1 First Class

Tuesday, August 20, 2024 11:43 AM

Back to the Basics:

Q: What is Computer Science?

"Problems:"

- What Problems can be solved in a systematic / automatic way and How?

Algorist vs. abacist (1508)

- To Solve a problem:

1) Abstract the problem and represent it as a mathematical object

2) Develop Algorithm on Mathematical Object

3) Identify important operations on the Mathematical Object.

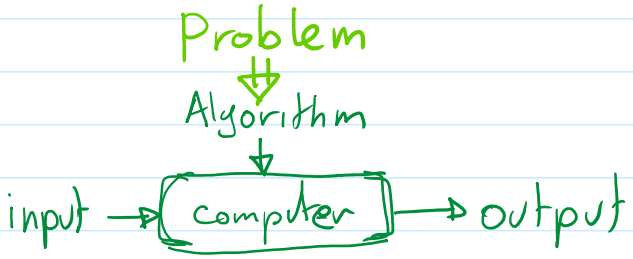
3) Identify important operations on the Mathematical Object.
Choose an appropiate Data Structure.
4) Implement.
5) ?
6) Profit....

We will take the Algorithm. → Correctness •
→ Efficiency •

DEF:
An algorithm is a sequence of unambiguous instructions for solving a problem.

unambiguous:- not subject to interpretation
do not require background knowledge.

solving:- obtaining a required output
from a specified input



Important points:
- the non-ambiguity of each step cannot be compromised.
- the range of the input must be specified carefully
- the same algorithm can have multiple

specified carefully

- the same algorithm can have multiple representations
- there may be more than one algorithm to solve the same problem.
- Algorithms for the same problem, based on different ideas can have different characteristics.
  - differences in resource needs → Time. / Memory

E.G.

Problem: gcd(m,n) - the biggest number that divides both m and n.

Euclid's Algorithm:

Let $m > n$

form 1:
1) if n is equal to 0, answer m and stop.
2) Divide m by n, and assign the remainder to r
3) Assign n to m and r to n
4) Repeat from step 1.

m	n	r
62	24	14
24	14	10
14	10	4
10	4	2
4	2	0
2	0	

form 2:

```
euclid(m, n)
    while n ≠ 0 do
        r ← m mod n
        m ← n
        n ← r
    return m.
```

← assignment =

= = ==

←	assignment	=
=		==
		===
and		&&
or.		||

```
gcd2(m, n)
  t ← n:
  while t > 0 do
      if m mod t = 0 and n mod t = 0 then
          return t
      t ← t-1
  return t.
```

gcd3(m, n):

Let m's prime factorisation be: $p_1^{q_1} * p_2^{q_2} * \ldots * p_i^{q_j}$

Let n's prime factorisation be $r_1^{s_1} * r_2^{s_2} * \ldots * r_l^{s_l}$

Multiply the common prime factors.

75 ~ 300 A.D.

- More on Algorithm Problem Solving

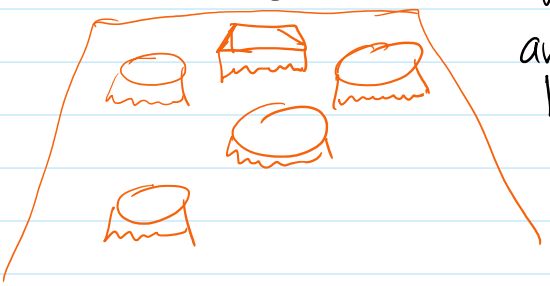
1):- Understand the Problem.

-assumptions.- "given an array of numbers"

1) Understand the problem.

- assumptions & clarity: "given an array of numbers"
  - integers? decimal?
- spec. of the input:
  - "in the array a number appears only once"
  - "numbers range from 0 to 256"
- abstraction: the removal of unnecessary information
  - the modeling of the problem as a mathematical object.

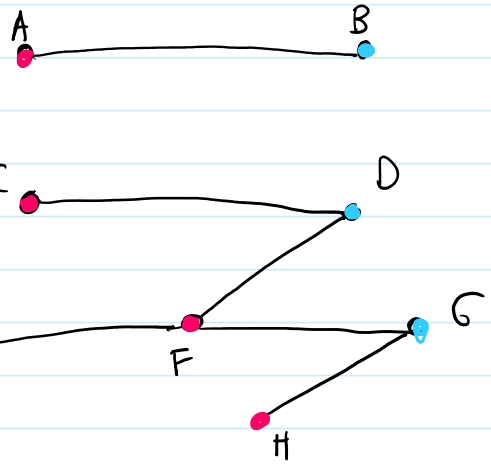
Ex: Romeo and Juliet are getting married.
Family members hate each other.
Therefore some family members cannot sit together in the same table at the reception.
How to sit the family members? which tables to assign them?



we have a list of guest an for each guest who do they hate.

Petro hates A, B, C.
John hates X, Y, Z...

Graph:



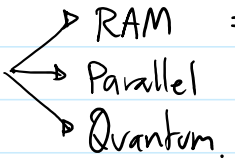
Partition a graph.

{A, C, F, H}
{B, D, E, G}

2) Decide on Computational Needs & constraints.
- Speed & Memory

4) Decide on Computational Needs & constraints.

- Speed & Memory

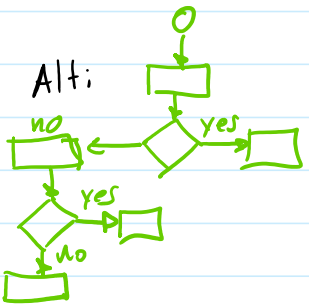


= { - sequential: one instruction at a time
- Memory with random access

- exact or approximate solution.

3) Specify Algorithm

- Pseudo code.



4) Show Correctness:-

- Empirical Argument = Test.
- Rigorous Argument (Proof):- • output will be correct for any valid input.
  • The Algorithm terminates.

5) Analyse Algorithm.

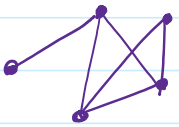
- resource utilization → time (number of steps)
  → space. (size of memory needed)
- use of randomness
- "stability"

- Important Problem Types:
  - Sorting
    rearranging a list of elements in non-decreasing order.
  - Searching
    find an item in a collection

Searching
find an item in a collection
- representation of the collection.

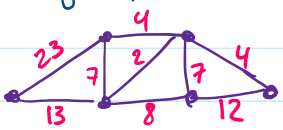
- String Processing.
  e.g. finding a substring - search & replace
  - match patterns.

- Graph Problems
  Graph $G = \langle V, E \rangle$



  - traversals.
  - Finding paths
  - Finding cycles
  - adding / removing edges
  - Partitioning.

If weighted

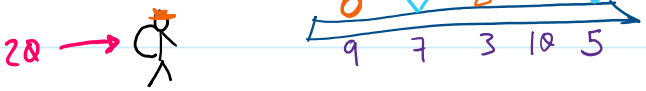


- find lowest-cost path

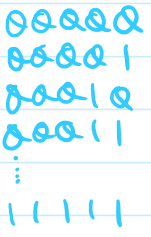
(TSP) Travelling Salesman problem.
- find a cycle that visits all nodes in the graph of minimum cost.

- Combinatorial Problems
  find a combination or permutation of items that satisfies a criteria, or maximize / minimize a property.

(Knapsack)



How many candidate selections are there?



$2^5 = 32$. $2^6 = 64$ $2^{16} = 65k$
$2^{32} = 4$ billion....

these problems are HARD.
- no efficient algorithm is known

These problems are HARD.
- no efficient algorithm is known
- there is no proof that an efficient algorithm does not exist

¿ P = NP ?

- Geometric Problems

  Problems on points, lines & polygons.

  e.g – closest pair.
  - convex hull
  - line–polygon / polygon–polygon intersection.

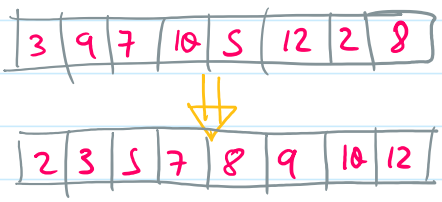
- Numerical Problems (CS-3200)
  - Numerical Methods (Calculus problems)
  - solving equations, systems of equations
  - Computing integrals
  - Roots of polynomials.
  - Tension networks.
  - approx solutions only.

- Important Algorithm Design Techniques:

1) Brute Force & Exhaustive search.

   list and generate all "candidate solutions" and then test each of them.

e.g. Sorting.

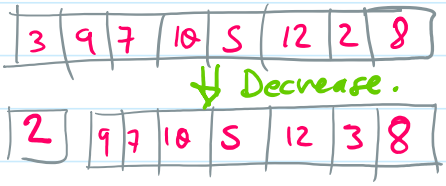


2) Decrease and Conquer.

2) Decrease and Conquer.

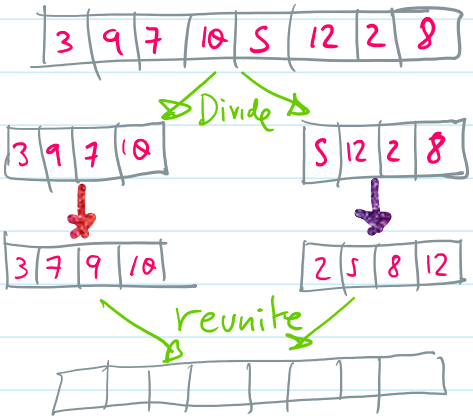
- reduce the size of the problem a little-bit
  solve the smaller problem

eg.



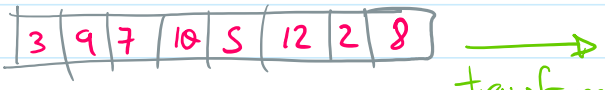
3) Divide and Conquer

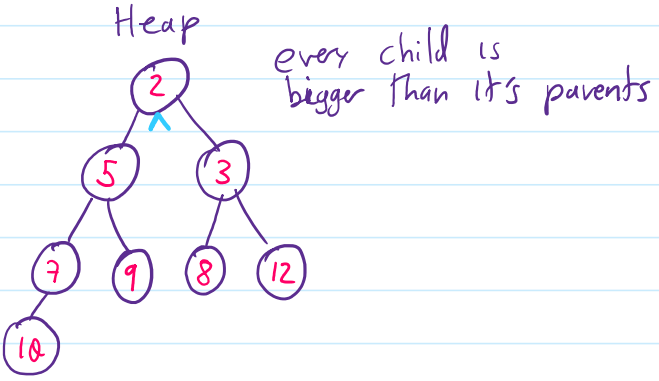
- split problem into subproblems of roughly the same size.

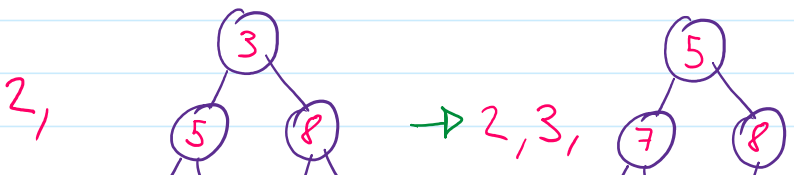


4) Transform & Conquer.

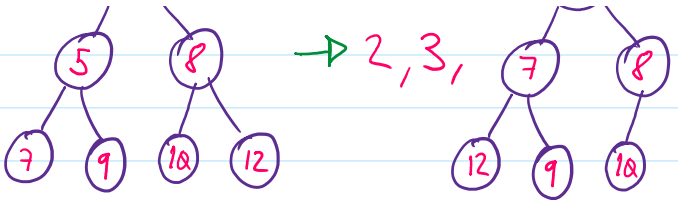
transform your problem into another one, or into a different data organization and solve the new problem.







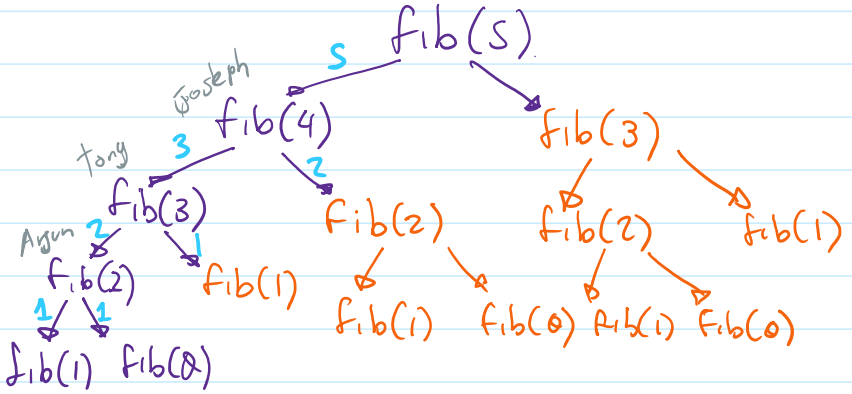
2,



5) Dynamic Programming.

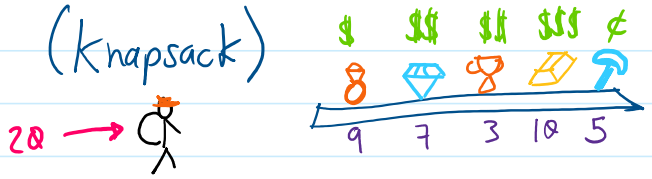
A technique to improve efficiency of recursive algorithms by remembering solutions to subproblems.

- fib(n) = f(n-1) + f(n-2)
- fib(0) = 1
  fib(1) = 1



6) Greedy techniques.
(in Combinatorial problems)

(Knapsack)



You build candidate solution, by trying first to maximize early gains.

-EOF.