CS F364
Design & Analysis of Algorithms

ALGORITHMS - COMPLEXITY

Structure of problems:

Strong NP-hardness and Pseudo-polynomial Time Algorithms

4/25/2015 Sundar B. CSIS, BITS, Pilani

Structure of problems – 0/1 Knapsack

- 0/1 KNAPSACK is an $\mathbb{NP}$–complete problem.
  - We have seen a dynamic programming algorithm for that solves this problem in time O(nW)
    - where n is the number of items in the input set and W is the capacity of the sack.
  - Such algorithms are referred to as pseudo-polynomial-time algorithms:
    - polynomial in one of the input numbers (but not its size).

Pseudo-polynomial time algorithms

- An algorithm A for a problem π runs in pseudo-polynomial time
  - if its running time is bounded by a polynomial function in |x| and max(x) for any instance x of π
  - where max(x) denotes the value of the largest number occurring in instance x.
- E.g. 0,1 Knapsack
  - max(x) = max (w_1, w_2, … w_n, p_1, p_2, … p_n, W)
  - The DP algorithm for 0,1 Knapsack runs in time O(n* W) i.e. O(n * max(x))
    - So, it is pseudo polynomial.

Strong $\mathbb{NP}$-hardness

- An $\mathbb{NP}$ problem Π is said to be strongly $\mathbb{NP}$-hard
  - if a polynomial p exists s.t. $\Pi^{max,p}$ is $\mathbb{NP}$-hard
  - where $\Pi^{max,p}$ is the problem obtained by restricting Π to only those instances x for which max(x) <= p(|x|)
- 0,1 Knapsack is not strongly NP-hard
  - Why?
- TSP is strongly NP-hard
  - Why?

Strong NP-hardness and Pseudo-polynomial-time algorithms

- Theorem:
  - No strongly NP-hard problem admits a pseudo-polynomial time algorithm unless P=NP
- Proof: (by contradiction.)
  - Let Π be strongly NP-hard with a pseudo-polynomial time algorithm A
    - i.e. A solves Π in time $q(|x|, max(x))$ for some polynomial q.
  - Then for any polynomial p, $\Pi^{max,p}$ can be solved in time $q(|x|, p(|x|))$.
  - But by strong-hardness of Π it follows there exists a polynomial p s.t. $\Pi^{max,p}$ is NP-hard i.e. P=NP