Issue: many interesting problems seem to have only “brute force” algorithms of exponential time

(p. 292) HAMPATH = \{ (G,s,t) | G is graph with Hamiltonian path from s to t \}

- Hamiltonian Path from s to t goes thru every other vertex
- Easy decider by variant of algorithm for PATH
  - Put PATH in a loop to generate all possible paths
  - With test after each one to verify if path is Hamiltonian
  - Verifier runs in polynomial time
    - Keep a list of vertices
    - Follow path
    - Cross off matching vertex as each step
    - At end, if all vertices crossed off, accept; else reject
  - But the generator from PATH is exponential!

- No known polynomial HAMPATH algorithm

(p. 293) COMPOSITES = \{ x | x=pq, for p,q>1 \}

- Verifier is trivial
- No known polynomial generator

(p. 293) Not all problems have polynomial verifiers
- e.g. \text{not}(HAMPATH)
(p. 293) Definition 7.18. **A verifier for language A is an algorithm V,** where \( A = \{ w \mid V \text{ accepts } <w,c> \text{ for some string } c \} \)

- c is “extra information” called a **certificate** or **proof**
- e.g. for above problems, c is a “guess” of answer
  - HAMPATH: a path that is a Hamiltonian
  - COMPOSITES: a divisor
- The ones that work are solutions to problem
- Equivalent to stating “a solution exists”
- Time for V expressed as a function of w
  - **Polynomial Time Verifier** for V runs in polynomial time
- Language A is **polynomially verifiable** if it has a polynomial time verifier

p. 294: example of NTM \( N_1 \) for HAMPATH that works in “nondeterministic polynomial time”

- Remember time of NTM is time used by longest branch
- Step 1 “generates” a solution (magically) as a series of vertex #s
- Step 2 ensures no repeats
- Step 3 ensures starts at s and ends at p
- Step 4 is a polynomial verifier that checks edges exist
• (p. 294) Definition 7.19: **NP is class of languages that have polynomial time verifiers**
  • HAMPATH and COMPOSITES both in class NP
• (p, 294) **Theorem 7.20** Language A is in NP iff it is decided by some polynomial time NTM
  • Proof: if A in NP then decided by NTM in polynomial time
    • Let V be matching polynomial verifier for A of O(n^k)
    • Define NTM N as follows: For input w of length n,
      • Nondeterministically select string c of length ≤ n^k
      • C is “solution”
      • Run V on <w,c>
      • If V accepts, accept, else reject
  • Proof: if Poly time NTM N exists, then A is in NP
    • V constructed on <w,c> as follows
      • Simulate N on input w, treating each symbol of c as description of NTM choice to make at each step
      • If this branch accepts, accept, else reject
• For HAMPATH
  • W is <G,s,t>
  • c is a path
• (p. 293) \( \text{NTIME}(t(n)) = \{L | L \text{ is language decided by some } O(t(n)) \text{ time NTM} \} \)

• **NP = \( \bigcup_k \text{NTIME}(n^k) \) for all \( k \)**

• (p. 295) **CLIQUE = \{<G,k> | G undirected graph with k-clique\} in in NP**
  
  • k-clique = set of k vertices with edges between each pair of vertices in set

• (p. 296) Proof by demonstrating polynomial time verifier

• (p. 297) **SUBSET-SUM = \{<S,t> | S = \{x_1, \ldots x_k\} and for some \{y_1, \ldots, y_l\} subset of S and }$$\Sigma y_i = t}$$

• (p. 299) **SAT =\{<wff> | wff a satisfiable Boolean formula\}**
  
  • **wff** is well-formed-formula constructed from
    
    • Boolean variables
    
    • Boolean operations AND, OR, NOT

  • **Satisfiability**: test if there is a substitution of 0s and 1s to variables that makes the wff true

• Summary:
  
  • P = class that can be **decided** quickly
  
  • NP = class that can be **verified** quickly

• **Biggest question in CS: Is P = NP, or P a subset of NP?**
  
  • Is there a language in NP that is not in P?