CS 314 - Divide & Conquer

Note Title 1/29/2010

Announcements

- HW2 up - will do oral grading
  (Please read website FAQ on this!)
  May still work in pairs

Recursive Algorithms

What is recursion?

A function that calls itself.

Guidelines:

induction →

- base case
- recursive call has smaller input

Subset Sum (in lecture notes)

Given a set X of positive integers and a target value T, is there a subset of X that adds up to T?

Ex: X = {8, 6, 7, 5, 3, 10, 9}

T = 15

true or false? **true**

How to set it up recursively?

Base case: ① $X = \{\}$ and $T > 0$

return false

② $T = 0$

return true

③ $T < 0$

return false

Recursive case: (Assume $x_i \leq T$)

Consider $X = \{x_1, x_2, x_3, \ldots, x_n\}$

Question: Is $x_1 \in$ solution summing to T?

- Yes: - Store x_1
  - Recurse on $\{x_2, \ldots, x_n\}$ with target $T - x_1$
- No: Recurse on $\{x_2, \ldots, x_n\}$ with target T

Pseudocode:

X is an array of N items

```
SUBSETSUM(X[1..N], T):
    If T = 0
        return true
    else if T < 0 or N = 0:
        return false
    else
        return (SUBSETSUM(X[2..N], T) or
                SUBSETSUM(X[2..N], T - X[1]))
```

T = 15

X = {8, 6, 7, 5, 3, 10, 9}

return SUBSETSUM(X[2..7], 15) or SUBSETSUM(X[2..7], 7)

{6, 7, 5, 3, 10, 9}, 15

{6, 7, 5, 3, 10, 9}, 7

{7, 5, 3, 10, 9}, 15

{7, 5, 3, 10, 9}, 9

{7, 5, 3, 10, 9}, 7

{7, 5, 3, 10, 9}, 1

Proof of correctness: induction on N and T

Base Cases:
- $T=0$ return true $\emptyset$
- $T<0$ return false (since X contains only pos. #s, can't make neg sums)
- $N=0$ (and $T>0$) return false (nothing left in set)

IH: Alg works correctly for arrays of size $<N$ or target values $<T$

IS: Consider $X = X[1], \ldots, X[N]$ and target T.
Consider X[1]. X[1] is either in subset or not in subset. We check both possibilities.
By IH, those checks work.

If either of those returns true,
then X also has same subset

That subset gives me value T
(either by adding X[i] or not)?

Runtime: Recurrence

Let $T(n)$ = running time on n element array

$$T(n) = 2T(n-1) + O(1) \leftarrow$$

$$T(0) = O(1) \leftarrow$$

$$\Rightarrow T(n) = O(2^n)$$

Longest Increasing Subsequence (LIS)

Input: an array $A[1..n]$

Find longest possible indices $i_1 \dots i_k$
with $1 \le i_1 < i_2 < \dots i_k \le n$ such that
$A[i_j] \le A[i_{j+1}]$ for all j.

Example: $A = [8, 6, 7, 5, 3, 10, 9]$

longest?

Define recursively again:

Either A[i] is part of LIS
or it isn't!