SNS College of Engineering Coimbatore - 641107

Brute Force- Closest pair

AP/IT

Brute Force

- Straightforward
- Based on Problem statement
- Concepts involved:
  - Force
  - Just Do It

♌ Closest pair

- **Problem**: find the closest pair among *n* points in *k*-dimensional space
- **Algorithm**: Compute distance between each pair of points and identify the pair resulting in the shortest distance
  - Basic operation:

$$d = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

$$d^2 = \sum_{i=1}^{k} (x_i - y_i)^2$$

- **Efficiency**: $\Theta(n^2)$

Euclidean distance

- Distance between two points $p_i=(x_i,y_i)$ & $p_j=(x_j,y_j)$

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

Algorithm

- //find two closest points in the plane by brute force
- //Input: A list p to n (n≥2) points p1=(X1,y1)…pn = (xn,yn)
- //Output: Indices index1 & index2 of the closest pair of points

dmin← ∞

For i←1 to n-1 do

    for j ← i+1 to n do

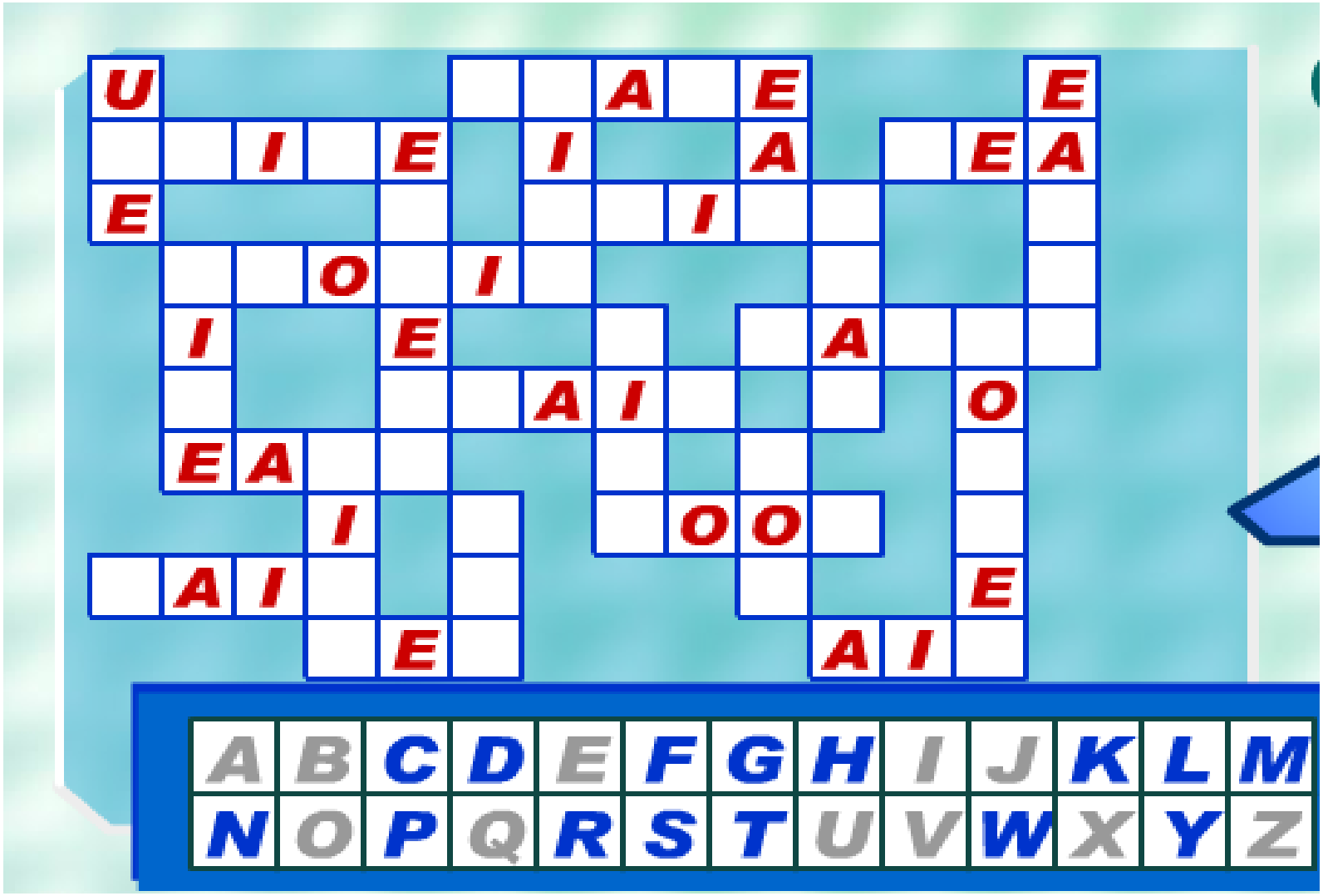
        d ← sqrt((xi-xj)2 +(yi-yj)2)

        if d < dmin

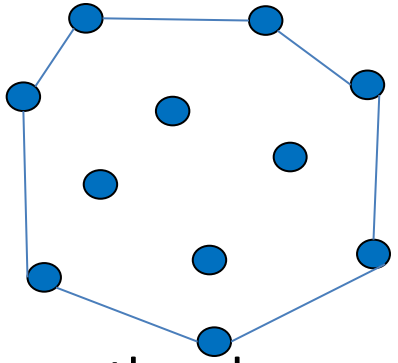
                dmin← d; index1←i; index2←j

return index1, index2

Break


U
A
E
E
I
E
I
A
E
A
E
I
O
I
I
E
A
A
I
O
E
A
I
O
O
A
I
E
A
I
E
A
I
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Convex hull problem



- Convex hull
  - Problem:

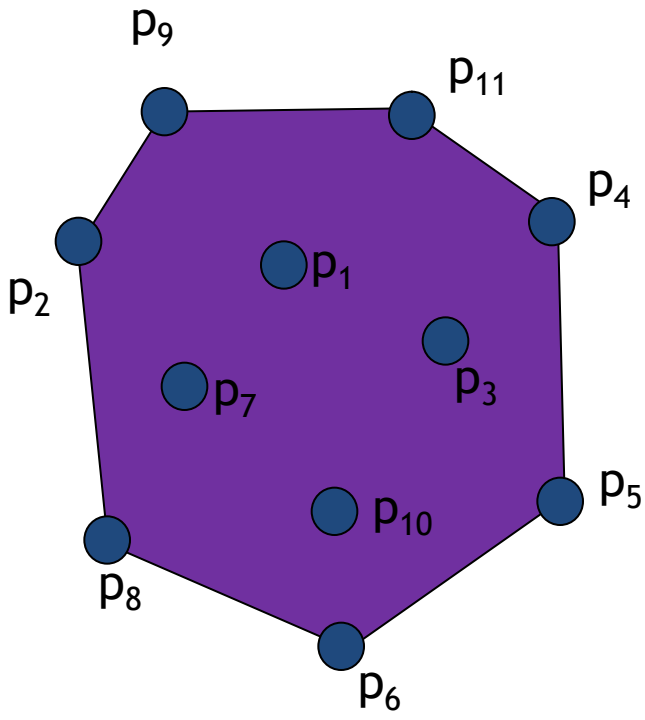
    Find smallest convex polygon enclosing *n* points on the plane

  - Convex:
    - A geometric figure with no indentations.
    - Formally, a geometric figure is convex if every line segment connecting interior points is entirely contained within the figure's interior.

Example: Convex Hull

Input: $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}$

Output: $p_2, p_9, p_{11}, p_4, p_5, p_6, p_8, p_2$



Applications

Computer visualization, ray tracing
(e.g. video games, replacement of bounding boxes)

Path finding
(e.g. embedded AI of Mars mission rovers)

Geographical Information Systems (GIS)
(e.g. computing accessibility maps)

Visual pattern matching
(e.g. detecting car license plates)

Verification methods
(e.g. bounding of Number Decision Diagrams)

Geometry
(e.g. diameter computation)

Convex hull brute force algorithm

- The straight line through two points (x1, y1), (x2, y2) in the coordinate plane can be defined by the following equation
  - $ax + by = c$

    where $a = y2 – y1$, $b = x1 – x2$, $c = x1y2 - y1x2$

- Such a line divides the plane into two half-planes: for all the points in one of them: $ax + by > c$, while for all the points in the other, $ax + by < c$.

Convex hull brute force algorithm

- Algorithm: For each pair of points p_1 and p_2 determine whether all other points lie to the same side of the straight line through p_1 and p_2, i.e. whether ax+by-c all have the same sign

- Efficiency: $\Theta(n^3)$

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.

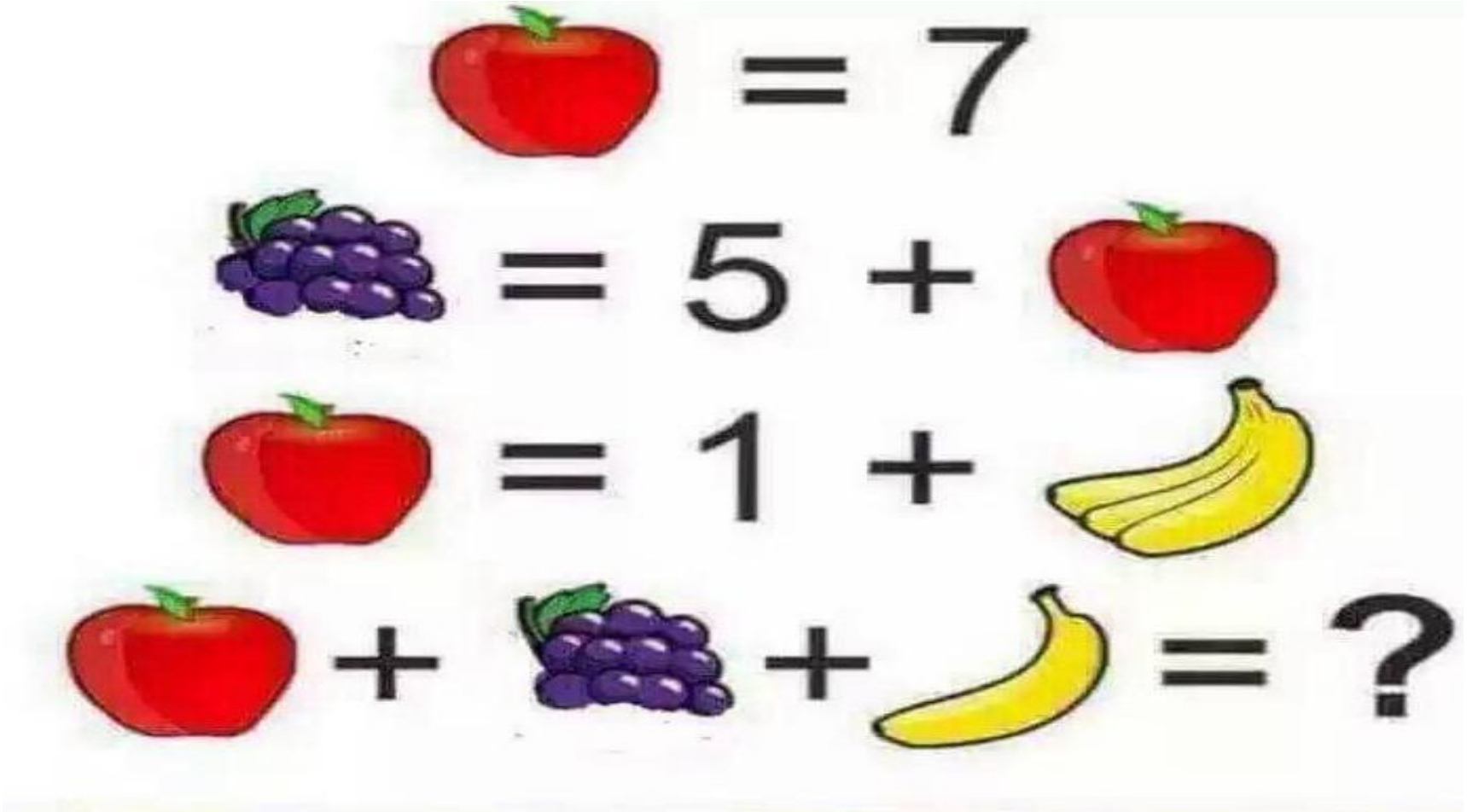
Method:

- generate a list of all potential solutions to the problem in a systematic manner .
- evaluate potential solutions one by one, disqualifying infeasible ones and, for an optimization problem, keeping track of the best one found so far
- when search ends, announce the solution(s) found

Exhaustive search

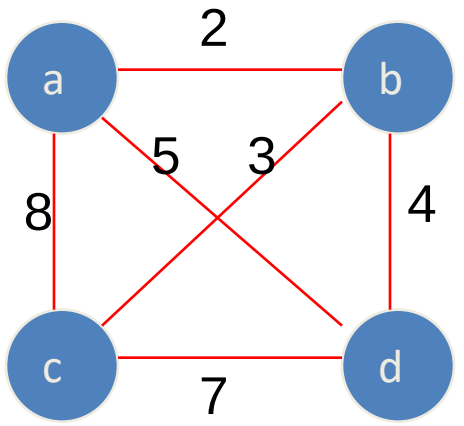
- Three important problems:
  - Travelling salesman problem
  - Knapsack problem
  - Assignment problem

Break



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
- Example:



Knapsack Problem

Given *n* items:

- weights: w_1 w_2 … w_n
- values: v_1 v_2 … v_n
- a knapsack of capacity *W*

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

Example: Knapsack capacity W=16

item	weight	value
1	2	$20
2	5	$30
3	10	$50
4	5	$10

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 0	Job 1	Job 2	Job 3
Person 0	9	2	7	8
Person 1	6	4	3	7
Person 2	5	8	1	8
Person 3	7	6	9	4

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

Activity

- 1. smallest
- 2. Θ(n^2)
- 3. simplicity
- 4. unacceptably slow