Complexity

Decidable problems

Computationally solvable in principle

Not in practice

Resourse Consumption of TIME SPACE

Let M be a deterministic ™ that halts on all inputs

The running time of M is a function f :N→ N where f(n) is max running time of M on input length n

Worst case v average case

Polynomial And Exponential

Algorithms are classified as:

0(1) runtime is independent of the size of problem, n

0(log n) occurs when a big problem is solved by transforming it into smaller size (logarithmic)

0(n) occurs when each element of the problem requires small amount of processing(linear)

0(n log n) occurs when problem is broken into smaller subproblems, solving them independently and combining solutions(linearithmic)

0(n2) occurs when an algorithm processes all pairs of elements(quadratic)

0(2n) exponential run time to be avoided, brute force approach.(exponential)

DTIME f(n) is a language decided by 0(f(n)) time deterministic ™

If there is an upper bound pf Nk where K>0 then it runs in polynomial time

The class EXPTIME is the set of languages decidable in an exponential time on deterministic TMs. Problems are intractable

Complexity Class P

Given directed graph G

Nodes s and t

Is there a path from s to t

PATH - {(g,s,t)|g has directed path from s to t}

A possible solution

Let m be the number of nodes in G

Any path from s to t need not repeat nodes

Examine each path in G of length <= m

Check if it goes from s to t

What is the complexity of this algorithm?

There are Mm possible paths so this algorithm is exponential

PATH solution

Place mark on S

Repeat until no additional nodes are marked

Scan edges of G

If there is an edge(a,b) where a is marked and b is unmarked then mark b

If t is marked accept,otherwise reject

Complexity of PATH

Stage 1 and 3 run once

Stage 2 runs m times because each time it marks a new node

Total number of stages is polynomial

Time complexity is therefore polynomial

Complexity Class NP

Non-deterministic Turing Machines

The runtime of ND ™ is a function where F(N) is the max number of steps that m uses on any branch of computation on input size n

P C= NP C= PSPACE C= EXPTIME

If P differs from NP then distinction between P and NP-P is meaningful and important

P is tractable

NP - P intractable

Until we prove that P!= NP there is no hope of proving that a language lies in NP-P

Cook-Levin Theorem: P=NP iff SAT E-- P

NP COMPLETENESS

Reduced a problem to another which is known to belong to that class

A language L is NP complete if it satisife that L is in NP and every L’ os in NP and polynoiaml time reducible to L

If L1 is NP complete and L1 is polynomial time reducible to L2 where L2 E-- NP then L2 is NP-Complete

The class of NP- complete languages are hardest languages in NP least likely to be in P

If any NP-Complete L E-- P then NP = P

HAMPATH TSP CLIQUE SUBSET SUM are also NP complete