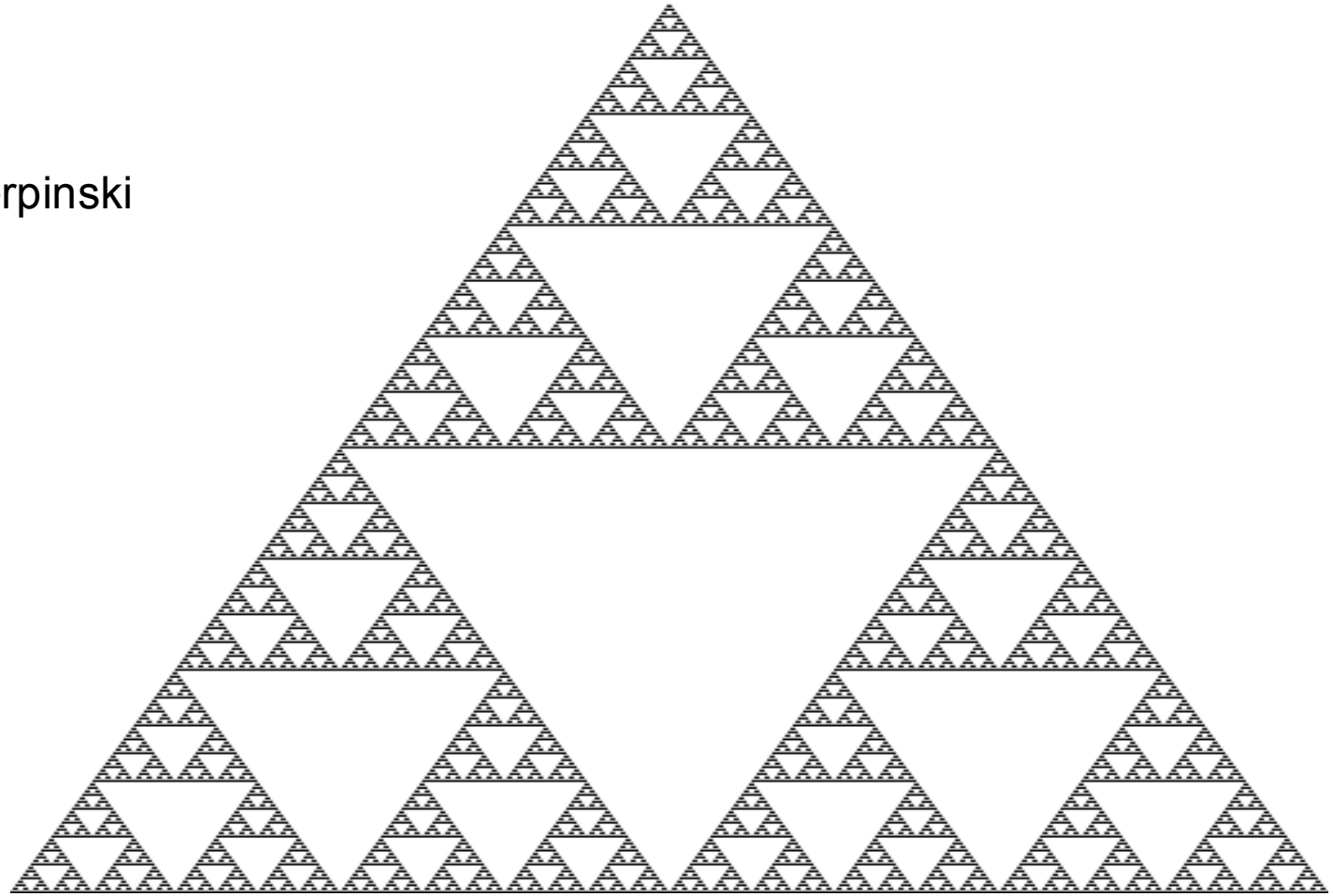


St. Joseph's Church, Havre

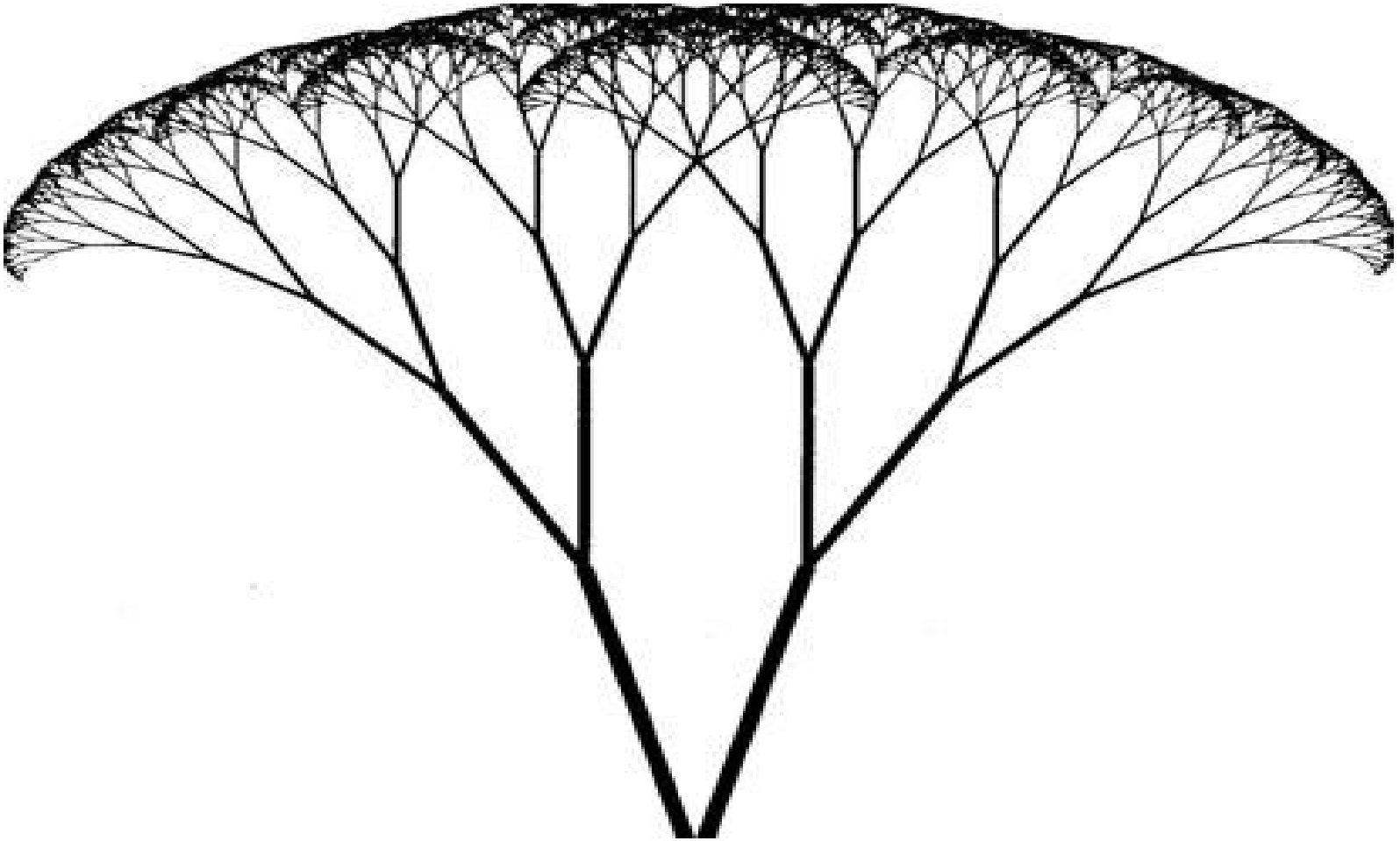
Auguste Perret (and R. Audigier) 1953-57

Fractals (2)

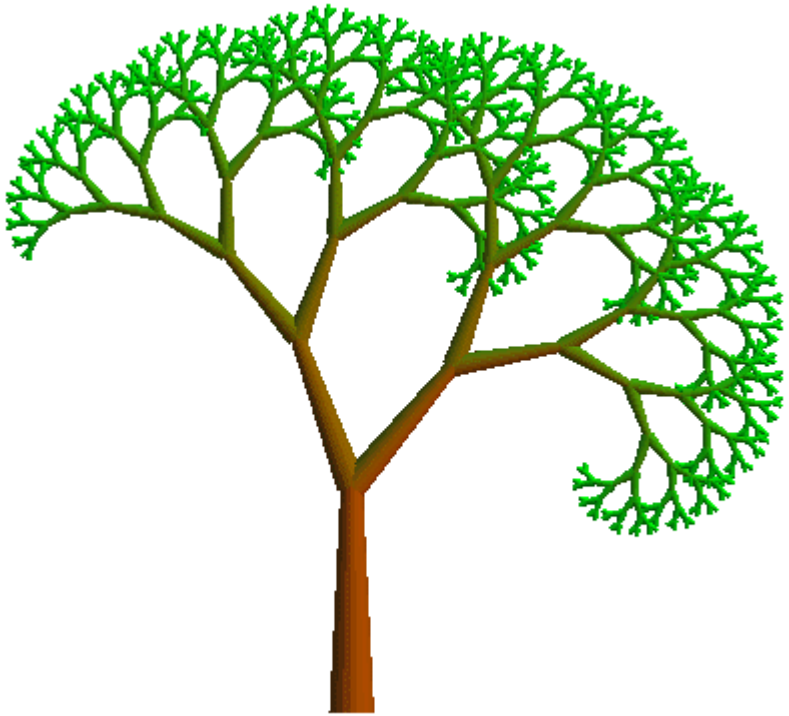
Τρίγωνο Sierpinski



Δένδρο Fractal (1)



Δένδρο Fractal (2)



Αναδρομή … στη φύση (1)

Αναδρομή … στη φύση (2)

Κούκλες «Μπαμπούσκα»

- Μια κούκλα μέσα σε μια ίδια κούκλα
- Μια συνάρτηση καλείται από μια «ίδια» συνάρτηση

Άλλα πεδία όπου είναι δυνατόν να προκύψει αναδρομή

- Λογοτεχνία
- Γλώσσα
- Μουσική
- …

Παράδειγμα: Αριθμοί Fibonacci

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

Πλήρες πρόγραμμα

```
#include <stdio.h>

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

main()
{
    int x;
    while(1){
          scanf("%d",&x);
          if (x<0)
                    return;
          printf("%d\n", fib(x));
    }
}
```

Ανάλυση αλγορίθμων

Αξιολόγηση της αποδοτικότητας αλγορίθμων

- Πόσο γρήγορα “εκτελείται” ένας αλγόριθμος;
- Η αξιολόγηση της **σχετικής** αποδοτικότητας αλγορίθμων είναι γνωστή ως **ανάλυση αλγορίθμων**
- Σχετική αποδοτικότητα
  - Πόσο αποδοτικότερος είναι ένας αλγόριθμος σε σχέση με έναν άλλον

Σχετική αποδοτικότητα

- Δεν ενδιαφέρει η απόλυτη ταχύτητα εκτέλεσης ενός αλγορίθμου καθώς εξαρτάται από
  - την ταχύτητα του εκάστοτε υπολογιστή στον οποίο εκτελείται
  - τη γλώσσα προγραμματισμού στον οποίο είναι υλοποιημένος
  - άλλους παράγοντες

Υπολογιστική πολυπλοκότητα

- Πόσο ποιο αργή είναι η ταξινόμηση ενός πίνακα 1000 στοιχείων σε σχέση με την ταξινόμηση ενός πίνακα 10000 στοιχείων;
- Το μέγεθος ενός προβλήματος (π.χ. το μέγεθος του πίνακα) παριστάνεται με το N
- “Η σχέση του N και του χρόνου εκτέλεσης ενός αλγορίθμου, όταν το N είναι μεγάλο, ονομάζεται **υπολογιστική πολυπλοκότητα** του συγκεκριμένου αλγορίθμου”.

Μέτρο πολυπλοκότητας

- Στην επιστήμη των υπολογιστών χρησιμοποιείται ένας ειδικός συμβολισμός που παριστάνει το **μέτρο της πολυπλοκότητας ενός αλγορίθμου**.
- Συμβολισμός του μεγάλου όμικρον O (από τη λέξη Order (τάξη).
- Παραδείγματα O(1), $O(N^2)$, O(logN), κ.λπ.

Παραδείγματα: O(1)

- Ένας αλγόριθμος για την εύρεση του **πρώτου** στοιχείου ενός πίνακα **δεν εξαρτάται** από το μέγεθος του πίνακα.
  - Λέμε ότι ο αλγόριθμος εκτελείται σε σταθερό χρόνο
  - Ο σταθερός χρόνος συμβολίζεται ως **O(1)**

Παραδείγματα: O(N)

- Αλγόριθμος γραμμικής αναζήτησης ενός στοιχείου σε έναν πίνακα:
- Για έναν πίνακα με N στοιχεία χρειάζονται N βήματα (συγκρίσεις)
- Ο αλγόριθμος εκτελείται σε **γραμμικό χρόνο**.
- Ο γραμμικός χρόνος εκτέλεσης συμβολίζεται ως O(N).

Παραδείγματα: O(logN)

- Αλγόριθμος δυαδικής αναζήτησης ενός στοιχείου σε έναν πίνακα:
- Για έναν πίνακα με N στοιχεία χρειάζονται (περίπτου) $\log_2 N$ βήματα (συγκρίσεις)
- Ο αλγόριθμος εκτελείται σε **λογαριθμικό χρόνο**.
- Ο γραμμικός χρόνος εκτέλεσης συμβολίζεται ως O(logN).

Παράδειγμα: $O(N^2)$

- Ο αλγόριθμος της ταξινόμησης με επιλογή εκτελείται σε

$$N+N-1+N-2+..+3+2+1=\sum_{i=0}^{N-1}(N-i)=\frac{N^2+N}{2}$$

- Στον παραπάνω τύπο:
  - Απαλοίφονται οι όροι που δεν είναι σημαντικοί για μεγάλες τιμές του N
  - Απαλοίφονται σταθεροί συντελεστές
- Η επίδοση του αλγορίθμου είναι $O(N^2)$. Λέμε ότι ο αλγόριθμος εκτελείται σε **τετραγωνικό χρόνο**.

Παράδειγμα: O(NlogN)

- Ο αλγόριθμος ταξινόμησης με συγχώνευση
  - Ο πίνακας διαιρείται σε δύο υπο-πίνακες μισού μεγέθους από τον αρχικό
  - Αν ο πίνακας δεν έχει κανένα ή έχει μόνο ένα στοιχείο τότε είναι ταξινομημένος
  - Οι δύο υποπίνακες ταξινομούνται (αναδρομικά)
  - Οι δύο ταξινομημένοι υποπίνακες συγχωνεύονται
- Αλγόριθμος “διαίρει και βασίλευε”.
- Ο χρόνος εκτέλεσης του αλγορίθμου είναι $N\log_2 N$

Ανάλυση του αλγορίθμου

- Απαιτούνται
  - N βήματα στο πρώτο επίπεδο
  - 2 N/2 βήματα στο δεύτερο επίπεδο
  - 4 N/4 βήματα στο τρίτο επίπεδο
  - 8 N/8 βήματα στο τέταρτο επίπεδο
  - ...
- Ο χρόνος εκτέλεσης του αλγορίθμου είναι $N\log_2 N$

```
  right)
{
   int i, last;
   if (left>= right) return;
   swap(v,left, left+right)/2);
   last = left;
   for (i = left+1; i <= right; i++)
      if (v[i] < v[left]) {
         last++;
         swap(v, last, i);
      }
      qsort(v,left, last-1);
      qsort(v, last+1, right);
   }
```

Μέτρο πολυπλοκότητας (συνέχ.)

- Με τον παραπάνω συμβολισμό ορίζονται κατηγορίες (κλάσεις) αλγορίθμων ανάλογα με τον χρόνο εκτέλεσής τους.
- Έτσι, δύο αλγόριθμοι που εκτελούνται σε γραμμικό χρόνο λέμε ανήκουν στην κλάση O(N)
- Στην περίπτωση αυτή λέμε ότι οι αλγόριθμοι “είναι O(N)”