DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

Lecture 12 & 13

Brute force Techniques:

Exhaustive Search

<u>Instructors:</u>

Dr. Savitha G,
Assistant Professor, DSCA, MIT, Manipal

Dr. Abhilash K. Pai,
Assistant Professor, DSCA, MIT, Manipal

Recap of L10 & L11

- Brute Force techniques
  - Definition
  - Brute Force Sorting
    - Algorithm : Selection sort
    - Algorithm : Bubble sort
  - Brute Force Searching
    - Sequential search
  - Brute Force String Matching

Exhaustive search

- A brute force solution to a problem involving search for an element with a special property, usually among combinatorial objects such as permutations, combinations, or subsets of a set.

Method:

1. Generate a list of all potential solutions to the problem in a systematic manner.
2. Evaluate potential solutions one by one, disqualifying infeasible ones and, for an optimization problem, keeping track of the best one found so far.
3. When the search ends, announce the solution(s) found.

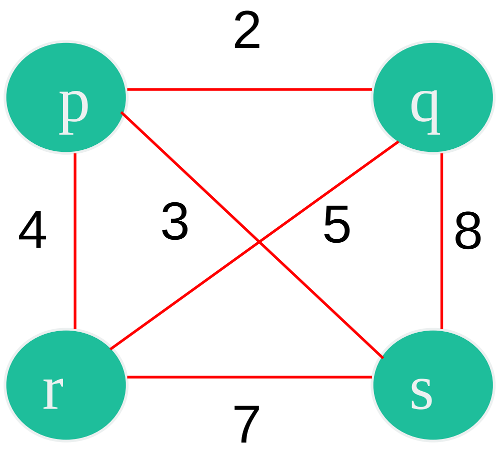
Traveling Salesman Problem using Exhaustive Search

The Travelling Salesman Problem:

- Given *n* cities with known distances between each pair, find the shortest tour that passes through all the cities exactly once before returning to the starting city.
- Alternatively: Find shortest Hamiltonian circuit in a weighted connected graph.

↳ A circuit that visits every vertex exactly once

Example:



Which are the Hamiltonian Circuits ?

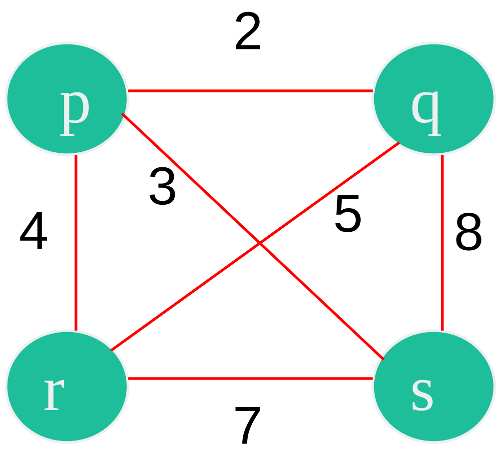
- If the tour starts with "p"
- If the tour starts with "r"

Traveling Salesman Problem using Exhaustive Search

Example: Let the start city be "q"

Tour	**Cost**
q→p→s→r→q	**2+3+7+5 = 17**
q→p→r→s→q	**2+4+7+8 = 21**
q→s→p→r→q	**8+3+4+5 = 20**
q→s→r→p→q	**8+7+4+2 = 21**
q→r→p→s→q	**5+4+3+8 = 20**
q→r→s→p→q	**5+7+3+2 = 17**

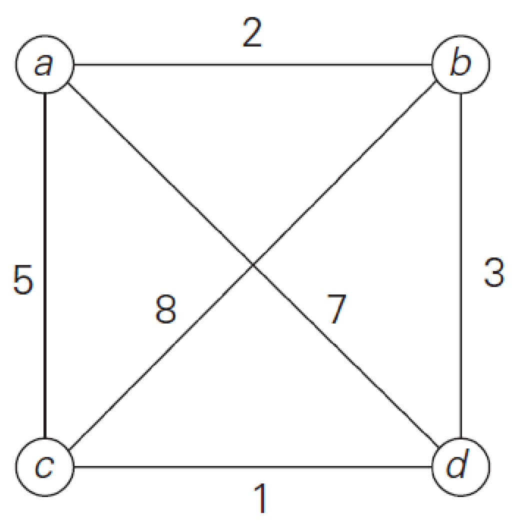
– optimal Solutions



Traveling Salesman Problem using Exhaustive Search

Class exercise:

- Solve the TSP for the following graph, where start city = "a"



Tour	Length	
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$	$l = 2 + 8 + 1 + 7 = 18$	
$a \rightarrow b \rightarrow d \rightarrow c \rightarrow a$	$l = 2 + 3 + 1 + 5 = 11$	optimal
$a \rightarrow c \rightarrow b \rightarrow d \rightarrow a$	$l = 5 + 8 + 3 + 7 = 23$	
$a \rightarrow c \rightarrow d \rightarrow b \rightarrow a$	$l = 5 + 1 + 3 + 2 = 11$	optimal
$a \rightarrow d \rightarrow b \rightarrow c \rightarrow a$	$l = 7 + 3 + 8 + 5 = 23$	
$a \rightarrow d \rightarrow c \rightarrow b \rightarrow a$	$l = 7 + 1 + 8 + 2 = 18$	

Efficiency : **$\Theta((n-1)!)$**

Knapsack Problem using Exhaustive Search

The Knapsack Problem:

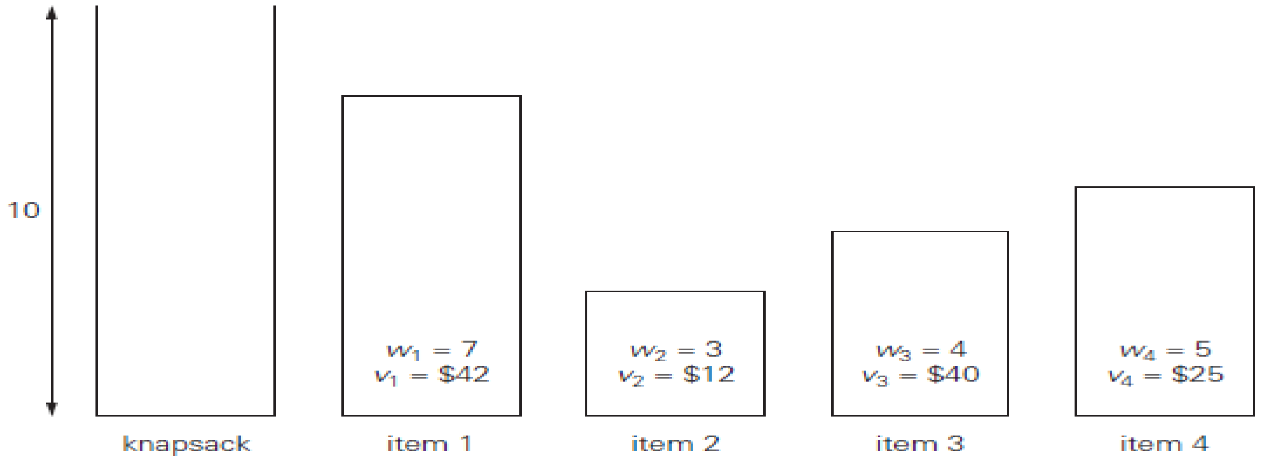
Given n items:

weights: $w_1 \quad w_2 \; \ldots \; w_n$

values: $v_1 \quad v_2 \; \ldots \; v_n$

&

a Knapsack of capacity W

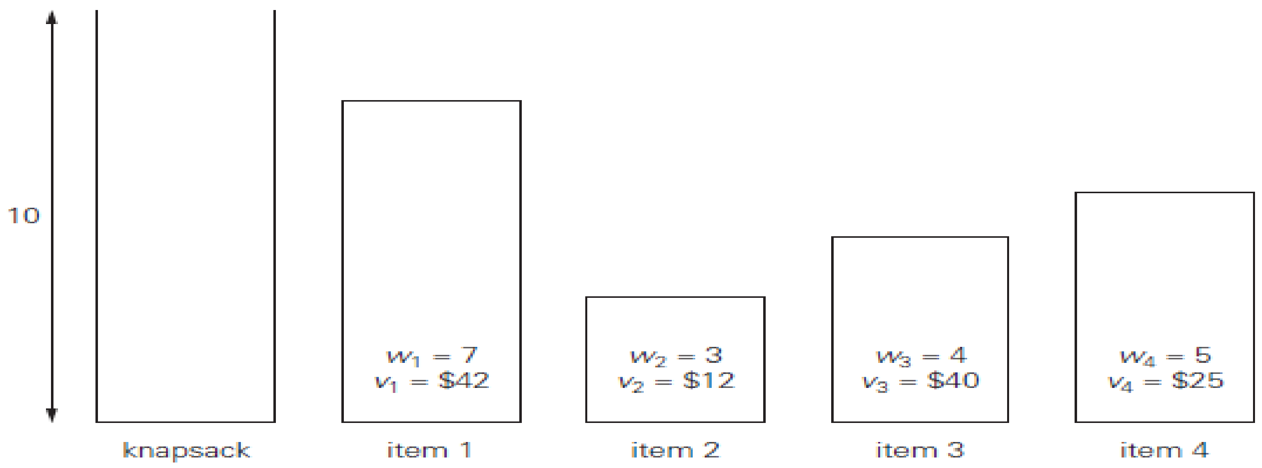


Find most valuable subset of the items that fit into the knapsack.

Knapsack Problem using Exhaustive Search

Subset	Total weight	Total value
∅	0	$ 0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$54
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	$52
{2, 4}	8	$37
{3, 4}	**9**	**$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

Knapsack of capacity = *10*



Knapsack Problem using Exhaustive Search

Class exercise:

- Given a knapsack of capacity = 16,
  Solve the knapsack problem for the following set of items.

Item	Weight	Value
1	2	$20
2	5	$30
3	10	$50
4	5	$10

Efficiency: Θ(2^n)

Subset	Total weight	Total value
{1}	2	$20
{2}	5	$30
{3}	10	$50
{4}	5	$10
{1,2}	7	$50
{1,3}	12	$70
{1,4}	7	$30
{2,3}	15	$80
{2,4}	10	$40
{3,4}	15	$60
{1,2,3}	17	Not feasible
{1,2,4}	12	$60
{1,3,4}	17	Not feasible
{2,3,4}	20	Not feasible
{1,2,3,4}	22	Not feasible

Assignment Problem using Exhaustive Search

The Assignment Problem:

- There are *n* people who need to be assigned to *n* jobs, one person per job.
- The cost of assigning person ***i*** to job ***j*** is **C[i, j].**
- Find an assignment that minimizes the total cost.

	Job 1	Job2	Job3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

Algorithmic Plan: Generate all legitimate assignments, compute their costs, and select the cheapest one.

How many assignments are there? n!

Assignment Problem using Exhaustive Search

Assignment (col.#s)	**Cost**		**Total Cost**	
<1, 2, 3, 4>	9+4+1+4	=	18	
<1, 2, 4, 3>	9+4+8+9	=	30	
<1, 3, 2, 4>	9+3+8+4	=	24	
<1, 3, 4, 2>	9+3+8+6	=	26	
<1, 4, 2, 3>	9+7+8+9	=	33	
<1, 4, 3, 2>	9+7+1+6	=	23	etc.

$$C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

Efficiency $\approx O(n!)$

For this particular instance, the optimal assignment can be found by exploiting the specific features of the number given. It is: **<2,1,3,4>**

Final Comments on Exhaustive Search

- Exhaustive-search algorithms run in a realistic amount of time only on very small instances

- In some cases, there are much better alternatives!
  - shortest paths
  - minimum spanning tree
  - assignment problem

- In many cases, exhaustive search or its variation is the only known way to get exact solution.

Thank you!

Any queries?