5 Brute-Force

Tuesday, September 19, 2023 11:03 AM

- Use the computer to "just go ahead and solve" problem
- Use the problems definition directly
- Usually this is the first solution.

- **PROBLEM : Power**

$$a^n = \underbrace{a * a * a * \dots a}_{n \text{ times}}$$

```
FUNCTION bfPow( a, n )
    r ← 1
    FOR i ← 1 to n-1
    {   r ← r * a
    RETURN r
```

Analysis:

basic operation: *

$$M(n) = \sum_{i=1}^{n-1} 1 = n-1 \in \Theta(n) \quad \text{Linear.}$$

- **PROBLEM : Sorting**

Given a sequence $\langle a_0, a_1, a_2, \dots a_{n-1} \rangle$

Problem

Find a permutation of the sequence such that

$$\langle a'_0 \le a'_1 \le a'_2 \le a'_3 \dots \le a'_{n-1} \rangle$$

- Solution: Selection Sort

Find the smallest element, and put it in the first place.
Find the next smallest element and put it in the second place
Find the " " " " " " " " third place
⋮ ⋮ ⋮ ⋮ ⋮ ⋮

after i iterations:

$$\langle \underbrace{a'_0, a'_1, a'_2, \dots a'_i}_{\text{Sorted.}}, \underbrace{a_{i+1}, a_{i+2}, \dots a_{n-1}}_{\text{unsorted}} \rangle$$

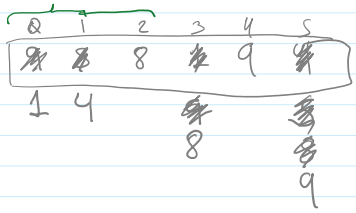
a' which is the minimum in $a_{i+1}, \dots a_{n-1}$

$$\langle \underbrace{a'_0, a'_1, a'_2, \dots a'_i, a'}_{\text{sorted}}, \underbrace{a_{i+2}, \dots a_{n-1}}_{\text{unsorted.}} \rangle$$

```
FUNCTION SelectionSort ( A[0... n-1] )
1   FOR i ← 0 TO n-2 DO
2       min ← i
3       FOR j ← i+1 TO n-1 DO
4           IF A[j] < A[min] THEN
5               min ← j
6       swap A[i] with A[min]
```

i ↑ swap A[i] with A[min]

Trace:



Analysis.

basic_operator. <

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2} (n-1-i)$$

$$\sum_{i=0}^{n-2} (n-1-i) = \underbrace{(n-1)}_{i=0} + \underbrace{(n-2)}_{i=1} + \underbrace{(n-3)}_{i=2} + (n-4) + \ldots + 3 + 2 + \underbrace{1}_{i=n-2}$$

$$= \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{1}{2}n^2 \in \Theta(n^2)$$ quadratic.

what about swapping? $S(n) = n-1 \in \Theta(n)$

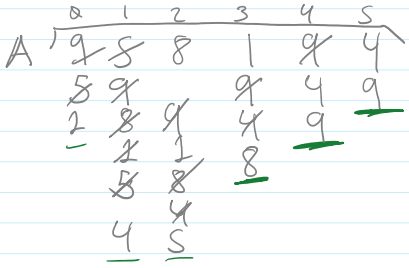
- Solution: [Bubble Sort]

repeat
{ scan array and if two consecutive elements are out of place, swap them.

```
FUNCTION BubbleSort( A[0...n-1] )
  FOR i ← 0 TO n-1 DO
    FOR j ← 0 TO n-2-i DO
      IF A[j+1] < A[j]
        swap A[j+1] with A[j]
```

Trace:

i: 1 2 3



Analysis: basic_operation. <

$$C(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-1} [(n-2-i) - 0 + 1]$$

$$= \sum_{i=0}^{n-1} (n-1-i) = \frac{(n-1)n}{2} \in \Theta(n^2)$$ quadratic.

What about swaps? = is $\Theta(n^2)$ quadratic.

- PROBLEM : Search
  - Sequential Search.

Given an array find if element x is in the array.

```
FUNCTION SeqSearch( A[0..n-1], x )
    FOR i←0 TO n-1 DO
        IF A[i]=x THEN
            RETURN True
    RETURN False.
```

Analysis.

Basic operation =

$$C(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$ Linear.

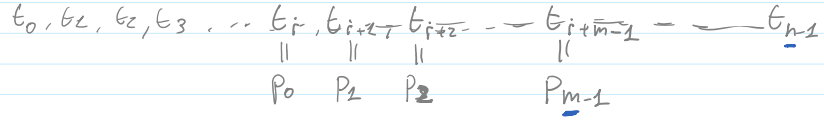
Python.

```
l.index(e)
```

```
for i in len(l):
    p = l[i:]
```

  - String Matching.

Given a large string "text" and a shorter string "pattern",
find the first occurance of pattern in text.



idea:

compare p starting at t_0.
compare p starting at t_1
" " " t_2.
until success
or
compare p starting at t_{n-m}

```
FUNCTION stringSearch( T[0..n-1], P[0..m-1])
    FOR i←0 TO n-m DO
        j←0
        WHILE j<m and P[j]=T[i+j] DO
            j←j+1
        IF j=m RETURN i
    RETURN -1
```

Trace. T= ME_THINKS_IS_A_WEASEL P= INK

```
   i
   INK
    INK
     INK
```

INK
INK
INK
jar

Analysis

Parameters m n

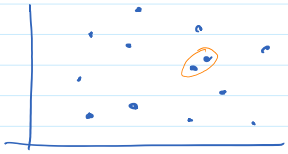
basic operation $\simeq$

$$C(n,m) = \sum_{i=0}^{n-m} \sum_{j=0}^{m-1} 1 = \sum_{i=0}^{n-m} m = (n-m+1)\,m \qquad O(n \cdot m)$$

$$m \cdot n - m^2 + m$$

- PROBLEM : Closest-Pair

Given a collection of points $P = \langle x, y \rangle$ in 2D space
find the closest 2 points



$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$$P = [(x_0, y_0), (x_1, y_1), (x_2, y_2) \ldots (x_{n-1}, y_{n-1})]$$

```
FUNCTION bf_closest_pair( P )
    d ← ∞
    FOR i ← 0 TO n-2 DO
        FOR j ← i+1 TO n-1 DO
            d ← min( d , sqrt( (P_i.x - P_j.x)^2 + (P_i.y - P_j.y)^2 ) )
    RETURN d
```

trace $[P_0 \; P_1 \; P_2 \; P_3 \; P_4 \; \ldots \; P_{n-1}]$

i

j

tip

$$\sum_{i=L}^{U} 1 = U - L + 1$$

Analyse:

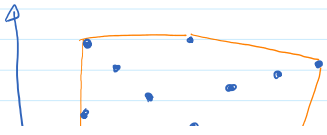
$$A(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1-i-1+1)$$

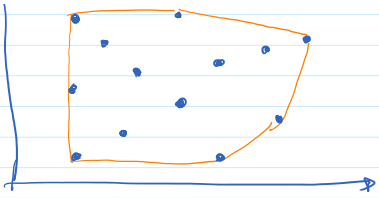
$$= \sum_{i=0}^{n-2} (n-1-i) = (n-1) + (n-2) + (n-3) + \ldots + 3 + 2 + 1$$

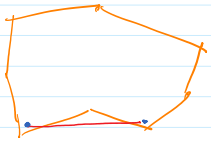
$$= \frac{n(n-1)}{2} = \in \Theta(n^2)$$

- PROBLEM : Convex Hull

find the smallest convex polygon that contains a set of points S

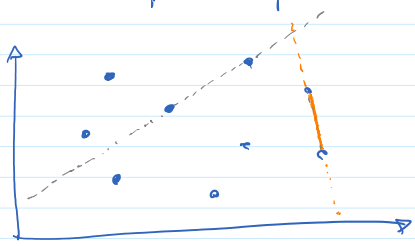






convex:
for every 2 points inside the poligon. the line between these 2 points is also inside the polygon

NOTE
- the convex hull has vertices in the input set.
"extreme points"



Algorithm draft:

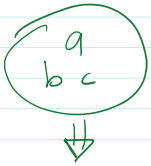
```
for every pair of points p1 , p2.
    draw line l from p1 to p2.
    for every point p3
        if all p3's are on the same side of l
            l belongs to the solution.
```

$\approx \Theta(n^3)$

- EXHAUSTIVE SEARCH

B.F. approach to problems of the following type.

Given a collection of input elements.
find a permutation / combination / subset that
- Satisfies certain constraint
and/or
- Maximizes or Minimizes a characteristic.



abc
acb
bac
bca
cab
cba

- PROBLEM (TSP) Traveling Salesman Problem

- Given a collection of cities find a tour that all cities, and returns to the starting point, minimizing the cost.

fully connected, weighted.

- The problem can be represented as a Graph where you need to find a Hamiltonian Circuit that minimizes cost.

$\langle v_0, v_1, v_2, v_3 \ldots, v_{k-1}, v_0 \rangle$

you are looking for the permutation of $v_1 \ldots v_{k-1}$ that minimizes cost.

BF Algorithm

```
cost ← ∞
for each permutation p of v1 ... vk-1
    if cost of p < cost
        keep p.
        cost ← cost of p.
```

$\approx \Theta(n!)$

- PROBLEM Knapsack Problem

Given a collection of items. $e_1, e_2 \ldots e_n$
each with a weight $w_1, w_2 \ldots w_n$
and a value $v_1, v_2, \ldots v_n$.

and a knapsack of capacity W

? Which items to load on the knapsack to maximize value?

What you need is to select the subset $\{e_1, \ldots e_n\}$ that - total weight is less than W
- maximizes value.

BF:

```
value ← 0
for every subset S of {e1, ... en}
    if weight of S ≤ W
        if value of S > value
            save S
            value ← value of S
```

$\approx \Theta(2^n)$

a b c d

{}	0 0 0 0
{a}	0 0 0 1
{b}	0 0 1 0
{c}	
{d}	
{a, b}	0 0 1 1
{a, c}	0 1 0 1
⋮	
{a, b, c, d}	1 1 1 1

- ? Is there a way to lower complexity?

Nobody knows.

NP-Hard. ∴ No polynomial algorithm is known.

- Common belief: "they don't exists"

- PROBLEM Assignment Problem
  - Given a list of n people, and a list of n jobs.
  - and the cost of person i doing job j

  find the assignment of jobs to people that minimizes cost.

e.g.

	Job1	Job2	Job3	Job4
Kevin	9	2	7	8
Chubi	6	4	3	7
John	5	8	1	8
Jake	7	6	9	4

⟨1, 2, 3, 4⟩

- You need to select one item per row.
  - making sure that no two selected items are in the same column
  - minimizing total cost.

BF:

```
cost ← ∞
for every permutation p of ⟨1...n⟩
    if cost of p < cost
        save p
        cost ← cost of p
```

≈ Θ(n!)

NOTE: Better algorithm do exists.

Just because the Brute Force approach is exponential (2^n or $n!$) does not means that all hope is lost and better algorithms don't exists.

- Exhaustive Search on a Graph:

  $G = \langle V, E \rangle$ V: vertices E: edges.

  - Consider Undirected graphs.
  - Exhaustively Searching a graph "testing" all nodes.

- Depth-First Search

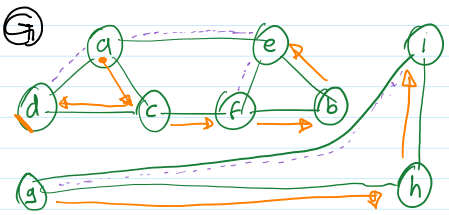
Algorithm:

```
FUNCTION DFS(G)
    mark each vertex V as "unvisited"
    count ← 0
    FOR EACH vertex v in V DO
        IF v is marked as unvisited
            dfs(v)

FUNCTION dfs(v)
    count ← count + 1
    mark v with count
```

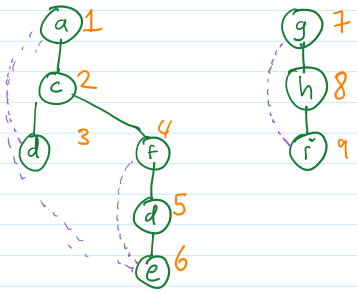
FOR EACH vertex u adjacent to v DO
  IF u is marked as unvisited
    dfs(u)

Trace:



DFS trace

count ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9



tree-edges
back-edges.

Applications:

- Connectivity
- Cycle Checking.
- path existance.
- Serializing the graph.

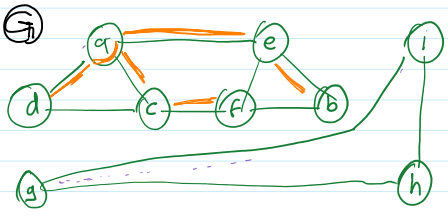
- Breadth-First Search

Algorithm.

```
FUNCTION   BFS(G)
    mark every vertex as unvisited
    FOR EACH vertex v
        IF v is unvited
            bfs(v)

FUNCTION bfs(v)
    count ← count + 1
    mark v with count
    enqueue v
    WHILE queue is not empty DO
        u ← front of queue
        dequeue
        FOR Each vertex w adjacent to u DO
            IF w is unmarked THEN
                count ← count + 1
                mark w
                enqueue w.
```

Trace:

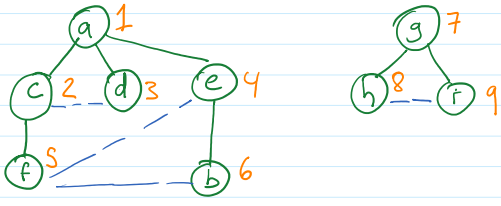


BFS

Q[~~g~~ ~~h~~ ~~r~~

count[~~1~~

u[i



tree-edges.
cross-edges.

1 :- the starting point.
2,3,4 :- vertices one edge removed.
5,6 :- vertices two edges removed.

Applications:

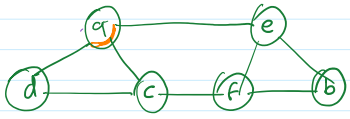
+ BFS finds paths with the least edges
- Same kind of applications as D.F.S.

- Complexity:

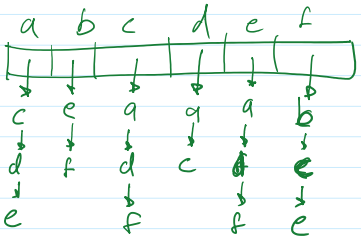
The Data Structure you use matters.

Adjacency Matrix

	a	b	c	d	e	f
a	1	0	1	1	1	0
b	0	1	0	0	1	1
c	1	0	1	1	0	1
d	1	0	1	1	0	0
e	1	1	0	0	1	1
f	0	1	1	0	1	1



Adjacency List



$\Theta(|V|^2)$

$\Theta(|V|+|E|)$

—EOF—