Polynomial vs. Exponential

- Big difference
- \( n^3 : n = 1000 \Rightarrow 10^9 \)
- \( 2^n : n = 1000 \Rightarrow 2^{1000} = 10^{1000 \log_{10} 2} \approx 10^{300} \gg 10^9 \)
- An algorithm with such complexity is not practical
**Definition 7.2**

$P$: decidable languages in polynomial time on a deterministic (single-tape) TM

$$P = \bigcup_k \text{TIME}(n^k).$$

- How important this is?
  - $P$: “roughly” corresponds to problems solvable on a computer
PATH problem I

\[
\text{PATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph s.t. } \exists \text{ path from } s \text{ to } t \}\}

- Example:
There is a path from $s = 1$ to $t = 3$

- We will prove that $\text{PATH} \in P$

- Let’s start with a brute force way
  1. $m$: # nodes
  2. $|\text{path}| \leq m$
  3. $\# \text{paths} \leq m^m$
  4. sequentially check if one has $s$ to $t$

- the cost is exponential

- A polynomial algorithm
  
  input $\langle G, s, t \rangle$, $G$ includes nodes and edges
PATH problem III

1. mark \( s \)
2. repeat until no new node can be marked
   scan all edges, if for an edge \( \langle a, b \rangle \):
   - \( a \) is marked but \( b \) is not \( \Rightarrow \) mark \( b \)
3. \( t \) marked \( \Rightarrow \) accept
   otherwise \( \Rightarrow \) reject

- \# of steps in the main loop: at most \( m \) (if no newly marked, stop)
- at each step, need to scan \( \# \text{edges} = m^2 \)
- cost to mark a node: polynomial
- whole algorithm: polynomial
Relatively Prime I

- $x, y$ are relatively prime if they have no common (> 1) factors
- Example: 10 and 21
  
  $10 = 2 \times 5, 21 = 3 \times 7$

- Example: 10 and 22
  
  $10 = 2 \times 5, 22 = 2 \times 11$

  They are not relatively prime
- Problem: test if two numbers are relatively prime
Euclidean Algorithm I

- It can be used to find gcd (greatest common divisor)
- Example: $\gcd(18,24)=6$
- We have
  \[
  \gcd(x, y)=1 \iff x, y \text{ relatively prime}
  \]
- Algorithm: input $\langle x, y \rangle$
  1. Repeat if $y \neq 0$
     \[
     x \leftarrow x \mod y
     \]
     exchange $x$ and $y$
  2. Output $x$
Euclidean Algorithm II

- The output is the gcd
- Note that in the beginning we don’t need $x \geq y$

If $x < y$, then

$$x = x \mod y$$

and

$$(x, y) \text{ becomes } (y, x)$$
Euclidean Algorithm III

- Why this works

\[ 18 = ab \]
\[ 24 = ac \]
\[ 24 = 18d + e \]
\[ ac = abd + e \]
\[ e = a(c - bd) \]
\[ a \mid 24 - 18d \]

- Is this algorithm polynomial?
- At each iteration, \( x \) or \( y \) reduced at least by half
Euclidean Algorithm IV

- If \( x > y \)
  
  \[
  x \mod y \leq x/2
  \]

  Proof

  if \( x/2 \geq y \), \( x \mod y \leq y \leq x/2 \)

  if \( x/2 < y \), \( x \mod y = x - y \leq x/2 \)

- Therefore,

  \[
  \#\text{iterations} \leq 2 \max(\log_2 x, \log_2 y) = O(n)
  \]

  \( n \): length of input (\( x \) and \( y \) are stored as bit strings), \( \log_2 x + \log_2 y \)
Euclidean Algorithm V

- Each iteration
  
  \[ x \mod y: \text{polynomial} \]
  
  see: 1100011 \(\%\) 101

  \#digit \(\leq O(n)\): each digit \(\leq O(n)\)

  exchange \(x\) and \(y\): polynomial
Th 7.16 I

- Context-free language $\in P$
- Proof omitted