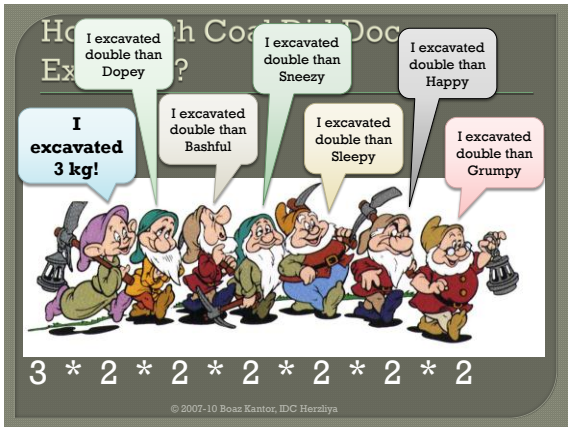
Recursions

Author: Boaz Kantor
The Interdisciplinary Center, Herzliya
Introduction to Computer Science
Winter 2009-10 Semester



3 * 2 * 2 * 2 * 2 * 2 * 2

The Coal Recursion

- How much coal did each dwarf excavate?

```
public class CoalRecursion {
  static int howMuchWasExcavated(int dwarfNumber) {
       if (dwarfNumber == 1) {
            return 3;
       }
       return 2 * howMuchWasExcavated(dwarfNumber - 1);
  }
}
```

I came, I saw, I conquered!

- Divide and conquer approach
- Like inductions:
  - 1 base rule: P
  - Assuming rule Q for n=k-1
  - Implementing Q for n=k
- Skeleton:

```
static int Q(k) {
   if (k == basic n) {
         return P;
   }
   return Q(k-1);
}
```


Examples

```
static int howMuchWasExcavated(int dwarfNumber) {
   if (dwarfNumber == 1) {
         return 3;
   }
   return 2 * howMuchWasExcavated(dwarfNumber - 1);
}

static int factorial(int num) {
   if (num == 1 || num == 0) {
         return 1;
   }
   return num * factorial(num - 1);
}

static int fibonacci(int num) {
   if (num < 2) {
         return 1;
   }
   return fibonacci(num - 1) + fibonacci(num - 2);
}
```


Recursions Theorem

- Recursions are always:
  - Replaceable by loops
- Recursions are sometimes:
  - More elegant than loops
- However, recursions usually perform
  - Worse than loops
- A powerful mechanism!

Practice

- How much **total coal** was excavated by the 7 dwarfs?
- Reminder: to calculate how much coal was excavated by a specific dwarf:

```
static int howMuchWasExcavated(int dwarfNumber) {
   if (dwarfNumber == 1) {
         return 3;
   }
   return 2 * howMuchWasExcavated(dwarfNumber - 1);
}

static int totalExcavatedCoal(int dwarfNumber) {
   if (dwarfNumber == 1) {
         return howMuchWasExcavated(1);
   }
   return totalExcavatedCoal(dwarfNumber - 1) +
          howMuchWasExcavated(dwarfNumber);
}
```


Writing your own recursions

1. Think of the stop condition (with factorial: fact(1) returns 1).
2. Think one step forward (with factorial: fact(2) returns 2*fact(1)).
3. Deduce (with factorial: fact(n) returns n*fact(n-1)).
4. Implement and test on small values first.

Analyzing recursions

```
public static String f1(String s) {
      if (s.length() <= 1) {
            return s;
      }
      return s.charAt(0) + f1(s.substring(1));
      return f1(s.substring(1)) + s.charAt(0);
}

public static int f2(int a, int b) {
         if (b == 0) {
               return 0;
         }
         if (b % 2 == 0) {
               return f2(a + a, b / 2);
         }
         return f2(a + a, b / 2) + a;
}
```

Additional material

- Mathematical proofs of recursions
- Tail vs. head recursions
- Recursions and the stack (memory)
- Procedural vs. functional recursions
- Fractals
- Recursive data structures

Extremitas

Recursions

Author: Boaz Kantor
The Interdisciplinary Center, Herzliya
Introduction to Computer Science
Winter 2009-10 Semester