SIOUX

SOURCE OF YOUR TECHNOLOGY

Algorithmics

23 april 2013

Rico Huijbers <rico.huijbers@sioux.eu>

Why?

- You will need to know this stuff:
  - If you ever need to *design* an algorithm or data structure
  - If your software will ever be used to process *a lot* of data (even if you use a library)
  - If you want to be more than a glue code monkey!

Complexity

A measure for how the *run time* of your algorithm varies based on the *size* of your input.

O(1)

O(log N)

O(N)

O(N log N)

O(N^2), O(N^3), ...

O(2^N)

O(N!)

Complexity

A measure for how the *run time* of your algorithm varies based on the *size* of your input.

Intuitively: how many “steps” are performed for every input element.

$O(1)$

$O(\log N)$

$O(N)$

$O(N \log N)$

$O(N^2)$, $O(N^3)$, ...

$O(2^N)$

$O(N!)$

Quiz

What's the complexity of this C# code?

```
IList<int> list;
int x;

Console.WriteLine( list[x] );
```

Quiz

What about now?

```
IList<int> list = new List<int>();
int x;

Console.WriteLine( list[x] );
```

Quiz

...and now?

```
IList<int> list = new LinkedList<int>();
int x;

Console.WriteLine( list[x] );
```

Reusable Code

What if you're not in control?

```
int Sum(IList<int> list)
{
    var sum = 0;
    for (var i = 0; i < list.Count(); i++)
    {
        sum += list[i];
    }
    return sum;
}
```

Reusable Code

What if you're not in control?

```
int Sum(IList<int> list)
{
    var sum = 0;
    for (var i = 0; i < list.Count(); i++)
    {
        sum += list[i];
    }
    return sum;
}
```

What about this operation?

Summary

Summary

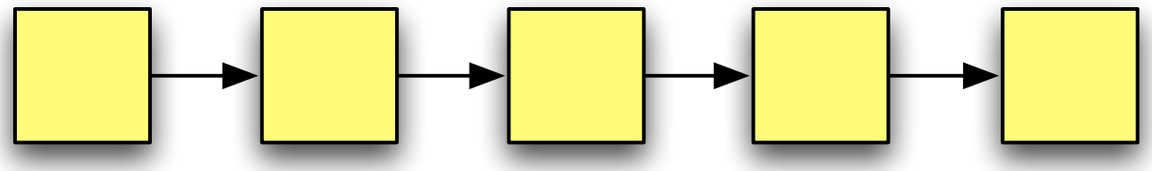
- Complexity is *not* about speed
  - It's about *scalability*
  - "Everything is fast for small *N*"

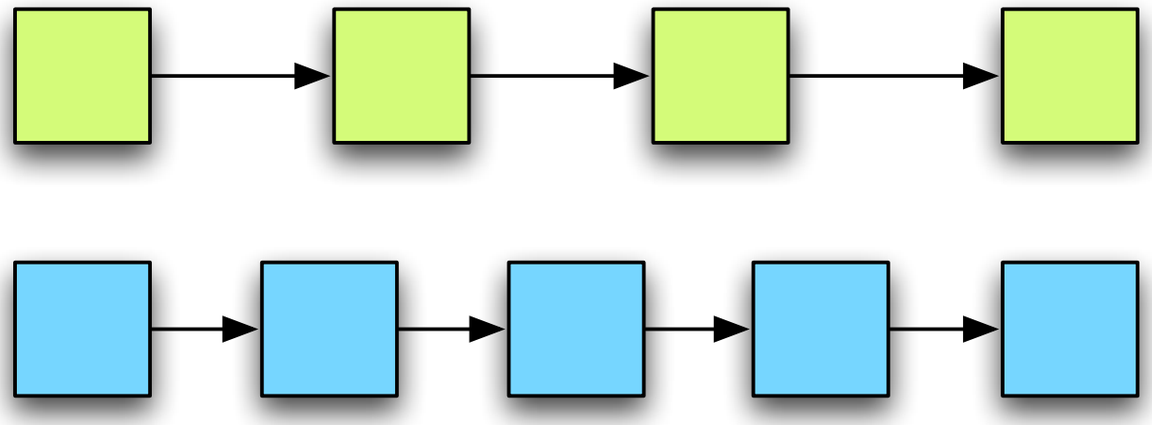
Summary

- Complexity is *not* about speed
  - It's about *scalability*
  - "Everything is fast for small *N*"
- Also: *space complexity*
  - Even more tricky: caches and virtual memory
  - O(N) algo can destroy O(log N) algo due to locality!

Exercise

Merging *k* sorted linked lists





http://leetcode.com/onlinejudge#question_23

Hints

Merging k sorted linked lists

Some possible solution shapes:

	Time	Space
A	O(N k)	O(1)
B	O(N log k)	O(N)
C	O(N log k)	O(k)

http://leetcode.com/onlinejudge#question_23

A

- Every iteration, add the smallest element of all heads to the new list and replace with the next element.
- Iterates over k lists for each of the N elements: O(N k)
- No extra memory needed: O(1)

B

- Pairwise merge of 2 lists until only one list remains.
- One set of pairwise merges takes O(N). Needs to be done log k times: O(N log k)
- O(N) for recursive merge, O(1) for in-place merge

C

- Like A, take the smallest head of the each list, but all heads in a min-heap and insert next list element on removal.
- Heap access is O(1), heap insert is O(log k): O(N log k)
- Heap contains k elements: O(k)

Moar...!

- http://leetcode.com/onlinejudge
- http://projecteuler.net