Lec 6 Asymptotic Complexity

* Big O notation refresher [partially omitted]

Suppose $f, g : \mathbb{N} \to \mathbb{R}^+$

$f(n) \in O(g(n))$ if $\exists_{c \in \mathbb{R},\, N \in \mathbb{N}}$. $\forall_{n > N}$, $f(n) \leq c g(n)$

Aside:

Ω is lower bound
Θ is both bounds

Complexity classes...
$O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$, ...

* Append complexity

```
fun append ([], Y) = Y
  | append (x::xs, Y) = X :: append (xs, Y)
```

$W_{append}(n, m)$ where m, n are length of inputs

"work"

Cases (for each clause)

looks like an inductive proof... but complicated

$W_{append}(0, m) = c_0$ (for all m and some c_0)

some constant

"recurrance"

$W_{append}(n, m) = c_1 + W_{append}(n-1, m)$ (for $n > 0$ and ...)

Unrolling method

easier way.
typically we do this

$= c_1 + c_1 + W_{append}(n-2, m)$
$= kc_1 + W_{append}(n-k, m)$
$= nc_1 + c_0$
$\in O(n)$

Reverse   using unrolling ← good when only one recursive call

Bad one

```
fun rev [] = []
  | rev (x::xs) = rev xs @ [x]
```

$W_{rev}(n)$ where n is length of input

Usually given. Otherwise prove it

Note we're using lemma about input and ouput size.

$W_{rev}(0) = c_0$

$W_{rev}(n) = c_1 + W_{rev}(n-1) + W_{append}(n-1, 1)$ $n > 0$

$\leq c_1 + W_{rev}(n-1) + c_2 n$ ← replacing with asymptotic bound

$\leq c_1 + c_2 n + W_{rev}(n-1)$

Unroll!

$\leq c_1 + c_2 n + c_1 + c_2(n-1) + W_{rev}(n-2)$

$\leq c_1 + c_2 n + c_1 + c_2(n-1) + c_1 + c_2(n-2) + W_{rev}(n-3)$

$\cdots \leq c_0 + nc_1 + c_2\left(\frac{n(n+1)}{2}\right)$

$\in O(n^2)$

Tail recursive rev

```
fun trev ([], acc) = acc
  | trev (x::xs, acc) = trev (xs, x::acc)
```

$W_{trev}(n, m)$ where m, n are len. of lists

$W_{trev}(0, m) = c_0$

$W_{trev}(n, m) = c_1 + W_{trev}(n-1, m+1)$ $(n > 0)$

$= 2c_1 + W_{trev}(n-2, m+2)$

$\cdots \in O(n)$

Treees ⚠ Unrolling may not work as well

```
datatype tree = Empty | Node of tree * int * tree

fun sum (Empty : tree) = 0
  | sum (Node(l, x, r)) = sum l + sum r + x
```

could have been depth, num leaves, etc.

$W_{sum}(n)$ where n is num. of nodes in input tree.

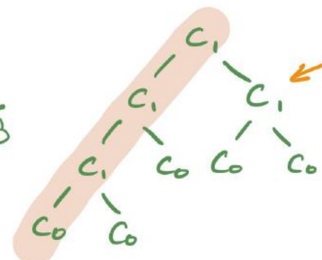
Uh oh we don't know these

All we know is $n_l + n_r = n - 1$

$W_{sum}(0) = c_0$

$W_{sum}(n) = W_{sum}(\ n_l\ ) + W_{sum}(\ n_r\ ) + c_1$

"The tree method"



Work at each node.

lemma: num leaf = num node + 1

Total $= nc_1 + (n+1)c_0$

* Notice we can run this in parallel. Finding **span** $S_{sum}(n)$

$Sum(0) = c_0$

$Sum(n) = c_1 + \max(S_{sum}(n_l), S_{sum}(n_r))$

In this case, we want the longest path ... but that depends on shape of tree.

Worse case: $n_l = n - 1$, $n_r = 0$.

Then $Sum(n) = c_1 + Sum(n-1)$ ← dominates

$\cdots \in O(n)$

Balanced tree

:= each subtree have roughly same size

└ turns out many def of roughly work, even high ~~tolerant~~ ones

Now suppose roughly := exactly

$$
\begin{aligned}
S_{sum}(n) &= c_1 + \max\left(S_{sum}\left(\tfrac{n}{2}\right), S_{sum}\left(\tfrac{n}{2}\right)\right) \\
&= c_1 + S_{sum}\left(\tfrac{n}{2}\right) \\
\cdots &= c_1 + c_1 + S_{sum}\left(\tfrac{n}{4}\right) \\
\cdots &= \log_2 n \, c_1 + c_0 \\
&\in O(\log n)
\end{aligned}
$$

Now consider size := depth

not necessarily balanced

$S_{sum}(d)$ where d is depth of tree

$S_{sum}(0) = c_0$

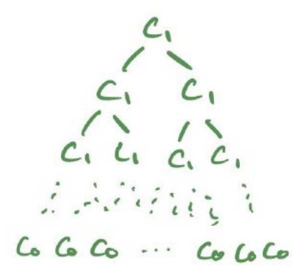
where $d > 0$, $d' \leq d-1$

$$
\begin{aligned}
S_{sum}(d) &= c_1 + \max\left(S_{sum}(d-1) + S_{sum}(d')\right) \\
&= c_1 + S_{sum}(d-1) \\
\cdots &= dc_1 + c_0 \\
&\in O(d)
\end{aligned}
$$

Notice depth gives more info about span

assuming balanced

$W_{sum}(d) =$



$$
\begin{aligned}
&c_1 \\
&+ 2c_1 \\
&+ 4c_1 \\
&+ \cdots \\
&+ 2^{d-1} c_0 \\
&\in O(2^d)
\end{aligned}
$$

Lemma: sum of exponential series is exponential

Bad Sorting

```
(* ins : int * int list -> int list
   REQ L sorted
   ENS ins (x, L) ↪ sorted permutation of x::L
*)
fun ins (x, []) = [x]
  | ins (x, y::ys) =
    ( case compare (x, y) of
         GREATER => y :: ins (x, ys)
        | _ => x::y::ys
    )

fun isort [] = []
  | isort (x::xs) = ins (x, isort xs)
```

$W_{ins}(0) = c_0$

$W_{ins}(n) = c_1 + W_{ins}(n-1)$ 1^{st} clause

$\quad = c_2$ 2^{nd} clause

$\quad \leq c_3 + W_{ins}(n-1)$

$\quad \in O(n)$

$W_{isort}(0) = c_0$

$W_{isort}(n) = c_1 + W_{isort}(n-1) + W_{ins}(n-1)$

$\quad \leq c_1 + c n_2 + W_{isort}(n-1)$

$\quad \in O(n^2)$