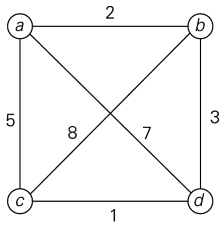
CSSE 473 – Day 6

EXHAUSTIVE SEARCH
GRAPH REPRESENTATIONS

Traveling Salesperson Problem (TSP)

n cities, length of traveling between pairs of cities

Must visit all cities (starting & ending at same place) with shortest possible distance—
i.e., find the *shortest Hamiltonian cycle*



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

Exhaustive search: how many circuits?
$(n-1)!/2 \in \Theta((n-1)!)$

Knapsack Problem

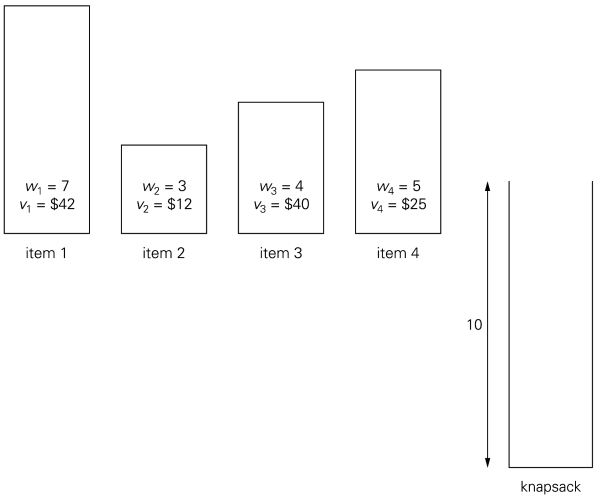
n items with given weights w_i and values v_i.

Knapsack with weight capacity *C*

Task: maximize value $\sum v_i$ while staying within knapsack capacity $\sum w_i \leq C$

Exhaustive search: how many subsets?

$2^n \in \Theta(2^n)$



Subset	Total weight	Total value
Ø	0	$ 0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$36
{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

NP-Hard Problems

E.g., TSP and Knapsack

No general polynomial-time algorithm is known to generate a solution.

However, a solution can be verified in polynomial time (P-time)

Thought not to exist ($P \neq NP$), but never been proven.

Some instances can be solved in sub-exponential time, some have approximation algorithms.

Typically us optimization algorithms such as genetic algorithms and swarm intelligence

More later in the course.

Assignment Problem

Assign *n* people to *n* jobs, one person per job

Given associated costs (person *i* on job *j*), minimize cost.

Cost matrix

	Job-> 1	2	3	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

Exhaustive search

<1, 2, 3, 4>	cost = 9 + 4 + 1 + 4 = 18
<1, 2, 4, 3>	cost = 9 + 4 + 8 + 9 = 30
<1, 3, 2, 4>	cost = 9 + 3 + 8 + 4 = 24
<1, 3, 4, 2>	cost = 9 + 3 + 8 + 6 = 26
<1, 4, 2, 3>	cost = 9 + 7 + 8 + 9 = 33
<1, 4, 3, 2>	cost = 9 + 7 + 1 + 6 = 23

etc.

Exhaustive search: how many assignments?

n!

There is a much more efficient algorithm for this problem.

Graphs

Basic ingredients: vertices & edges

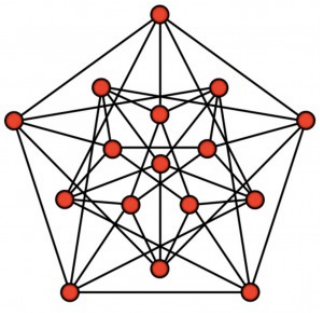
Often specified with one of these data structures:

- Adjacency matrix (good for dense graph)
- Adjacency list (good for sparse graph)

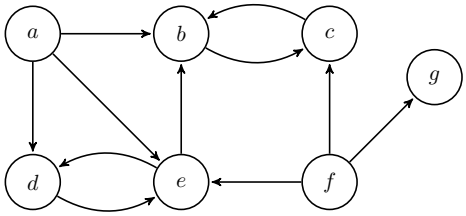
Edges can be directed/undirected

edges/vertices can have weights/costs/values/colors

many variants



Directed Graphs



How is a directed graph implemented as an…

- Adjacency matrix?
- Adjacency list?

Directed acyclic graph (dag): no directed cycles