CS F364
Design & Analysis of Algorithms

Algorithm Design Techniques

Kinds of Problems -

Optimization Problems - Example

Characterizing Problems

- One way of *characterizing and classifying problems* is by nature of the the output to be computed of a problem:

Category	Output	Examples
Decision	Is **p(i)** for input **i** ?	??
Search	Select one or more items **i** such that **p(i)** given a collection of items	??
Function	Compute **f(i)** on input **I**	??
Enumeration	Enumerate all **j** such that **p(i,j)** given input **i**	??
Counting	Count the number of **j** such that **p(i,j)** given input **i**	??
Optimization	Optimize **m(i,j)** such that **p(i,j)** given input **i**	??

Optimization Problems - Characterization

- Formal Description:
  - An optimization problem π is characterized by the quadruple $(I_\pi, F_\pi, m_\pi, goal_\pi)$
    - $I_\pi = \{ x \mid x$ is an input instance of $\pi \}$
    - $F_\pi(x) = \{ s \mid s$ is a feasible solution for $x \in I_\pi \}$
    - $m_\pi(x, y) = v$ where v is a quantitative measure of the "value" of the feasible solution i.e. $y \in F_\pi(x)$ for $x \in I_\pi$
    - $goal_\pi \in \{ min, max \}$

Problem – 0/1 KnapSack

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

KnapSack – Brute-Force Algorithm

- Algorithm KnapSack(S, B) // S - Set of items; B-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 the cumulative prices be **p(T1), p(T2),…,p(Tk)** respectively.
     2. return **(Tm,v)** such that
        v == max { p(Ti) | w(Ti) <= B and 1<i<=k } == p(Tm)

What is the time complexity of this algorithm?