Polynomial-time computations

- An algorithm has time complexity $O(T(n))$ if it runs in time at most $cT(n)$ for <u>every</u> input of length n.
- An algorithm is a polynomial-time algorithm if its time complexity is $O(p(n))$, where $p(n)$ is polynomial in n.

Polynomials

- If f(n) = polynomial function in n,

  then $f(n) = O(n^c)$, for some fixed constant c
- If f(n) = exponential (super-polynomial) function in n,

  then $f(n) = \omega(n^c)$, for any constant c
- Composition of polynomial functions are also polynomial, i.e.,

  f(g(n)) = polynomial if f() and g() are polynomial
- If an algorithm calls another polynomial-time subroutine a polynomial number of times, then the time complexity is polynomial.

The class $\mathcal{P}$

- A problem is in $\mathcal{P}$ if there exists a polynomial-time algorithm that solves the problem.
- Examples of $\mathcal{P}$
  - ***DFS:*** Linear-time algorithm exists
  - ***Sorting:*** O(n log n)-time algorithm exists
  - ***Bubble Sort:*** Quadratic-time algorithm $O(n^2)$
  - ***APSP:*** Cubic-time algorithm $O(n^3)$
- $\mathcal{P}$ is therefore a class of problems (not algorithms)!

The class $\mathcal{NP}$

- A problem is in $\mathcal{NP}$ if there exists a non-deterministic polynomial-time algorithm that solves the problem.
- A problem is in $\mathcal{NP}$ if there exists a (deterministic) polynomial-time algorithm that verifies a solution to the problem.
- All problems in $\mathcal{P}$ are in $\mathcal{NP}$

TSP: Traveling Salesperson Problem

- **Input:**
  - Weighted graph, *G*
  - Length bound, *B*
- **Output:**
  - Is there a traveling salesperson tour in *G* of length at most *B*?
- Is TSP in $\mathcal{NP}$?
  - YES. Easy to verify a given solution.
- Is TSP in $\mathcal{P}$?
  - OPEN!
  - One of the greatest unsolved problems of this century!
  - Same as asking: **Is $\mathcal{P} = \mathcal{NP}$?**

So, what is *NP-Complete*?

- *NP-Complete* problems are the “hardest” problems in *NP*.
- We need to formalize the notion of “hardest”.

Terminology (Cont'd)

- Decision Problems:
  - Are problems for which the solution set is {yes, no}
  - Example: Does a given graph have an odd cycle?
  - Example: Does a given weighted graph have a TSP tour of length at most B?
- Complement of a decision problem:
  - Are problems for which the solution is "complemented".
  - Example: Does a given graph NOT have an odd cycle?
  - Example: Is every TSP tour of a given weighted graph of length greater than B?
- Optimization Problems:
  - Are problems where one is maximizing (or minimizing) some objective function.
  - Example: Given a weighted graph, find a MST.
  - Example: Given a weighted graph, find an optimal TSP tour.
- Verification Algorithms:
  - Given a problem instance i and a certificate s, is s a solution for instance i?

Terminology (Cont'd)

- Complexity Class *𝒫* :
  - Set of all problems *p* for which polynomial-time algorithms exist.
- Complexity Class *𝒩𝒫* :
  - Set of all problems *p* for which polynomial-time verification algorithms exist.
- Complexity Class *co-𝒩𝒫* :
  - Set of all problems *p* for which polynomial-time verification algorithms exist for their **complements**, I.e., their complements are in *𝒩𝒫.*

Terminology (Cont'd)

- **Reductions:** $p_1 \rightarrow p_2$
  - A problem p_1 is reducible to p_2, if there exists an algorithm R that takes an instance i_1 of p_1 and outputs an instance i_2 of p_2, with the constraint that the solution for i_1 is YES if and only if the solution for i_2 is YES.
  - Thus, R converts YES (NO) instances of p_1 to YES (NO) instances of p_2.
- Polynomial-time reductions: $p_1 \xrightarrow{P} p_2$
  - Reductions that run in polynomial time.

- If $p_1 \xrightarrow{P} p_2$, then
  - If p_2 is easy, then so is p_1. $p_2 \in \mathcal{P} \Rightarrow p_1 \in \mathcal{P}$
  - If p_1 is hard, then so is p_2. $p_1 \notin \mathcal{P} \Rightarrow p_2 \notin \mathcal{P}$

The problem classes and their relationships