Introduction to Algorithms Lecture 2

By: Darmen Kariboz

Content

- Sets intersection
- Selection Sort
- Bubble Sort
- Insertion Sort
- Time Sorting
- Counting Sort
- Recursion
- Towers of Hanoi
- HomeWork

Sets intersection

You need to output all numbers (without repetition) that both sets contain

Sample input:	Sample output:
11 6 2 4 6 8 10 12 10 8 6 4 2 3 6 9 12 15 18	6 12

Sets intersection

1. setA <- array1 values
2. setB <- array2 values
3. For i = 1 to size(setA) do
4. For j = 1 to size(setB) do
5. if setA[i]==setB[j] then
6. output setA[i]

Sorting

- Non-decreasing vs increasing…
- Algorithm complexity:
  - O(log(N)), O(N), O(N^2), O(e^N),…
  - sorting algorithms can run either: O(N*log(N)) or O(N^2)

Selection Sort

Selection Sort - puts the minimal element on its own place

```
1. for i=1 to n-1 do
2.     for j=i+1 to n do
3.         if(a[i] > a[j]) index=j
       a[i] <---> a[index];
```

Bubble Sort

Bubble Sort -starts from the end of an array and swaps two neighboring elements if they don‘t correspond.

1. for i=1 to n-1 do
2. for j=n-1 downto i do
3. if(a[j] > a[j+1]) a[j] <---> a[j+1];

Insertion Sort

Insertion Sort - like Card Deck, puts an element in already sorted array

1. **for i ←2 to length(A) do**
2. **key ← A[i]**
3. **j ← i - 1**
4. **while j >= 1 and A[j] > key do**
5. **A[j+1] ← A[j]**
6. **j ← j -1**
7. **A[j+1] ← key**

Time sorting

You have to sort time, given in a input file in the following format: “hh:mm:ss”

Sample input:	Sample output:
4 10:20:30 7:30:00 23:59:59 13:30:30	7:30:00 10:20:30 13:30:30 23:59:59

Time sorting

- Convert to seconds:
- x = 3600*h + 60*m + s

- Convert from seconds:
- h = x div 3600
- m = (x div 60) mod 60
- s = x mod 60

Counting Sort

Sorts information if number of possible values are limited.

Complexity is O(N)

Counting Sort – count quantity of each value.

1. **Create array Q for all values**
2. **for i ← 1 to length(A) do**
3. **Q[A[i]] ← Q[A[i]] + 1**
4. **for j ← 1 to length(Q) do**
5. **if Q[j] > 0 then**
6. **Q[j] times output j**

Recursion

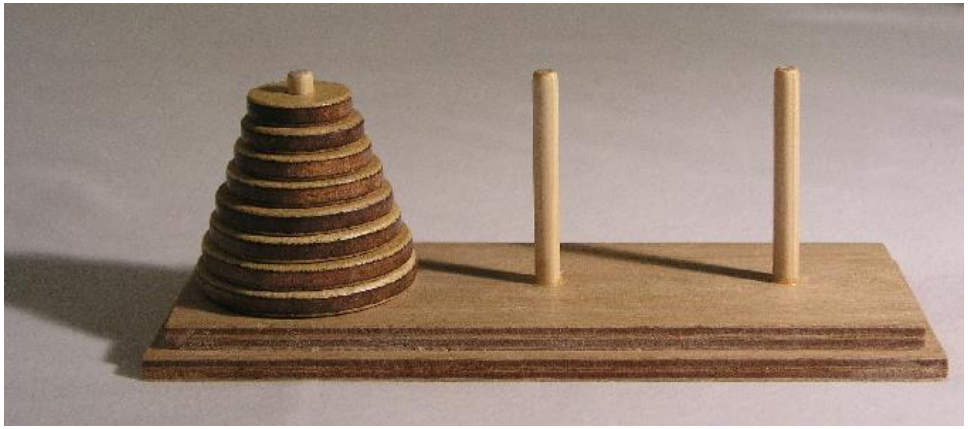
Recursion

- Function that calls itself.
- Recursion memory, stack overflow
- A simple base case (or cases)
- A set of rules which reduce all other cases toward the base case.

The Fibonacci sequence is a classic example of recursion:

- Fib(0) is 0 [base case]
- Fib(1) is 1 [base case]
- For all integers n > 1: Fib(n) is (Fib(n-1) + Fib(n-2)) [recursive definition]

Towers of Hanoi



Towers of Hanoi

```
void hanoi (int n, char from, char temp, char  to)
    if  n = 1 do
        print (from + " -> " + to);
    else
        hanoi(n - 1, from, to, temp);
        print (from + " -> " + to);
        hanoi(n - 1, temp, from, to);
```

Home Work

- **Sudoku**
- **Museum**

Sudoku

- Sudoku of size N is a square with a side N^2, that contains N^2 inner squares with sides equal to N.
- The sudoku is called correct, if and only if, each row, each column and each inner square contain all numbers from 1 to N^2.
- Given sudoku of size N. You have to find out, where it is correct or not.

Sample input:	Sample output:
3 1 3 2 5 4 6 9 8 7 4 6 5 8 7 9 3 2 1 7 9 8 2 1 3 6 5 4 9 2 1 4 3 5 8 7 6 3 5 4 7 6 8 2 1 9 6 8 7 1 9 2 5 4 3 5 7 6 9 8 1 4 3 2 2 4 3 6 5 7 1 9 8 8 1 9 3 2 4 7 6 5	Correct

Museum

- In a museum, the time when a customer comes in and gets out is registered.
- Therefore, there is a journal, where you can get N intervals of time, where the value shows time when customer came in, the second number shows time when customer got out of a museum.
- You have to write a program, that calculates the maximum number of customers who were in a museum at the same time.

Sample input:	Sample output:
6 9:00 10:07 10:20 11:35 12:00 17:00 11:00 11:30 11:20 12:30 11:30 18:15	4