Fundamentals of AI Alan Smaill

2/10/08

Tutorial for Week 3

You should work through these questions for the tutorials held in week 3..

1. Estimate as best you can the “big oh” worst case running time of the following procedures as a function of integer n.

   (a) In the data type of two-dimensional arrays, we imagine that data is stored at points in a rectangle — for array C the information at point (3,4) is written as C[3,4]. Accessing the information, and updating it, count as single steps (constant time operations).

   ```
   To multiply two nxn matrices A, B:
      For i goes from 1 to n
         For j goes from 1 to n
            Set C[i,j] = 0
            For k = goes from 1 to n
                Set C[i,j] = C[i,j] + A[i,k]*B[k,j]
   ```

   (b)
   ```
   To workHard with integer n:
      If n<=2 then return 1 and halt
         Otherwise let Y = workHard (n-1)
                   let Z = workHard (n-1)
                   Return Y + Z
   ```

   (c) Assume the parameter n is a positive power of 2 (e.g. 2,4,8,16,32,. . .).

   ```
   To compute mystery of integer n:
      Set Count = 0
      Set X = 2
      While X < N
         Set X = 2 * X
         Set Count = Count + 1
      Return Count
   ```

2. We saw an algorithm for evaluating boolean combinations of truth conditions (i.e. tests combined with AND and OR). This procedure does more work than is necessary. Suggest how to make it more efficient. Does you improved algorithm change the complexity measure in “big oh” terms?

   The primitive operations on trees here are:

   - forming a tree from two trees and a connective
   - deciding if a tree is a leaf, or has sub-trees
   - accessing the test identity from a leaf
   - accessing connective and subtrees from an internal node.

```
To evaluate test T:
   If T is a leaf
      determine test and look up test result
   Otherwise
      Let L be result of evaluating left of T
      Let R be result of evaluating right of T
      If connective of T is AND
          If both L and R are TRUE, return TRUE
          Otherwise return FALSE
      Otherwise (so connective is OR)
          If one of L or R is TRUE, return TRUE
          Otherwise return FALSE
```

3. Give an algorithm that will compute the list of tests from a test condition given in the form above. Thus for the condition

   `and(or(and(raining,warm),windy),or(humid,overcast))`

   the algorithm should return `[raining,warm,windy,humid,overcast]`.

   You can assume you have available an append procedure that runs in time linear with respect to the first argument; it takes two lists and returns the list got by tacking the first on the front of the second.

   What is the complexity of your algorithm? Is there a better algorithm? (There is an algorithm which is linear in the size of the input tree.)