COMP 3170 - Analysis of Algorithms & Data Structures

Shahin Kamali

Computational Complexity

CLRS 34.1-34.4

University of Manitoba
Most algorithms you have seen have running times $\Theta(\log n)$ (e.g., binary search), $\Theta(n)$ (e.g., searching in a linked list), $\Theta(n \log n)$ (e.g., merge-sort), $\Theta(n^2)$ (e.g., bubble-sort), $\Theta(n^3)$ (e.g., matrix multiplication), etc.
Polynomial Algorithms

- Most algorithms you have seen have running times $\Theta(\log n)$ (e.g., binary search), $\Theta(n)$ (e.g., searching in a linked list), $\Theta(n \log n)$ (e.g., merge-sort), $\Theta(n^2)$ (e.g., bubble-sort), $\Theta(n^3)$ (e.g., matrix multiplication), etc.

- The running time of all these algorithms can be bounded by some polynomial function, e.g., $n^5$. 
Polynomial Algorithms

- Most algorithms you have seen have running times $\Theta(\log n)$ (e.g., binary search), $\Theta(n)$ (e.g., searching in a linked list), $\Theta(n \log n)$ (e.g., merge-sort), $\Theta(n^2)$ (e.g., bubble-sort), $\Theta(n^3)$ (e.g., matrix multiplication), etc.

- The running time of all these algorithms can be bounded by some polynomial function, e.g., $n^5$.

- A **Polynomial Algorithm** has running time $O(n^c)$ on input size of $n$, where $c$ is a constant independent of $n$
  - E.g., $O(n), O(n^2), O(n^3), O(n^{2018})$.
  - Also $O(1), O(\alpha(n)), O(\log n), O(n \log n), O(\sqrt{n}), O(n^{3/2})$, etc.
Polynomial Algorithms

- Most algorithms you have seen have running times $\Theta(\log n)$ (e.g., binary search), $\Theta(n)$ (e.g., searching in a linked list), $\Theta(n \log n)$ (e.g., merge-sort), $\Theta(n^2)$ (e.g., bubble-sort), $\Theta(n^3)$ (e.g., matrix multiplication), etc.

- The running time of all these algorithms can be bounded by some polynomial function, e.g., $n^5$.

- A **Polynomial Algorithm** has running time $O(n^c)$ on input size of $n$, where $c$ is a constant independent of $n$.
  - E.g., $O(n)$, $O(n^2)$, $O(n^3)$, $O(n^{2018})$.
  - Also $O(1)$, $O(\alpha(n))$, $O(\log n)$, $O(n \log n)$, $O(\sqrt{n})$, $O(n^{3/2})$, etc.

- A function is **super polynomial** if $f(n) \in \omega(n^c)$ for all $c$. 

COMP 3170 - Analysis of Algorithms & Data Structures
Polynomial Algorithms

- Most algorithms you have seen have running times $\Theta(\log n)$ (e.g., binary search), $\Theta(n)$ (e.g., searching in a linked list), $\Theta(n \log n)$ (e.g., merge-sort), $\Theta(n^2)$ (e.g., bubble-sort), $\Theta(n^3)$ (e.g., matrix multiplication), etc.

- The running time of all these algorithms can be bounded by some polynomial function, e.g., $n^5$.

- A **Polynomial Algorithm** has running time $O(n^c)$ on input size of $n$, where $c$ is a constant independent of $n$
  - E.g., $O(n)$, $O(n^2)$, $O(n^3)$, $O(n^{2018})$.
  - Also $O(1)$, $O(\alpha(n))$, $O(\log n)$, $O(n \log n)$, $O(\sqrt{n})$, $O(n^{3/2})$, etc.

- A function is **super polynomial** if $f(n) \in \omega(n^c)$ for all $c$.
  - E.g., $2^n$, $3^n$, $n!$, $n^n$, etc.
Exhaustive Search

- Many problems have an exponential number of possible solutions.
- An algorithm which applies an exhaustive search on the solution space will eventually find a solution.
- The time will be proportional to the size of solution space in the worst case, i.e., it will be super-polynomial.
  - This is not good!
Exhaustive Search

- Many problems have an exponential number of possible solutions.
- An algorithm which applies an exhaustive search on the solution space will eventually find a solution.
- The time will be proportional to the size of solution space in the worst case, i.e., it will be super-polynomial.
  - This is not good!
  - For many problems, we have failed to do much better.
Hamiltonian Path

Instance: a graph $G$ with vertex set $V$ and edge set $E$.

Question: Does there exist a path in $G$ that visits every vertex in $V(G)$ exactly once along a sequence of edges in $E(G)$?
Hamiltonian Path

Instance: a graph $G$ with vertex set $V$ and edge set $E$.

Question: Does there exist a path in $G$ that visits every vertex in $V(G)$ exactly once along a sequence of edges in $E(G)$?
Exhaustive Search for HP

- Try all paths and check whether the sequence of edges exist in $G$
- In other words, try all permutations of vertices
  - $v_1, v_2, v_3, v_4, \ldots, v_n$
  - $v_2, v_1, v_3, v_4, \ldots, v_n$
  - $\ldots$

There are $n!$ different paths. Some paths are redundant, e.g., $v_1, v_2, \ldots, v_n$ is the same as $v_n, v_{n-1}, \ldots, v_1$. Regardless, the number of distinct paths is still $\Theta(n!)$.
Exhaustive Search for HP

- Try all paths and check whether the sequence of edges exist in $G$
- In other words, try all permutations of vertices
  - $v_1, v_2, v_3, v_4, \ldots, v_n$
  - $v_2, v_1, v_3, v_4, \ldots, v_n$
  - $\ldots$
- There are $n!$ different paths
  - Some paths are redundant, e.g., $v_1, v_2, \ldots, v_n$ is the same as $v_n, v_{n-1}, \ldots, v_1$.
  - Regardless, the number of distinct paths is still $\Theta(n!)$. 
Exhaustive Search for HP

- Try all paths and check whether the sequence of edges exist in $G$
- In other words, try all permutations of vertices
  - $v_1, v_2, v_3, v_4, \ldots, v_n$
  - $v_2, v_1, v_3, v_4, \ldots, v_n$
  - $\ldots$
- There are $n!$ different paths
  - Some paths are redundant, e.g., $v_1, v_2, \ldots, v_n$ is the same as $v_n, v_{n-1}, \ldots, v_1$.
  - Regardless, the number of distinct paths is still $\Theta(n!)$. 
- $\rightarrow$ exhaustive search requires $\Omega(n!)$ in the worst case
Complexity of HP

- There are ‘faster’ algorithms, e.g., $O(n^22^n)$ deterministic and $O(1.415^n)$ randomized algorithms.
Complexity of HP

- There are ‘faster’ algorithms, e.g., $O(n^2 2^n)$ deterministic and $O(1.415^n)$ randomized algorithms.
- Is there a polynomial algorithm for Hamiltonian Path:
  - We don’t know, but no such algorithm is discovered yet, and it is unlikely that we can find one!
  - This relates to $P \neq NP$ conjecture that we see in a minute.
Complexity of HP

- There are ‘faster’ algorithms, e.g., $O(n^22^n)$ deterministic and $O(1.415^n)$ randomized algorithms.

- Is there a polynomial algorithm for Hamiltonian Path:
  - We don’t know, but no such algorithm is discovered yet, and it is unlikely that we can find one!
  - This relates to $P \neq NP$ conjecture that we see in a minute.

- There are many ‘Hard’ problems like Hamiltonian path problem for which we do not know whether a polynomial algorithm exists; they form a complexity class.
Complexity of HP

- There are ‘faster’ algorithms, e.g., $O(n^2 2^n)$ deterministic and $O(1.415^n)$ randomized algorithms.

- Is there a polynomial algorithm for Hamiltonian Path:
  - We don’t know, but no such algorithm is discovered yet, and it is unlikely that we can find one!
  - This relates to $P \neq NP$ conjecture that we see in a minute.

- There are many ‘Hard’ problems like Hamiltonian path problem for which we do not know whether a polynomial algorithm exists; they form a complexity class.
  - If there is a polynomial algorithm for any of these problems, there will be polynomial algorithms for all of them.
  - When you fail to come up with a polynomial algorithm for a problem, investigate whether it is ‘Hard’.
Application of Reductions

Assume you have a problem $P$ for which you look for an efficient, polynomial algorithm, and you fail after trying a bit.
Application of Reductions

- Assume you have a problem $P$ for which you look for an efficient, polynomial algorithm, and you fail after trying a bit.
- How can you determine whether you should keep searching for an efficient algorithm or whether it’s unlikely that any efficient algorithm for problem $P$ exists?
Application of Reductions

Assume you have a problem $P$ for which you look for an efficient, polynomial algorithm, and you fail after trying a bit.

How can you determine whether you should keep searching for an efficient algorithm or whether it’s unlikely that any efficient algorithm for problem $P$ exists?

If you can reduce one of those Hard problems to $P$ in polynomial time, then there is a polynomial algorithm for $P$ if and only if there is a polynomial algorithm for all those hard problems.
Application of Reductions

Since none of those Hard problems have any known polynomial algorithm, it is unlikely that you can come up with a polynomial algorithm for $P$.

- Informally, to give up searching for a polynomial algorithm for $P$, it suffices to reduce a ‘Hard’ problem to $P$ in polynomial time.
- We say the problem is **NP-Hard** in that case!
- To show $P$ is NP-Hard, we reduce another NP-Hard problem to $P$.
A complexity class is a set of problems that can be solved with a similar amount of time/space/cost resources.

- E.g., 3Sum-hard problems are those which require at least the same amount of time than 3Sum problem.
Complexity Classes

- A complexity class is a set of problems that can be solved with a similar amount of time/space/cost resources.
  - E.g., 3Sum-hard problems are those which require at least the same amount of time than 3Sum problem.

- Important complexity classes: \( P \), \( NP \), \( EXP \), \( R \), etc.

\[ P = \text{problems that can be solved in polynomial time}, \quad O(n^c) \text{ for some fixed } c \]

- E.g., given a graph on \( n \) vertices and \( m \) edges, find its MST; it can be done in \( m \alpha(m, n) \in O(n^{2 \alpha(n^2, n)}) \in O(n^3) \).

- Basically, all problems for which you have seen an algorithm belong to class \( P \) of problems.
A complexity class is a set of problems that can be solved with a similar amount of time/space/cost resources.

- E.g., 3Sum-hard problems are those which require at least the same amount of time than 3Sum problem.

Important complexity classes: P, NP, EXP, R, etc.

- P = problems that can be solved in polynomial time, i.e., $O(n^c)$ for some fixed $c$
  - E.g., given a graph on $n$ vertices and $m$ edges, find its MST; it can be done in $m\alpha(m, n) \in O(n^2\alpha(n^2, n)) \in O(n^3)$.
  - Basically, all problems for which you have seen an algorithm belong to class $P$ of problems.
A problem belongs to class \textit{NP} if a non-deterministic Turning machine can solve it in polynomial time.
A problem belongs to class \textit{NP} if a non-deterministic Turning machine can solve it in polynomial time.

These are problems whose solutions can be verified in polynomial time.

For decision problems, instances with a \textit{yes} answer can be verified.

E.g., Hamiltonian Path is an NP problem: given an instance of the problem we can verify if a solution gives a ‘yes’ answer in polynomial time.

Given a solution path, we can verify whether it is a Hamiltonian path, i.e., check whether it visits every vertex exactly once, in polynomial time (in $O(n \log n)$ exactly).
Class NP

- Is Hamiltonian Path in P?
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!

- Is Hamiltonian Path in NP?
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!

- Is Hamiltonian Path in NP?
  - Yes, we just showed given a solution (a candidate path), we can check in polynomial time whether it is Hamiltonian.
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!

- Is Hamiltonian Path in NP?
  - Yes, we just showed given a solution (a candidate path), we can check in polynomial time whether it is Hamiltonian.

- Is 3SUM in $P$?
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!
- Is Hamiltonian Path in NP?
  - Yes, we just showed given a solution (a candidate path), we can check in polynomial time whether it is Hamiltonian.
- Is 3SUM in $P$?
  - Yes, because it can be solved in $O(n^2)$.
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!

- Is Hamiltonian Path in NP?
  - Yes, we just showed given a solution (a candidate path), we can check in polynomial time whether it is Hamiltonian.

- Is 3SUM in P?
  - Yes, because it can be solved in $O(n^2)$.

- Is 3SUM in NP?
Class NP

- Is Hamiltonian Path in P?
  - We don’t know but it is unlikely!

- Is Hamiltonian Path in NP?
  - Yes, we just showed given a solution (a candidate path), we can check in polynomial time whether it is Hamiltonian.

- Is 3SUM in \( P \)?
  - Yes, because it can be solved in \( O(n^2) \).

- Is 3SUM in \( NP \)?
  - Yes, given a solution (3 numbers from the set), we can verify in polynomial time whether they sum to 0.
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
- Does the other direction hold?
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
- Does the other direction hold?
  - If a solution to a problem can be checked in polynomial time (e.g., Hamiltonian path), is it true that a polynomial-time algorithm exists for the problem?

We do not know the answer.

Question: Does any problem in NP belong to P? Is it that P=NP?

It is one of seven Millennium Prize problems in mathematics announced in 2000 by Clay Mathematics Institute with a prize of $1M for solving any of the problems. To date only one has been solved: the Poincare Conjecture, solved by Perelman in 2006; he declined the money.
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
- Does the other direction hold?
  - If a solution to a problem can be checked in polynomial time (e.g., Hamiltonian path), is it true that a polynomial-time algorithm exists for the problem?
  - We do not know the answer.
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
- Does the other direction hold?
  - If a solution to a problem can be checked in polynomial time (e.g., Hamiltonian path), is it true that a polynomial-time algorithm exists for the problem?
  - We do not know the answer.
- Question: Does any problem in $NP$ belong to $P$?
  - Is it that $P=NP$?
P vs NP

- If a problem can be solved in polynomial time (belongs to $P$), a solution to that can be checked in polynomial time, i.e., it belongs to $NP$.
  - Every problem in $P$ also belongs to $NP$.
- Does the other direction hold?
  - If a solution to a problem can be checked in polynomial time (e.g., Hamiltonian path), is it true that a polynomial-time algorithm exists for the problem?
  - We do not know the answer.
- Question: Does any problem in $NP$ belong to $P$?
  - Is it that $P=NP$?
  - It is One of seven Millennium Prize problems in mathematics announced in 2000 by Clay Mathematics Institute with a prize of $1M for solving any of the problems. To date only one has been solved: the Poincare Conjecture, solved by Perelman in 2006; he declined the money.