CS F364
Design & Analysis of Algorithms

Algorithm Design Techniques

Divide & Conquer:

Memoization

- Example: 0,1 Knapsack

Memoization

- Memo (store) the results of the sub-problems
  - with the hope that *they will be re-used*
- Space-Time tradeoff:
  - If the results are reused time is saved
  - If the results are not reused space is wasted
- When do we know that results will be reused?
  - Structure of the problem
    - If the problem can be specified in terms of the sub-problems
    - then we can identify whether sub-problems overlap.

PROBLEM – 0/1 KNAPSACK

- Given:
  - A sack with max. capacity by weight: W kg.
  - Set S of items j (in store) labeled with
    - Weight : w_i (< W)
    - Price : p_j
- Assumption:
  - An item is either taken (in full) or not
  - All values (w_j, p_j, and W) are positive.
- Goal:
  - Fill the sack with maximum value (by price)
    - i.e. Find T subset of S, such that
    - $\Sigma_{i \text{ in } T}\, p_i$ is maximum and $\Sigma_{i \text{ in } T}\, w_i <= W$

KnapSack – Divide-And-Conquer

- Algorithm KnapSack(S, W) // S - Set of items; W-capacity
  1. Find all the subsets of S say T1, T2, …,Tk
     1. Let the cumulative weights be w(T1), w(T2), …, w(Tk) and prices be p(T1), p(T2),…,p(Tk)
     2. return (Tm,v) such that
        v = max { p(Ti) | w(Ti) <= W and 1<i<=k }
        and Tm such that w(Tm) = v

What is the time complexity of this algorithm?

MEMOIZED KNAPSACK

1. Let |S| be n; Denote items in S as j1, j2, … jn
2. Let the subsets of { j_1, j_2, …, j_{n-1} } be T1, T2, …,Tk
    1. Let the cumulative weights be w(T1), w(T2), …, w(Tk) and prices be p(T1), p(T2),…,p(Tk)
    2. Let Sol_{n-1} = (Tm,v) such that v = max { p(Ti) | w(Ti) <= W and 1<i<=k } = p(Tm)
3. Compute P_n = { p(Ti)+p_n | w(Ti)+w_n <= W and 1<=i<=k}
    - Compute (Tm’,v’) s.t. v’ = max(Pn) = p(Tm’)+p_n
4. return the pair with maximum value among (Tm,v) and (Tm’,v’)

What is the time complexity of this algorithm?