COMP9020 20T1

Week 7 Problem Set Induction, Recursion, Complexity Analysis

Foundations of Computer Science

[Show with no answers] [Show with all answers]

1. (Induction proofs)

   a. Prove by induction that $1 \cdot 1! + 2 \cdot 2! + \ldots + n \cdot n! \ = \ (n+1)! - 1$ for all $n \geq 1$ ($n \in \mathbb{N}$).

   b. Given the definition,

   $$\begin{aligned} s_1 &= 1 \\ s_{n+1} &= \tfrac{1}{1+s_n} \quad (n > 1) \end{aligned}$$

   prove by induction that

   $$s_n = \frac{\mathrm{FIB}(n)}{\mathrm{FIB}(n+1)}$$

   for all $n \geq 1$ ($n \in \mathbb{N}$).

   c. Suppose you would like to conclude that $P(n)$ is true for all $n \geq 0$ ($n \in \mathbb{N}$). For each of the following conditions, determine whether the condition is sufficient to prove this.

      i. $P(0)$ and $\forall n \geq 1\,(P(n-1) \Rightarrow P(n+1) \wedge P(n+2))$

      ii. $P(1)$ and $\forall n \geq 0\,(P(n+1) \Rightarrow P(n) \wedge P(n+2))$

      iii. $P(0)$ and $P(1)$ and $\forall n \geq 1\,(P(n) \wedge P(n+1) \Rightarrow P(n+2))$

      iv. $P(0)$ and $P(1)$ and $\forall n \geq 1\,(P(n) \Rightarrow P(n+2))$

      v. $P(0)$ and $P(1)$ and $\forall n \geq 1\,(P(n) \Rightarrow P(2 \cdot n) \wedge P(2 \cdot n+1))$

      vi. $P(0)$ and $P(1)$ and $\forall n \geq 1\,(P(2 \cdot n) \Rightarrow P(2 \cdot n-1) \wedge P(2 \cdot n+1))$

   [show answer]

2. (Recursive definitions)

   Recall the recursive definition of a rooted tree:

   $$\begin{array}{ll} \langle v;\ \rangle & \text{is a tree consisting only of a root node} \\ \langle r; T_1, T_2, \ldots, T_k \rangle\ \rangle & \text{is a tree with root } r \text{ and subtrees } T_1, T_2, \ldots, T_k \text{ at the root } (k \geq 1) \end{array}$$

   Prove that in any rooted tree, the number of leaves is one more than the number of nodes with a right sibling.

   Hint: This assumes a given order among the children of every node from left to right; see slide 22 (week 7) for an instance of this theorem.

   [show answer]

3. (Recurrences)

Recall the recurrence for Mergesort:

- $T(1) = 0$
- $T(n) = 2T(\frac{n}{2}) + (n-1)$

Prove that $n \cdot (\log_2 n - 1) + 1$ is a valid formula for $T(n)$ for all $n = 2^k$ (with $k \geq 1$).

[show answer]

4. (Asymptotic running times)
   a. Suppose you have the choice between three algorithms:
      i. Algorithm A solves your problem by dividing it into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
      ii. Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
      iii. Algorithm C solves problems of size n by dividing them into nine subproblems of size $\frac{n}{3}$, recursively solving each subproblem, and then combining the solutions in $\mathcal{O}(n^2)$ time.

      Estimate the running times of each of these algorithms. Which one would you choose?
   b. Order the following functions in increasing asymptotic complexity:
      i. $(n-1) \cdot (n-2) \cdot \sqrt{n}$
      ii. $\frac{3n}{\sqrt{n+1}}$
      iii. $\sqrt{7n^3 + 3n + 1}$
      iv. $5n^{\log(\log(n))}$
      v. $3n\log(n) + 2n^2$
      vi. $8 + \log(n) \cdot (n-1)$

[show answer]

5. (Big-Oh)
   a. *Without using the Master Theorem,* give tight big-Oh upper bounds for the divide-and-conquer recurrence $T(1) = 1; T(n) = T(\frac{n}{2}) + g(n)$, for $n > 1$, where
      i. $g(n) = 1$
      ii. $g(n) = 2n$
      iii. $g(n) = n^2$
   b. For each of the following functions, use the Master Theorem to determine the best upper bound complexity of $T(n)$.
      i. $T(n) = 9 \cdot T(\frac{n}{3}) + 3n(n+1)$
      ii. $T(n) = 8 \cdot T(\frac{n}{2}) + 8n(n+1)$

iii. $T(n) = 8 \cdot T(\frac{n}{2}) + 2n^2(n+1)$

iv. $T(n) = 6 \cdot T(\frac{n}{2}) + n^3$

v. $T(n) = 6 \cdot T(\frac{n}{3}) + n^2$

c. Analyse the complexity of the following recursive algorithm to test whether a number x occurs in an *unordered* list $L = [x_1, x_2, \ldots, x_n]$ of size n. Take the cost to be the number of list element comparison operations.

```
*Search*$(x, L = [x_1, x_2, \ldots, x_n])$:
   **if** $x_1 = x$ **then return** yes
   **else if** $n > 1$ **then return** *Search*$(x, [x_2, \ldots, x_n])$
   **else return** no
```

d. Analyse the complexity of the following recursive algorithm to test whether a number x occurs in an *ordered* list $L = [x_1, x_2, \ldots, x_n]$ of size n. Take the cost to be the number of list element comparison operations.

```
*BinarySearch*$(x, L = [x_1, x_2, \ldots, x_n])$:
   **if** $n = 0$ **then return** no
   **else if** $x_{\lceil \frac{n}{2} \rceil} > x$ **then return** *BinarySearch*$(x, [x_1, \ldots, x_{\lceil \frac{n}{2} \rceil - 1}])$
   **else if** $x_{\lceil \frac{n}{2} \rceil} < x$ **then return** *BinarySearch*$(x, [x_{\lceil \frac{n}{2} \rceil + 1}, \ldots, x_n])$
   **else return** yes
```

[show answer]

6. **Challenge Exercise**

Prove by induction that every connected graph $G = (V, E)$ must satisfy $e(G) \geq v(G) - 1$.

Hint: You can use the fact from a previous lecture that $\sum_{v \in V} deg(v) = 2 \cdot e(G)$.

[show answer]

Assessment

After you have solved the exercises, go to COMP9020 20T1 Quiz Week 7 to answer 5 quiz questions on this week's problem set (Exercises 1-5 only) and lecture.

The quiz is worth 2.5 marks.

There is no time limit on the quiz once you have started it, but the deadline for submitting your quiz answers is **Thursday, 9 April 10:00:00am**.

Please continue to respect the **quiz rules**:

Do …

- use your own best judgement to understand & solve a question
- discuss quizzes on the forum only **after** the deadline on Thursday

Do not …

- post specific questions about the quiz **before** the Thursday deadline
- agonise too much about a question that you find too difficult