SAMFYB doctoc midterm 1 review notes 58f6e4b 6 minutes ago

1 contributor

69 lines (45 sloc) 2.48 KB

Midterm 1 Review

Work Analysis

```
(* Find : int * tree -> bool
 * REQ: true
 * ENS: Find (x, T) ===> true if x is in T | false otherwise
 *)
fun Find (x, Empty) = false
  | Find (x, Node(L, y, R)) =
    case (x = y) of
      true => true
    | false => let
                 val (b1, b2) = (Find (x, L), Find (x, R))
               in
                 b1 orelse b2
               end
```

> Note: This code is not optimized, but only for work/span analysis purpose.

Work/Span Analysis of the Above Function regarding Size

$$
\begin{aligned}
W(0) &= k_0 \\
W(n) &\leq k_1 + 2W(\frac{n}{2}) \\
&= \sum_{i=0}^{log(n)} 2^i k_i + c \cdot k_0 \\
&= (2^{log(n)+1} - 1) \cdot k_1 + c \\
&\in O(n)
\end{aligned}
$$

> Note: This is also of $O(2^d)$.

Exam. We are looking for 1. Recurrence, 2. Summation, 3. Big-O bound.

$$
\begin{aligned}
S(0) &= k_0 \\
S(d) &= k_1 + S(d-1) \\
&= k_1 + (k_1 + S(d-2)) \\
&= ... \\
&= d \cdot k_1 + k_0 \\
&\in O(d)
\end{aligned}
$$

> Note: This is also of $O(log(n))$.

Totality

Definition

A function `f` is total if for any valid input **value** `v`, `f v` always reduces to a **value**.

Consider: `f (g ()) ≅ 1` ?

This is wrong! `g ()` might not reduce to a **value**. (In fact, we have to know the totality of `g`.)

> Note: `f` could be total while a non-valid input could break it.

Tail Recursion

```
fun fact (0 : int) : int = 1
  | fact x = x * fact (x - 1)

fun fact' (0 : int, Acc : int) = Acc
  | fact' (x, Acc) = fact' (x - 1, x * Acc)
```