CS F364
Design & Analysis of Algorithms

ALGORITHM DESIGN TECHNIQUES

Divide & Conquer

Optimal Substructure Property

- Example: 0,1 Knapsack – Algorithm and Memoization.

2/4/2016 Sundar B. CSIS, BITS, Pilani

OPTIMAL SUBSTRUCTURE

- An optimization problem exhibits ***optimal substructure*** if
  - *an optimal solution to the problem contains within it optimal solutions to sub-problems*.
- Optimal Substructure holds for 0-1 KnapSack:
  - Consider the most valuable subset of items with weight at most W
  - If we remove item j from this subset, the remaining subset must be the most valuable, weighing at most $W - w_j$
- While we are constructing the solution (any) item j may or may not be part of the optimal solution
  - If item j is not part of the optimal solution, then the optimal solution is same as that for the set without j

OPTIMAL SUBSTRUCTURE

- Thus the problem structure of 0/1 Knapsack can be formulated as follows:
  - Let P(k,w) be
    - the maximum cumulative price obtainable from a subset of items { 1, 2, … k} weighing no more than w in total.
  - Then for any k>=1,
    - **P(k,w) =**

$$P(k,w) = \begin{cases} P(k-1, w) & \text{if } w_k > w \\ \max\{ P(k-1, w), P(k-1, w-w_k) + p_k \} & \text{otherwise} \end{cases}$$

2/4/2016 Sundar B. CSIS, BITS, Pilani

Divide and Conquer using Optimal Substructure

- KnapSack(S,B) { KS(|S|, B, weight, price); }
- KS(k,w, weight, price) // weight and price are functions on S

```
 if (k==0) return ({},0);
 if  (weight(k) > w) return KS(k-1, w)
 else {
         (m1,v1) = KS(k-1,w);
          (m2, v2) = KS(k-1, w-weight(k))
         if (v1 > v2+price(k)) return (m1,v1)
         else return (m2 U { k }, v2+price(k) );
 }
```

- Exercise: *Memoize* ***KS!***
  - What is the structure of the memo storage? What is its size?