Week 7

Display replies in nested form

Settings

The cut-off date for posting to this forum is reached so you can no longer post to it.

Week 7

by Romana Riyaz (Instructor) - Thursday, 20 June 2024, 10:32 AM

Describe in your own words what a NP complete problem is and why a NP complete program is considered to be 'hard.'

Include one or two examples to explain your thought process to show what is occurring and how the methodology works. Use APA citations and references for any sources used.

51 words

Permalink

Re: Week 7

by Cherkaoui Yassine - Thursday, 1 August 2024, 3:44 PM

NP-complete problems are a class of problems in computational complexity theory that are both in NP (nondeterministic polynomial time) and NP-hard. A problem is in NP if its solution can be verified in polynomial time by a deterministic Turing machine. A problem is NP-hard if every problem in NP can be reduced to it in polynomial time. Thus, NP-complete problems are those for which a solution, if found, can be verified quickly, but finding the solution itself is as hard as solving any problem in NP.

Why NP-Complete Problems Are Considered Difficult: These problems are considered particularly difficult because no efficient algorithm is known for solving all NP-complete problems, and it's widely believed that no such algorithm exists. The difficulty arises from the fact that solving an NP-complete problem requires an exhaustive search through a vast number of possible solutions. As the input size increases, the time required to solve these problems grows exponentially, making them practically unsolvable for large inputs. The crux of the issue lies in the question of whether P equals NP, meaning whether every problem whose solution can be quickly verified can also be quickly solved. This question remains one of the biggest open problems in computer science.

Examples:

1. Subset Sum Problem: This problem involves determining whether there exists a subset of a given set of integers that sums to a specific target value. The problem is NP-complete because, while verifying a given subset's sum is straightforward, finding the subset involves checking all possible combinations of the integers.

2. Graph Coloring Problem: This involves assigning colors to the vertices of a graph in such a way that no two adjacent vertices share the same color, using the fewest number of colors. Determining whether a valid coloring exists with a given number of colors is NP-complete. Verifying a valid coloring can be done in polynomial time, but finding one may require examining a combinatorial explosion of possibilities as the graph grows.

Conclusion: The classification of NP-complete problems highlights their significance in understanding computational limits. While these problems can be verified quickly if a solution is presented, finding the solution itself is computationally intensive. This characteristic makes them a focal point in studies related to algorithm efficiency and computational theory.

References: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

392 words

Permalink Show parent

Re: Week 7

by Romana Riyaz (Instructor) - Saturday, 3 August 2024, 3:54 AM

Hello Yassine,
Thank You for your reply.Your explanation of NP-complete problems is clear and concise, effectively conveying why these problems are considered difficult in computational complexity theory. You define NP and NP-hard accurately, emphasizing the challenges in solving NP-complete problems due to the necessity of exhaustive searches and the exponential growth of possible solutions. The examples of the Subset Sum Problem and the Graph Coloring Problem aptly illustrate the complexities involved. Your conclusion highlights the significance of NP-complete problems in understanding computational limits and the ongoing challenge in algorithm efficiency, centering on the unresolved P versus NP question. Overall, your analysis is thorough and informative.
Regards,
Romana
107 words

Permalink Show parent

Re: Week 7
by Moustafa Hazeen - Sunday, 4 August 2024, 10:50 PM

Hey Cherkaoui, Great job on the discussion post your answer is clear and well written.
15 words

Permalink Show parent

Re: Week 7
by Akomolafe Ifedayo - Monday, 5 August 2024, 9:46 PM

Hi Yassine, you described the NP-complete problem, why it is always considered difficult, and also gave two examples with detailed explanations. Keep it up.
24 words

Permalink Show parent

Re: Week 7
by Manahil Farrukh Siddiqui - Tuesday, 6 August 2024, 3:36 AM

Hi Yassine,

Your answer is very well structured. Your explanation of NP-complete problems is super clear and helps me understand why these problems are so challenging in computational complexity theory. You nailed the NP and NP-hard definitions, and I understand why solving NP-complete problems is hard. It requires exhaustive searches, and the number of possible solutions grows exponentially.

The examples you gave, like the Subset Sum Problem and the Graph Coloring Problem, really show how complex these issues can get. Your conclusion about the importance of NP-complete problems in understanding the limits of computation and the challenge of improving algorithm efficiency is spot on. The whole P versus NP question is vast, and figuring that out would have massive implications for fields like cryptography and operations research. Overall, your answer is thorough and informative, making the topic more accessible.
139 words

Permalink Show parent

Re: Week 7
by Benjamin Chang - Tuesday, 6 August 2024, 10:25 PM

Hi Cherkaoui,
I appreciate your contribution to the initial post of the week. Your post effectively emphasizes the NP-complete problems and provides two straightforward examples that illustrate the difficulty of NP-complete problems. Well done!

Yours sincerely
Benjamin
37 words

Permalink Show parent

Re: Week 7
by Fadi Al Rifai - Wednesday, 7 August 2024, 12:56 PM

Hi Yassine,
Thank you for your thoughtful contribution to a detailed explanation of the NP-complete problem, I like your description of why an NP-complete program is considered to be 'hard.'
Keep it up.
33 words

Permalink Show parent

Re: Week 7
by Naqaa Alawadhi - Wednesday, 7 August 2024, 8:06 PM

Good job
2 words

Permalink Show parent

Re: Week 7
by Tousif Shahriar - Wednesday, 7 August 2024, 10:27 PM

Hello Yassine,
Your breakdown of why these problems are difficult to solve is particularly clear, emphasizing the exponential growth of possible solutions and the necessity of exhaustive searches. The examples you provided, the Subset Sum Problem and the Graph Coloring Problem, are excellent illustrations of the complexities involved. Your explanation effectively highlights the ongoing challenge in algorithm efficiency and the significance of the P versus NP question. Overall, your analysis is comprehensive and very informative.
Thanks for sharing!
78 words

Permalink Show parent

Re: Week 7
by Nour Jamaluddin - Thursday, 8 August 2024, 2:57 AM

Good job,
You understand problem clearly. Your post explains the most important information about the NP complete problem. It was helpful for me reading your post.
Thank you,
28 words

Permalink Show parent

Re: Week 7
by Siraajuddeen Adeitan Abdulfattah - Thursday, 8 August 2024, 5:14 AM

Hi Cherkaoui,

You provided a well articulated response to the questions asked. Your explanation on Np-complete problem is simple and quite clear to understand, you did a really good job with the description and the given examples , stating why each is regarded an NP-complete problem.
46 words

Permalink Show parent

Re: Week 7

by Jerome Bennett - Thursday, 8 August 2024, 8:40 AM

Greetings Cherkaoui,

I appreciate your post. You didn't just define NP-complete problems, you gave examples that were concise and clear, which aided my understanding.

24 words

Permalink Show parent

Re: Week 7

by Mejbaul Mubin - Saturday, 3 August 2024, 11:25 PM

What is an NP-Complete Problem?

In computational complexity theory, an **NP-complete problem** is a type of problem that is both in **NP** (nondeterministic polynomial time) and **NP-hard**. To understand this, we need to break down these terms:

NP (Nondeterministic Polynomial Time): This class contains decision problems for which a solution can be verified quickly (in polynomial time) if we are given the right information or "witness." However, finding that solution from scratch may not be efficient.

NP-hard: A problem is considered NP-hard if solving it in polynomial time would allow us to solve all problems in NP in polynomial time. In other words, an NP-hard problem is at least as hard as the hardest problems in NP.

NP-complete: A problem is NP-complete if it is in NP and as hard as any problem in NP. This means if you can find a polynomial-time solution for one NP-complete problem, you could solve all NP problems in polynomial time. However, no polynomial-time solutions for NP-complete problems are known, which is why they are considered difficult or "hard" (Cormen et al., 2009).

Why are NP-Complete Problems Considered Hard?

NP-complete problems are considered hard because no efficient algorithm is known to solve them in polynomial time. They require potentially exponential time to solve as the problem size grows, making them impractical for large instances. The difficulty arises from the combinatorial nature of these problems, where the number of possible solutions increases exponentially with input size.

If a polynomial-time algorithm were discovered for an NP-complete problem, it would imply that P = NP, meaning every problem whose solution can be verified quickly can also be solved quickly. However, this is one of the biggest unsolved questions in computer science, known as the **P vs. NP problem** (Garey & Johnson, 1979).

Examples of NP-Complete Problems

1. The Traveling Salesman Problem (TSP):

Problem Statement: Given a list of cities and the distances between them, find the shortest possible route that visits each city exactly once and returns to the origin city.

Explanation:

Verification: Given a specific tour (sequence of cities), checking if the total distance is less than a certain value can be done quickly in polynomial time.

Solution Finding: However, finding the shortest tour requires checking potentially all permutations of city orders, leading to a factorial number of possibilities as the number of cities increases (Held & Karp, 1962).

2. The Knapsack Problem:

Problem Statement: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight does not exceed a given limit and the total value is as large as possible.

Explanation:

Verification: For a given selection of items, it is easy to calculate the total weight and value to verify the solution.

Solution Finding: Finding the optimal combination of items to maximize value without exceeding weight involves considering all possible subsets of items, which is exponential in complexity (Kellerer et al., 2004).

Methodology and Implications

The methodology for tackling NP-complete problems often involves heuristic or approximation algorithms that provide near-optimal solutions in a reasonable amount of time. These approaches do not guarantee the optimal solution but can be practical for real-world applications where exact solutions are computationally infeasible.

Approximation Algorithms: For problems like TSP, approximation algorithms such as the nearest neighbor or minimum spanning tree-based methods provide solutions close to the optimum.

Heuristic Methods: Techniques such as genetic algorithms or simulated annealing are used to find good enough solutions without exhaustive search.

Conclusion

NP-complete problems are fundamental in understanding computational complexity and optimization. While these problems are challenging due to their exponential nature, advances in approximation algorithms and heuristics continue to provide practical ways to address them. The ongoing research in this field aims to uncover deeper insights into these problems, with the ultimate question of P vs. NP remaining a cornerstone of theoretical computer science.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* W.H. Freeman.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics, 10*(1), 196-210.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack problems.* Springer.

725 words

Permalink Show parent

Re: Week 7

by Romana Riyaz (Instructor) - Sunday, 4 August 2024, 10:08 AM

Hello Mubin,
Your explanation of NP-complete problems is clear and thorough, effectively breaking down complex concepts into understandable parts. You succinctly describe the definitions of NP, NP-hard, and NP-complete, and why NP-complete problems are challenging due to their combinatorial nature. The inclusion of specific examples like the Traveling Salesman Problem and the Knapsack Problem, along with their explanations, illustrates the theoretical and practical implications of these problems well. The discussion on heuristic and approximation algorithms adds practical insight into how these problems are approached in real-world scenarios. Overall, your explanation provides a solid foundation for understanding NP-complete problems and their significance in computational complexity theory.
Regards,
Romana

107 words

Permalink Show parent

Re: Week 7
by Moustafa Hazeen - Sunday, 4 August 2024, 10:50 PM

Hey Mejbaul, Great job on the discussion post your answer is clear and well written.
15 words

Permalink Show parent

Re: Week 7
by Cherkaoui Yassine - Monday, 5 August 2024, 4:14 PM

Mubin, I wanted to drop you a quick note to say excellent job on your discussion post! Your answer is not only clear but also very well-written. It's evident that you put thought and effort into it, and it really shines through. Keep up the great work!
47 words

Permalink Show parent

Re: Week 7
by Akomolafe Ifedayo - Monday, 5 August 2024, 9:48 PM

Hi Mubin, overall your post was engaging to go through. You defined the NP, NP-hard, and NP-complete. You also discussed why the NP-complete problems are considered hard. Lastly, you discussed in detail the Knapsack problem being an example of the NP-complete problems. Keep it up.
45 words

Permalink Show parent

Re: Week 7
by Benjamin Chang - Tuesday, 6 August 2024, 10:27 PM

Hi Mejbaul
I am in love of your post because it not only provides a definition of an NP-complete problem but also provides a comprehensive catalog of examples that address all of the topics we have discussed. I appreciate the way you emphasize the keywords that aid in comprehension. By the way, you did a great job. well done!
Yours sincerely
Benjamin
62 words

Permalink Show parent

Re: Week 7
by Fadi Al Rifai - Wednesday, 7 August 2024, 12:57 PM

Hi Mubin,
Good work, Thanks for sharing your explanation about the NP-complete problem, I like your description of why an NP-complete program is considered to be 'hard.'
Keep it up.
30 words

Permalink Show parent

Re: Week 7
by Naqaa Alawadhi - Wednesday, 7 August 2024, 8:06 PM

Good job
2 words

Permalink Show parent

Re: Week 7
by Nour Jamaluddin - Thursday, 8 August 2024, 2:59 AM

Perfect job.
Your post is very interesting. You explained the full description of the problem in details. I liked your way and your post organization very much.
Good luck!
29 words

Permalink Show parent

Re: Week 7
by Siraajuddeen Adeitan Abdulfattah - Thursday, 8 August 2024, 5:19 AM

Hi Mubin,

Very well structured and articulated response to the question asked on NP-complete problem. I like your approach to explaining and breaking down the problem in details and how it relates to other problem. You did an excellent job with your explanation on NP-complete problem and why they are regard as hard, you also gave some well explained examples to proof your point which I think are quite informative and helps to understand your point better.
77 words

Permalink Show parent

Re: Week 7
by Jerome Bennett - Thursday, 8 August 2024, 8:05 AM

Greetings Mejbaul,

You've done a great job explaining NP-complete problems in a way that's easy to understand. You broke down some tricky concepts like NP, NP-hard, and NP-complete, and explained why these problems are such a headache - mainly because there are so many possible solutions to check.
To make it all more concrete, you brought in some classic examples like the Traveling Salesman Problem (which I also wrote about) and the Knapsack Problem.
75 words

Permalink Show parent

Re: Week 7
by Chong-Wei Chiu - Thursday, 8 August 2024, 10:52 AM

Hello, Mejbaul Mubin. Thank you for sharing your opinion about NP-complete problems. In your post, you illustrate the basic concepts of NP, NP-hard, and NP-complete problems. Furthermore, you also list two examples to illustrate why NP-complete problems are hard to solve. That is amazing.
44 words

Permalink Show parent

Re: Week 7
by Benjamin Chang - Sunday, 4 August 2024, 8:46 AM

Hello, everyone!

This week, we learned about NP-complete theory, which is a concept in computer science that measures the difficulty of solving problems using polynomial-time algorithms. According to GeeksforGeeks (2024), NP-completeness applies to decision problems because it's easier to compare the difficulty of decision problems than that of optimization problems. Simply put, if

someone shows you how to solve a problem, you can follow the instructions without needing to understand how it works, and you will get the solution. Moreover, if someone provides a method to solve the problem, you can also verify whether the solution is correct using that method.

Now that we understand the basic concept of NP-complete theory, let's explore how this term works with Big-O complexity for algorithms. NP (non-deterministic polynomial) problems take longer to solve, even though checking the answer has polynomial run-time. For example, we have learned about the traveling salesman problem and the knapsack problem, both classic NP-complete algorithms. The challenging part of these problems is that they require an impractical amount of time to find a solution.

Another example to consider is the Sudoku game. The more numbers we confirm in the puzzle, the quicker we can solve it. This illustrates polynomial run-time similar to NP-complete problems. It is easy to verify a solution but takes time to solve for all possible solutions.

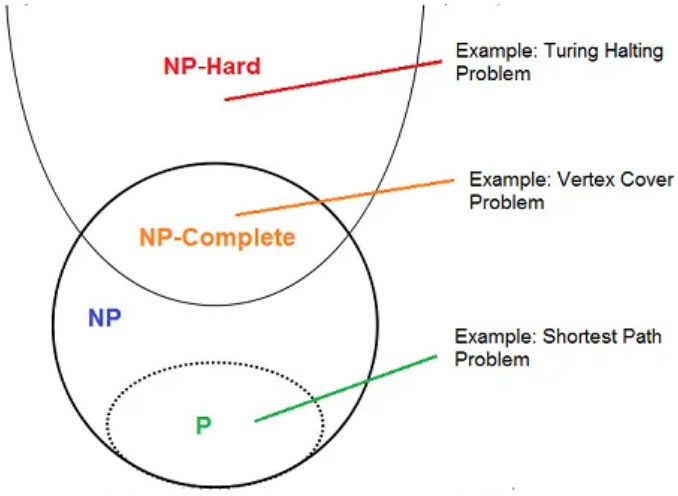


Image from Vaghela (2021) P and NP

In essence, NP-complete problems are considered hard because they involve a vast number of potential solutions that must be checked one by one, making it time-consuming to find the correct answer. However, verifying the solution, once found, is straightforward and quick. Understanding and tackling NP-complete problems is crucial in computer science, as it helps us push the boundaries of what can be efficiently solved.

That is why we say an NP-complete problem is considered to be hard. It's easy to verify a solution once someone provides it to you, but finding that solution is difficult due to its exponential time complexity. As more problems come together to form larger ones, the complexity and time required to solve them increase significantly. This time-consuming nature makes NP-complete problems particularly challenging.

Reference

GeeksforGeeks. (2024, May 15). *Introduction to NP-Complete Complexity Classes*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-np-completeness/

-, C. A. (2010). *A Practical Introduction to data Structures and algorithm Analysis third edition (C++ Version)* (3rd ed.). https://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

Vaghela, V. (2021, December 16). Introduction to NP Completeness - Vipasha Vaghela - Medium. *Medium*. https://vips3201v.medium.com/introduction-to-np-completeness-d3dfa771d994

404 words

Re: Week 7
by Romana Riyaz (Instructor) - Sunday, 4 August 2024, 10:09 AM

Hello Chang,
Your explanation of NP-complete theory is engaging and informative, effectively conveying the key concepts of polynomial-time algorithms and the practical challenges associated with NP-complete problems. The use of examples like the Traveling Salesman Problem, the Knapsack Problem, and Sudoku helps illustrate the complexity and the nature of NP-complete problems clearly. The distinction between the difficulty of finding a solution and the ease of verifying it is well-articulated, reinforcing the core idea behind NP-completeness. Additionally, the inclusion of references and visual aids, such as the image from Vaghela, enhances the comprehension of the topic. Overall, your summary provides a solid overview of NP-complete problems and their significance in computer science.
Regards,
Romana
113 words

Permalink Show parent

Re: Week 7
by Moustafa Hazeen - Sunday, 4 August 2024, 10:51 PM

Hey Benjamin, Great job on the discussion post your answer is clear and well written.
15 words

Permalink Show parent

Re: Week 7
by Cherkaoui Yassine - Monday, 5 August 2024, 4:15 PM

Hey, just wanted to give you props for your discussion post—it's fantastic! Your response is clear, articulate, and well-structured. It's evident you know your stuff and put in the effort to communicate effectively. Keep up the great work!
39 words

Permalink Show parent

Re: Week 7
by Akomolafe Ifedayo - Monday, 5 August 2024, 9:52 PM

Hi Chang, your work was engaging to go through. I liked the visual representation you also included. You defined the NP-complete problem and also discussed why it is always considered hard. As part of this problem's example, I never knew the Sudoku game was an example. It was interesting to learn from your post. I never liked the game because I found it difficult, this explains why I found it that way all along (The NP-complete problem haha). Keep it up.
81 words

Permalink Show parent

Re: Week 7
by Anthony Jones - Wednesday, 7 August 2024, 3:29 AM

Hello,

Good post! I liked your examples and diagram illustrating how different problem categorizations relate. Where do EXP problems fall in that graph (EXP problems are ones that are exponentially difficult to even check plausible solutions)?

God bless!
Anthony
39 words

Permalink Show parent

Re: Week 7
by Naqaa Alawadhi - Wednesday, 7 August 2024, 8:07 PM

Good job
2 words

Permalink Show parent

Re: Week 7
by Nour Jamaluddin - Thursday, 8 August 2024, 3:03 AM

Excellent work
Thanks for sharing your full description of the NP complete problem. You were able to highlight the importance of understanding this problem even if there is no clear solution. Also, your references are perfect.
Keep it up
39 words

Permalink Show parent

Re: Week 7
by Siraajuddeen Adeitan Abdulfattah - Thursday, 8 August 2024, 5:27 AM

Hi Chang,

I thought your explanation was excellent and easy to understand. Thank you for that simple approach you took explaining what NP-complete theory is all about. I am able to relate well with your explanation and given examples because you made look so simple and easy to digest. An important factor that makes NP-complete problem hard is the large amount of time required to arrive at the the best solution because there is need to check all possible solutions there is.
82 words

Permalink Show parent

Re: Week 7
by Moustafa Hazeen - Sunday, 4 August 2024, 7:25 PM

An NP-complete problem is a type of problem in computational complexity theory that resides within the class NP (nondeterministic polynomial time). These problems share two critical characteristics:
1. Verification in Polynomial Time: Given a proposed solution, it can be verified whether it is correct or not in polynomial time.
2. NP-Hardness: Every problem in NP can be transformed into this problem in polynomial time. This implies that an efficient algorithm for solving an NP-complete problem would provide efficient solutions for all problems in NP.
The hardness of NP-complete problems stems from the fact that, despite the solutions being verifiable in polynomial time, no polynomial-time algorithm is known to solve these problems. This means that as the size of the input grows, the time required to solve the problem grows exponentially in the worst case, making them impractical to solve for large instances.
Example 1: The Traveling Salesman Problem (TSP) The Traveling Salesman Problem asks whether a salesman can visit a set of cities exactly once and return to the origin city while minimizing the total travel distance. Given a proposed route, it is easy to verify if the route is valid and to calculate the total distance, which can be done in polynomial time. However, finding the optimal route requires checking all possible permutations of the cities, which grows factorially with the number of cities, making it computationally infeasible for large numbers of cities.
Example 2: The Subset Sum Problem The Subset Sum Problem asks whether there exists a subset of a given set of integers that sums up to a specific target value. Given a subset, it is straightforward to verify if the sum matches the target, a task that can be completed in polynomial time. However, finding such a subset involves evaluating potentially all possible subsets, which exponentially increases with the size of the input set.
These examples illustrate that while solutions can be verified quickly, finding those solutions is computationally challenging due to the exponential growth of possible solution spaces as the problem size increases.
References:
• Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company.

• Karp, R. M. (1972). Reducibility among combinatorial problems. In Complexity of Computer Computations (pp. 85-103). Springer.

379 words

Permalink Show parent

Re: Week 7

by Mejbaul Mubin - Monday, 5 August 2024, 2:00 PM

Hi Moustafa,

Your post provides a clear and concise explanation of NP-complete problems, highlighting their key characteristics and the inherent difficulty in solving them. The examples of the Traveling Salesman Problem and the Subset Sum Problem effectively illustrate these concepts. Your references to foundational works by Garey & Johnson and Karp add credibility and depth to your discussion. Well done!

60 words

Permalink Show parent

Re: Week 7

by Benjamin Chang - Tuesday, 6 August 2024, 10:29 PM

Hi Moustafa
Your explanation of the NP-problem was straightforward and facilitated our comprehension. I particularly enjoyed the first of your two examples, which is the traveling salesman problem that we learned in this class to illustrate the NP-complete problem. Keep up the good work!
Yours sincerely
Benjamin

47 words

Permalink Show parent

Re: Week 7

by Tousif Shahriar - Wednesday, 7 August 2024, 10:29 PM

Hi Moustafa,
I appreciate your thorough explanation of NP-complete problems. You've clearly articulated why these problems are considered difficult to solve, emphasizing the verification in polynomial time and the concept of NP-hardness. The examples of the Traveling Salesman Problem and the Subset Sum Problem are particularly helpful in demonstrating the exponential growth of solution spaces. Your discussion highlights the practical challenges in computational complexity theory.
Thank you for providing such a comprehensive overview.

73 words

Permalink Show parent

Re: Week 7

by Siraajuddeen Adeitan Abdulfattah - Thursday, 8 August 2024, 5:33 AM

Hi Moustafa,

Good submission in response to the question asked. You gave a good explanation on NP-complete problem and why they are regard and hard. you stated essentially that the time required to solve the problem grows larger as the size of input grows, making it impractical to solve the problem. You gave good example to demonstrate the NP-complete problem and its impractical time required to solve the problem.

69 words

Permalink Show parent

Re: Week 7

by Fadi Al Rifai - Sunday, 4 August 2024, 10:33 PM

An NP-complete problem is a type of computational problem that is both challenging to solve and to verify. To understand why these problems are considered 'hard,' let's break it down:

What is NP-complete?

1. **NP (Nondeterministic Polynomial time):** This class contains decision problems for which a given solution can be verified quickly (in polynomial time). In other words, if you are given a potential solution to the problem, you can check whether it is correct in a time that scales polynomially with the size of the input.
2. **NP-complete:** These are the hardest problems in NP in the sense that if you can solve any NP-complete problem efficiently (in polynomial time), then you can solve all problems in NP efficiently. Conversely, if an efficient solution to an NP-complete problem can't be found, it suggests that solving any problem in NP efficiently might also be impossible. Thus, NP-complete problems are a subset of NP problems that are believed to be computationally intractable.

Why are NP-complete problems 'hard'?

The 'hardness' comes from two main aspects:

1. **Finding a Solution:** For NP-complete problems, no polynomial-time algorithms are known to solve them. This means that as the size of the input grows, the time it takes to find a solution can grow exponentially.
2. **Verification:** While a solution can be verified quickly, finding that solution in the first place is a daunting task. The difficulty lies in the exhaustive search or the combinatorial explosion of possible solutions.

Examples and Methodology

Example 1: Traveling Salesperson Problem (TSP)

- **Problem:** Given a list of cities and the distances between each pair, find the shortest possible route that visits each city exactly once and returns to the origin city.
- **Why it's hard:** There are factorial (n!) possible routes for n cities, so finding the shortest one involves checking all possible routes, which becomes impractical for large n. Though you can verify a given route quickly by adding up the distances and checking if it's the shortest, finding this route from scratch is computationally intense.

Example 2: Knapsack Problem

- **Problem:** Given a set of items, each with a weight and a value, determine the most valuable subset of items that fit within a given weight capacity.
- **Why it's hard:** The problem requires exploring all possible subsets of items to find the one that maximizes value without exceeding the weight capacity. For a large number of items, the number of subsets grows exponentially, making it difficult to solve efficiently.

Methodology and Approach

To handle NP-complete problems, several strategies are employed:

1. **Exact Algorithms:** These are often impractical for large instances but guarantee the optimal solution. Examples include exhaustive search or dynamic programming, which may work for smaller problems but become infeasible as size increases.
2. **Heuristic Methods:** These provide good enough solutions within a reasonable time, but they don't guarantee optimality. Examples include greedy algorithms or approximation algorithms.
3. **Approximation Algorithms:** These offer solutions close to optimal within a known factor. For example, a 2-approximation algorithm for the TSP guarantees that the solution is at most twice as long as the optimal route.
4. **Special Cases:** Some problems have special cases or restrictions where they become solvable in polynomial time. Identifying and exploiting these special cases can sometimes lead to practical solutions for those specific instances.

In my opinion, NP-complete problems are challenging because finding a solution quickly (in polynomial time) is currently beyond reach for general cases, though verifying a solution once found is relatively straightforward. Their 'hardness' reflects the computational complexity and the large number of potential solutions that must be considered.

Reference:

Chapter 17: Limits to Computation Section 17.1 in A Practical Introduction to Data Structures and Algorithm Analysis by Clifford A. Shaffer.

Read “lecture 21: NP-Hard Problems”, sections 21.1 through 21.3. Download the pdf.

Chapter 8 NP Complete Problems in Algorithms by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani available at http://www.cs.berkeley.edu/~vazirani/algorithms/chap8.pdf

Reductions and NP-Completeness lecture notes available at

Lecture notes presentation on NP-Complete problems https://courses.engr.illinois.edu/cs473/sp2010/notes/21-nphard.pdf

652 words

Permalink Show parent

Re: Week 7

by Romana Riyaz (Instructor) - Monday, 5 August 2024, 12:53 AM

Hello Fadi,
Thank you for your submission. Your explanation of NP-complete problems effectively captures their inherent complexity and why they are considered challenging. You clearly define NP and NP-complete, illustrating the difficulty of finding solutions versus verifying them. The examples of the Traveling Salesperson Problem and the Knapsack Problem are well-chosen to demonstrate the exponential growth in potential solutions. Your discussion on methodologies, including exact algorithms, heuristics, and approximation algorithms, provides a comprehensive overview of current approaches to tackling these problems. Overall, your analysis is insightful and highlights the fundamental computational challenges posed by NP-complete problems.
Regards,
Romana
98 words

Permalink Show parent

Re: Week 7

by Cherkaoui Yassine - Monday, 5 August 2024, 4:15 PM

Fadi, I wanted to take a moment to acknowledge your discussion post—it's really well-done! Your answer is clear, concise, and well-articulated. It's evident you put thought and care into your response, and it's paying off. Keep up the excellent work!
41 words

Permalink Show parent

Re: Week 7

by Anthony Jones - Wednesday, 7 August 2024, 3:47 AM

Hello,

When we are trying to solve np-complete problems, how might we construct a heuristic? What are some examples of ones that are used for existing np-complete problems?

God bless!
Anthony
31 words

Permalink Show parent

Re: Week 7

by Nour Jamaluddin - Sunday, 4 August 2024, 11:43 PM

NP-complete problem is considered one of the most difficult and complex problems facing programmers, and it is essential to know about it. It is regarded as a branch of the NP problems, but it is the most difficult and complex problem. It represents a challenge for computers because of the complexity of the problem and the solution together.

According to Erickson (2009) "a problem is NP-complete if it is both NP-hard and an element of NP (or 'NP-easy'). problems are the hardest problems in NP. If anyone finds a polynomial-time algorithm for even one NP-complete problem, then that would imply a polynomial-time algorithm for every NP-complete problem" (P. 3).

For example, the Sudoku puzzle. Finding the correct solution can be challenging, especially for larger grids. However, once someone provides a solution, it's relatively easy to verify if it's correct by checking each row, column, and box.

NP-complete problems are considered "hard" because, as of now, there is no known efficient algorithm that can solve all instances of any NP-complete problem in polynomial time. The crux of the difficulty lies in the fact that if you can find a polynomial-time algorithm for one NP-complete problem, you can use it to solve all NP-complete problems efficiently. Despite decades of research, no one has found a way to solve NP-complete problems efficiently. This question, known as the P vs NP problem, is one of computer science's most important open problems.

References

Erickson. J. 2009. *NP-Hard Problems*.
https://my.uopeople.edu/pluginfile.php/1861950/mod_book/chapter/512270/Jeff_nphardRA7.pdf

244 words

Permalink Show parent

Re: Week 7

by Romana Riyaz (Instructor) - Monday, 5 August 2024, 10:22 AM

Nour,
Thank you for your submission. The explanation of NP-complete problems provides a clear and concise overview of their complexity and significance in computer science. You effectively describe NP-complete problems as both challenging and crucial for understanding computational limits, especially in the context of the P vs NP problem. The definition from Erickson (2009) accurately highlights the concept of NP-completeness, underscoring its implications for finding polynomial-time solutions. The Sudoku example illustrates the difficulty of these problems well, showing how verification is simpler than solving. It might be beneficial to briefly mention some other classic NP-complete problems, such as the Traveling Salesman Problem or Knapsack Problem, to give a broader sense of their impact.
Best,
Romana
115 words

Permalink Show parent

Re: Week 7
by Mejbaul Mubin - Monday, 5 August 2024, 2:03 PM

Hi Nour Jamaluddin,

Your post effectively highlights the significance and complexity of NP-complete problems, emphasizing their challenging nature for programmers. The example of the Sudoku puzzle is a great choice to illustrate the concept. Your reference to Erickson strengthens your explanation. Well done!

43 words

Permalink Show parent

Re: Week 7
by Mahmud Hossain Sushmoy - Tuesday, 6 August 2024, 1:51 AM

Hello Nour,
Your explanation of NP-complete problems effectively highlights their significance and complexity in the field of computer science. You succinctly describe the nature of these problems, emphasizing their challenging nature and the critical relationship between NP-complete and NP problems. The use of Erickson's definition strengthens your explanation, clearly stating that NP-complete problems are both NP-hard and part of NP. The Sudoku example is apt, illustrating how solutions can be verified quickly while finding them is computationally intensive. Thank you for your contribution to the discussion forum this week.

89 words

Permalink Show parent

Re: Week 7
by Manahil Farrukh Siddiqui - Tuesday, 6 August 2024, 3:43 AM

Hi Nour,

Your answer is very well structured. Your explanation of NP-complete problems is super clear and helps me understand why these problems are so challenging in computational complexity theory. You nailed the NP and NP-hard definitions, and I understand why solving NP-complete problems is hard. It requires exhaustive searches, and the number of possible solutions grows exponentially.

The examples you gave, like the Subset Sum Problem and the Graph Coloring Problem, really show how complex these issues can get. Your conclusion about the importance of NP-complete problems in understanding the limits of computation and the challenge of improving algorithm efficiency is spot on. The whole P versus NP question is vast, and figuring that out would have massive implications for fields like cryptography and operations research. Overall, your answer is thorough and informative, making the topic more accessible.

139 words

Permalink Show parent

Re: Week 7
by Mahmud Hossain Sushmoy - Monday, 5 August 2024, 3:44 AM

An NP problem, or Nondeterministic Polynomial time problem, is one where, given a solution, we can verify its correctness quickly, specifically in polynomial time. However, finding the solution itself might not be quick or straightforward (Shaffer, 2011). These problems are significant because, while the verification of a solution is efficient, the process of finding the solution is not necessarily so.

A problem is NP-hard if solving it quickly (in polynomial time) would enable us to solve all other problems in NP quickly as well. Essentially, NP-hard problems are at least as tough as the hardest problems in NP. This characteristic makes them crucial in understanding computational complexity, as they represent the upper bounds of problem difficulty within NP.

An NP-complete problem belongs to both NP and NP-hard categories. This means we can quickly verify a given solution, and if we could solve this problem quickly, we could solve all NP problems quickly. In other words, NP-complete problems are the most challenging problems within NP. They are solvable in polynomial time, but no one has discovered a polynomial-time algorithm for any NP-complete problem, and it is widely suspected that none exists. If we were to find such an algorithm, it would revolutionize the field of computer science by allowing us to solve all NP problems efficiently (Shaffer, 2011).

Consider the chess game problem as an example. In this problem, the goal is to determine whether there is a winning strategy for a player from a given position. Given a specific sequence of moves, verifying whether it leads to a win can be done quickly, especially using chess engines that evaluate positions. However, determining the best sequence of moves from any given position to ensure a win is computationally challenging and represents an NP-complete problem. This is because the number of possible moves in a chess game grows exponentially with each turn, making it extremely difficult to find the optimal sequence in polynomial time.

NP-complete problems are central to theoretical computer science because they embody the most difficult problems for which solutions can be verified quickly. The challenge lies in their inherent complexity, which makes finding efficient solutions extremely hard. Understanding these problems and their implications continues to be a fundamental area of research.

References

Shaffer, C. (2011). *A Practical Introduction to Data Structures and Algorithm Analysis*. Blacksburg: Virginia. Tech.

388 words

Permalink Show parent

Re: Week 7
by Mejbaul Mubin - Monday, 5 August 2024, 2:02 PM

Hi Mahmud,

Your post does an excellent job of explaining NP, NP-hard, and NP-complete problems, clarifying their significance in computational complexity. The chess game example is particularly effective in illustrating the practical implications of these concepts. Your reference to Shaffer adds a solid academic backing to your explanation. Great work!
50 words

Permalink Show parent

Re: Week 7
by Anthony Jones - Wednesday, 7 August 2024, 4:02 AM

Hello,

Good post! It actually turns out that the chess example is not NP. It's harder than NP. NP means Non-deterministic Polynomial, meaning that it is difficult to determine feasible solutions, but we can check those solutions in polynomial time. The chess example is really difficult to even check (exponential time). So it is worse than NP-complete. I believe it would be what is called EXP.

God bless!
Anthony
69 words

Permalink Show parent

Re: Week 7
by Tousif Shahriar - Wednesday, 7 August 2024, 10:31 PM

Hi Mahmud,
You've done an excellent job of describing the critical aspects, such as the efficiency of verifying solutions versus the

difficulty in finding them. The chess game problem example is particularly effective in demonstrating the exponential growth of possible moves and the resulting computational challenge. Your discussion underlines the importance of these problems in understanding computational complexity.
Thank you for providing such a clear and insightful overview.
68 words

Permalink Show parent

Re: Week 7
by Akomolafe Ifedayo - Monday, 5 August 2024, 9:39 PM

Describe in your own words what an NP-complete problem is and why an NP-complete program is considered to be 'hard.'

An NP-complete problem is both NP-hard and an element of NP. These problems can be verified quickly, and any NP problem can be transformed into any NP-complete problem in polynomial time (Shaffer, 2011).

NP-complete problems are the hardest problems in NP. They are considered 'hard' because no efficient algorithm is known to solve all NP-complete problems quickly in polynomial time. If someone discovers a polynomial-time algorithm for any NP-complete problem, it would solve all NP problems, however, despite extensive research, no such algorithm has been found, and it is widely believed that P ≠ NP, meaning no polynomial-time solution exists for NP-complete problems.

Examples:

1. **The Traveling Salesman Problem (TSP)**:

· **Problem**: Given a list of cities and distances between each pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point (GeeksforGeeks, 2023). This problem is a famous NP-hard problem and there is no polynomial-time known solution for this problem. This is because, for a small number of cities, the problem can be solved by checking all possible routes. However, as the number of cities increases, the number of possible routes grows factorially (n!), making it impractical to solve by brute force. Verifying a given route's length is quick (polynomial time), but finding the shortest route is computationally intense.

2. The Boolean Satisfiability Problem (SAT): It is also an NP-complete problem that determines if a Boolean formula is satisfiable or not (GeeksforGeeks, 2024).

· **Problem**: Given a boolean formula, determine if there exists an assignment of truth values to variables that make the formula true. It Checks quickly if a particular assignment satisfies the formula. However, finding such an assignment (or proving none exists) requires examining many possible assignments, and the number of these grows exponentially with the number of variables.

References

GeeksforGeeks. (2023). Travelling Salesman Problem (TSP) Implementation. https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/

GeeksforGeeks. (2024). 2-Satisfiability (2-SAT) Problem. https://www.geeksforgeeks.org/2-satisfiability-2-sat-problem/

Shaffer, C., A. Chapter 17: Limits to Computation Section 17.1 in A Practical Introduction to Data Structures and Algorithm Analysis. pdf

361 words

Permalink Show parent

Re: Week 7
by Mahmud Hossain Sushmoy - Tuesday, 6 August 2024, 1:49 AM

Hello Akomolafe,
Your explanation of NP-complete problems is comprehensive and well-structured, effectively conveying the complexity and significance of these problems within computational theory. The inclusion of definitions, criteria, and examples like the Traveling Salesman Problem (TSP) and the Boolean Satisfiability Problem (SAT) enhances understanding by illustrating both the theoretical aspects and practical implications of NP-complete problems. Thank you for your contribution to the discussion forum this week.
67 words

Permalink Show parent

Re: Week 7
by Naqaa Alawadhi - Tuesday, 6 August 2024, 1:38 AM

A NP-complete problem is a type of computational problem in complexity theory that belongs to the set of NP problems and is at least as hard as the hardest problems in NP. A problem is considered NP-complete if it is both in NP (nondeterministic polynomial time) and any problem in NP can be reduced to it by a polynomial-time algorithm.

NP-complete problems are considered hard because they are believed not to have polynomial-time algorithms for their solutions. This means that as the size of the input grows, the time required to solve these problems increases exponentially. The difficulty arises from the fact that verifying a potential solution to an NP-complete problem can be done quickly, but finding a solution itself may require an exhaustive search through all possible solutions.

An example of an NP-complete problem is the Traveling Salesman Problem (TSP), where given a list of cities and distances between them, the task is to find the shortest possible route that visits each city exactly once and returns to the origin city. Another example is the Boolean Satisfiability Problem (SAT), where given a Boolean formula, determining if there exists an assignment of truth values to variables that satisfies the formula.

These problems are considered hard due to their combinatorial nature and lack of known efficient algorithms for solving them optimally in all cases.

Reference:
Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman & Co Ltd.
246 words

Permalink Show parent

Re: Week 7
by Mahmud Hossain Sushmoy - Tuesday, 6 August 2024, 1:47 AM

Hello Naqaa,
Your explanation of NP-complete problems is clear and concise, effectively covering the fundamental aspects of these computational challenges. You accurately describe the criteria for a problem to be classified as NP-complete, emphasizing the difficulty in finding solutions despite the ease of verification. Thank you for your contribution to the discussion forum this week.
55 words

Permalink Show parent

Re: Week 7

by Manahil Farrukh Siddiqui - Tuesday, 6 August 2024, 3:42 AM

Hi Naqaa,

Your answer is very well structured. Your explanation of NP-complete problems is super clear and helps me understand why these problems are so challenging in computational complexity theory. You nailed the NP and NP-hard definitions, and I understand why solving NP-complete problems is hard. It requires exhaustive searches, and the number of possible solutions grows exponentially.

The examples you gave, like the Subset Sum Problem and the Graph Coloring Problem, really show how complex these issues can get. Your conclusion about the importance of NP-complete problems in understanding the limits of computation and the challenge of improving algorithm efficiency is spot on. The whole P versus NP question is vast, and figuring that out would have massive implications for fields like cryptography and operations research. Overall, your answer is thorough and informative, making the topic more accessible.

139 words

Permalink Show parent

Re: Week 7

by Manahil Farrukh Siddiqui - Tuesday, 6 August 2024, 2:56 AM

There are two types of NP-complete computing problems: NP-hard and NP (nondeterministic polynomial time) (GeeksforGeeks, 2013a). We can swiftly confirm a possible answer because we are in NP. Any other NP problem that can be effectively transformed into it is said to be NP-hard. It is possible to solve all NP issues rapidly if we can solve an NP-hard problem quickly (Lark, 2024).

NP-complete issues are complex because there needs to be an easy technique to solve them. A central open question in computer science concerning P and NP could be resolved if we could swiftly solve an NP-complete problem, enabling us to solve any NP problem (Haroon, 2023). These problems are complicated because, although it is quick to look for a solution, it frequently takes a long time to locate one, mainly as the problem grows larger (Haroon, 2023).

An NP-complete problem is the Traveling Salesman Problem (TSP). It entails figuring out the quickest path that makes one stop at each city before returning to the beginning. It is fast to calculate the distance and determine whether a route is feasible (GeeksforGeeks, 2013b). TSPs are complex NP-hard issues, as demonstrated by the fact that many NP problems can be converted into TSPs. Resolving a TSP can facilitate the solution of the Hamiltonian Cycle problem.

However, another example is the Knapsack dilemma. It entails determining how to fill a backpack with items while underweight restrictions (Haroon, 2023). Because solving it aids in solving other NP problems, such as employment scheduling, this problem is NP-hard.

Two things must be demonstrated for a problem to be NP-complete. First and foremost, it must be in the NP to verify any solution rapidly. Second, we must demonstrate that it can effectively transform from an existing NP-complete issue. To demonstrate that the Hamiltonian Cycle issue is NP-complete, for instance, we need to demonstrate that it is in NP and can be logically transformed from the 3-SAT problem.

In a nutshell, comprehension of NP-complete issues is essential to comprehending computational complexity. The main P vs. NP debate in computer science would be resolved by solving these issues rapidly, which makes them troublesome. Because of their complexity, NP-complete problems aid in our understanding of the boundaries and powers of computers.

References

GeeksforGeeks. (2013, August 22). Introduction to NP-Completeness. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-np-completeness/

Haroon, R. (2023, September 13). Decoding Complexity: A Deep Dive into NP-complete Problems. Nerd for Tech. https://medium.com/nerd-for-tech/decoding-complexity-a-deep-dive-into-np-complete-problems-fe72d8efc677

Lark. (2024). Np Hard Definition of Np Hardness. Larksuite.com. https://www.larksuite.com/en_us/topics/ai-glossary/np-hard-definition-of-np-hardness

410 words

Permalink Show parent

Re: Week 7

by Loubna Hussien - Wednesday, 7 August 2024, 12:46 AM

You did great work, keep it up.

7 words

Permalink Show parent

Re: Week 7

by Fadi Al Rifai - Wednesday, 7 August 2024, 12:58 PM

Hi Manahil,
Your post was well-detailed about the NP-complete problem, I like your description of why an NP-complete program is considered to be 'hard.'
Keep it up.

27 words

Permalink Show parent

Re: Week 7

by Tousif Shahriar - Wednesday, 7 August 2024, 10:32 PM

Hello Manahil,
Your distinction between NP-hard and NP problems is well-articulated, and your discussion on the significance of solving NP-hard problems to potentially solve all NP problems is insightful. The examples of the Traveling Salesman Problem and the Knapsack problem effectively illustrate the complexities involved. Your explanation underscores the importance of understanding NP-complete problems in the broader context of computational complexity theory and the P vs. NP debate.
Overall, a very informative and comprehensive discussion.

75 words

Permalink Show parent

Re: Week 7

by Manahil Farrukh Siddiqui - Tuesday, 6 August 2024, 2:56 AM

There are two types of NP-complete computing problems: NP-hard and NP (nondeterministic polynomial time) (GeeksforGeeks, 2013a). We can swiftly confirm a possible answer because we are in NP. Any other NP problem that can be effectively transformed into it is said to be NP-hard. It is possible to solve all NP issues rapidly if we can solve an NP-hard problem quickly (Lark, 2024).

NP-complete issues are complex because there needs to be an easy technique to solve them. A central open question in computer science concerning P and NP could be resolved if we could swiftly solve an NP-complete problem, enabling us to solve any NP problem (Haroon, 2023). These problems are complicated because, although it is quick to look for a solution, it frequently takes a long time to locate one, mainly as the problem grows larger (Haroon, 2023).

An NP-complete problem is the Traveling Salesman Problem (TSP). It entails figuring out the quickest path that makes one stop at each city before returning to the beginning. It is fast to calculate the distance and determine whether a route is feasible (GeeksforGeeks, 2013b). TSPs are complex NP-hard issues, as demonstrated by the fact that many NP problems can be converted into TSPs. Resolving a TSP can facilitate the solution of the Hamiltonian Cycle problem.

However, another example is the Knapsack dilemma. It entails determining how to fill a backpack with items while underweight restrictions (Haroon, 2023). Because solving it aids in solving other NP problems, such as employment scheduling, this problem is NP-hard.

Two things must be demonstrated for a problem to be NP-complete. First and foremost, it must be in the NP to verify any solution rapidly. Second, we must demonstrate that it can effectively transform from an existing NP-complete issue. To demonstrate that the Hamiltonian Cycle issue is NP-complete, for instance, we need to demonstrate that it is in NP and can be logically transformed from the 3-SAT problem.

In a nutshell, comprehension of NP-complete issues is essential to comprehending computational complexity. The main P vs. NP debate in computer science would be resolved by solving these issues rapidly, which makes them troublesome. Because of their complexity, NP-complete problems aid in our understanding of the boundaries and powers of computers.

References

GeeksforGeeks. (2013, August 22). Introduction to NP-Completeness. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-np-completeness/

Haroon, R. (2023, September 13). Decoding Complexity: A Deep Dive into NP-complete Problems. Nerd for Tech. https://medium.com/nerd-for-tech/decoding-complexity-a-deep-dive-into-np-complete-problems-fe72d8efc677

Lark. (2024). Np Hard Definition of Np Hardness. Larksuite.com. https://www.larksuite.com/en_us/topics/ai-glossary/np-hard-definition-of-np-hardness

410 words

Permalink Show parent

Re: Week 7

by Prince Ansah Owusu - Thursday, 8 August 2024, 4:51 AM

Your submission offers a solid overview of NP-complete problems with relevant examples. However, it would benefit from clearer definitions and more precise language. Also, ensure all references are properly formatted and checked for credibility.

34 words

Permalink Show parent

Re: Week 7

by Winston Anderson - Tuesday, 6 August 2024, 5:27 AM

An NP-complete problem is a type of computational problem that is both in the class NP (nondeterministic polynomial time) and as hard as any problem in NP. This means that any problem in NP can be transformed into an NP-complete problem in polynomial time. NP-complete problems are considered "hard" because, although their solutions can be verified quickly (in polynomial time), there is no known algorithm to solve them quickly (in polynomial time) for all instances (GeeksforGeeks, 2023; Britannica, 2024).

Characteristics of NP-Complete Problems

1. Decision Problems: NP-complete problems are decision problems, meaning they have a yes-or-no answer.

2. Polynomial-Time Verification: If a solution to an NP-complete problem is given, it can be verified in polynomial time.

3. Polynomial-Time Reduction: Any problem in NP can be reduced to an NP-complete problem in polynomial time. This means that if we could solve one NP-complete problem quickly, we could solve all NP problems quickly.

Why NP-Complete Problems are Considered Hard

The difficulty of NP-complete problems stems from the fact that no polynomial-time algorithm is known for solving any NP-complete problem. The lack of such an algorithm means that the time required to solve these problems increases rapidly as the problem grows, often exponentially (GeeksforGeeks, 2023; Britannica, 2024). This makes them intractable for large instances.

The famous "P vs NP" problem in computer science asks whether every problem whose solution can be quickly verified (NP) can also be quickly solved (P). If P = NP, it would mean that every NP-complete problem could be solved in polynomial time, revolutionizing fields that rely on solving complex problems (GeeksforGeeks, 2023; Britannica, 2024).

Examples of NP-Complete Problems

1. Travelling Salesman Problem (TSP): Given a list of cities and the distances between each pair of cities, the problem asks whether there exists a route that visits each city exactly once and returns to the origin city with a total distance less than or equal to a given number. While verifying a given route is easy (sum up the distances and compare), finding the optimal route is computationally hard (GeeksforGeeks, 2023; Britannica, 2024).

2. Boolean Satisfiability Problem (SAT): This problem involves determining if there exists an assignment of truth values to variables that makes a given Boolean formula true. For example, given the formula (A \lor \neg B) \land (B \lor C), we need to find if there is a combination of true/false values for A, B, and C that satisfies the formula. Verifying a given assignment is straightforward, but finding such an assignment is difficult (GeeksforGeeks, 2023; Britannica, 2024).

Methodology and Reduction

To understand how problems are reduced to NP-complete problems, consider the SAT problem. Many problems can be transformed into SAT:

- Graph Coloring: Can we color a graph with three colors such that no two adjacent vertices share the same color? This can be reduced to a SAT problem by creating a Boolean formula that represents the coloring constraints.

- Hamiltonian Cycle: Does a given graph contain a cycle that visits each vertex exactly once? This can also be reduced to SAT by encoding the constraints of the cycle into a Boolean formula.

Conclusion

NP-complete problems are central to the field of computational complexity because they encapsulate the difficulty of a wide range of problems. Solving any NP-complete problem efficiently would imply that all problems in NP can be solved efficiently, which remains one of the most important open questions in computer science (GeeksforGeeks, 2023; Britannica, 2024).

References

Britannica. (2024, June 28). *NP-complete problem*. Encyclopedia Britannica. https://www.britannica.com/science/NP-complete-problem

GeeksforGeeks. (2023f, October 3). P, NP, CoNP, NP hard and NP complete Complexity Classes. GeeksforGeeks. https://www.geeksforgeeks.org/types-of-complexity-classes-p-np-conp-np-hard-and-np-complete/

597 words

Permalink Show parent

Re: Week 7
by Romana Riyaz (Instructor) - Tuesday, 6 August 2024, 10:39 AM

Hello Winston,
Thank you for your submission. Your explanation of NP-complete problems effectively conveys their significance in computational complexity. By outlining their characteristics, such as being decision problems and having polynomial-time verification and reduction, you provide a clear understanding of why these problems are considered challenging. The emphasis on the absence of known polynomial-time algorithms for solving NP-complete problems highlights their intractability for large instances. The examples of the Traveling Salesman Problem and Boolean Satisfiability Problem illustrate the practical implications of these theoretical concepts. Additionally, discussing the "P vs NP" problem underscores the broader importance of NP-complete problems in computer science. Overall, your detailed and structured explanation offers a comprehensive overview of NP-complete problems and their role in computational theory.
Regards,
Romana
122 words

Permalink Show parent

Re: Week 7
by Loubna Hussien - Wednesday, 7 August 2024, 12:46 AM

You did great work, keep it up.
7 words

Permalink Show parent

Re: Week 7
by Wingsoflord Ngilazi - Wednesday, 7 August 2024, 3:57 PM

Your submission clearly demonstrates how well you have grasped the NP-complete problem. Keep up the good work.
17 words

Permalink Show parent

Re: Week 7
by Jerome Bennett - Thursday, 8 August 2024, 8:23 AM

Greetings Winston,

You did well explaining why NP-complete problems are a big deal in computer science. You broke down what makes a problem NP-complete in a way that's easy to grasp. The main point you drove home is that we don't know any quick ways to solve these problems when they get really big. That's what makes them such a headache for computer scientists and why they're so important in the field.
72 words

Permalink Show parent

Re: Week 7

by Anthony Jones - Tuesday, 6 August 2024, 10:50 PM

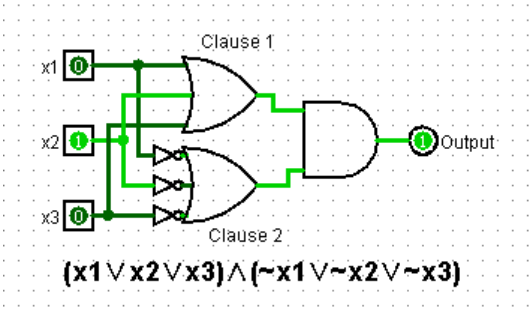
Problem complexity is an important aspect of algorithm development. Quite a number of problems can be solved in polynomial time (that is, the asymptotic complexity for solving the problem is some polynomial like $O(n^K)$, K being a constant, of course). We consider this problem to be within the set of problems P representing all polynomially computable problems. However, there are problems that we don't know polynomial solutions for, but candidate solutions are polynomially easy to check and these are called NP problems.

NP

NP refers to Nondeterministic Polynomial problems. These are problems where we cannot easily determine plausible solutions (Nondeterministic), however we can check possible solutions within polynomial time. For example, Sudoku is a NP problem because it is difficult to solve a puzzle $O(n^{n*n})$ (GeeksforGeeks, 2012), but easy to check a solution $O(n^2)$ (GeeksforGeeks, 2020). Often we think of NP as deternining if a solution exists and we check that by finding a solution or exhausting all possible solutions that could bring about a solution. There is a similar category of problems called co-NP where we try to prove that a there is a solution that does not satisfy the problem. Within NP, there is a category of problems that share the same core problem, such that finding a good algorithm for one would contribute to the other problems. This section of problems is called NP complete

NP Complete

NP Complete problems reduce to other NP complete problems. The most famous NP complete problem is 3SAT which determines if a given circuit of logic gates (or alternatively an equation of boolean agebra) could ever evaluate to True. 3SAT uses clauses of 3 inputs (which may be inverted) that are all combined using the OR operation. Then we combine all the AND operations and we get the result. It's easy to check the results, just perform the boolean algebra or construct the circuit and check, but it is difficult to find a plausible solution. Examples of a very simple valid 3SAT circuit:



Knowing how to solve 3SAT efficiently would provide insight into other problems like sudoku or protein folding because they are all NP complete and can reduce to each other (Hackerdashery, 2014).

NP complete difficulty

NP complete problems are hard because we cannot easily find feasible solutions, in fact it is usually exponential or worse. 3SAT is O(2^n) to check in the worst case. This makes it impractical to compute for more complex 3SAT circuits. That means we currently need to use a heuristic, branch and bound, or some way of making educated guesses and eliminating bad guesses early on in order to solve the problems, but it still performs poorly even in these cases. So, NP complete problems are hard because they do not run in polynomial or faster time, but instead run at exponential or worse time.

References

GeeksforGeeks. (2012, July 14). *Algorithm to solve sudoku*. GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/sudoku-backtracking-7/#

GeeksforGeeks. (2020, October 21). *Check if given sudoku solution is valid or not*. GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/check-if-given-sudoku-solution-is-valid-or-not/

Hackerdashery. (2014, August 26). *P vs. NP and the computational complexity zoo*. Www.youtube.com. https://youtu.be/YX40hbAHx3s

518 words

Permalink Show parent

Re: Week 7
by Loubna Hussien - Wednesday, 7 August 2024, 12:45 AM

You did great work, keep it up.
7 words

Permalink Show parent

Re: Week 7
by Wingsoflord Ngilazi - Wednesday, 7 August 2024, 3:54 PM

I sincerely enjoyed reading your post. The examples are clear, relevant and insightful.
13 words

Permalink Show parent

Re: Week 7
by Chong-Wei Chiu - Thursday, 8 August 2024, 10:40 AM

Hello, Anthony Jones. Thank you for sharing your point of view in this week's discussion. You explained the definitions of NP problems and NP complete problems, making the whole post easier to understand. Furthermore, you used SAT as an example to illustrate NP-complete problems, which makes your post more complete.
50 words

Permalink Show parent

Re: Week 7
by Loubna Hussien - Wednesday, 7 August 2024, 12:24 AM

An NP-complete problem is a decision problem that can be verified in polynomial time but cannot be solved in polynomial time unless P=NP, according to computational complexity theory. Algorithms are often tested against NP-complete problems. A problem is considered NP-complete if every other NP-complete problem can be solved in polynomial time. This implies that if an efficient algorithm exists for one NP-complete problem, then an efficient algorithm exists for all NP-complete problems (Britannica, 2023).

Several NP-complete problems are significant in real life, such as the traveling salesman problem and the knapsack problem. These problems are deemed "hard" because they are difficult to solve in polynomial time. They are often used as models for real-world issues, like finding the shortest route between cities or packing the most items into a given space.

Various methods exist to solve NP-complete problems, but none guarantee efficiency. The most common method is to use a heuristic algorithm, which often finds a good solution within a reasonable time but does not always find the optimal solution. Another approach is to use a metaheuristic algorithm, which employs a heuristic algorithm to explore a search space. Although often more efficient than heuristic algorithms, metaheuristic algorithms can still be time-consuming (Quora, n.d.).

One of the most important unresolved problems in computer science is the P versus NP dilemma. If P=NP, all NP-complete problems can be solved in polynomial time. This would profoundly impact many areas of computer science, including artificial intelligence, optimization, and cryptography. Conversely, if P≠NP, it would indicate that some problems are inherently difficult to solve in polynomial time, necessitating the development of new approaches such as heuristic and metaheuristic algorithms.

Examples of NP-complete problems include the traveling salesman problem, the knapsack problem, and the Boolean satisfiability problem (Science & Science, 2022). These are just a few of the many NP-complete problems that exist. Computer scientists continue to study NP-complete problems because they are a critical part of the field.

References:

Britannica, T. (2023). *NP-complete problem. Encyclopedia Britannica.*

https://www.britannica.com/science/NP-complete-problem

Quora. (n.d.). *Were there any NP complete problems which are solved now and have solution in polynomial time?* https://www.quora.com/Were-there-any-NP-complete-problems-which-are-solved-now-and-have-solution-in-polynomial-time

Science, B. O. C., & Science, B. O. C. (2022). P, NP, NP-Complete and NP-Hard Problems in Computer Science | Baeldung on Computer Science. *Baeldung on Computer Science*. https://www.baeldung.com/cs/p-np-np-complete-np-hard

380 words

Permalink Show parent

Re: Week 7

by Winston Anderson - Wednesday, 7 August 2024, 3:01 AM

Hi Loubna,
The definition of NP-complete problems is mostly accurate but could be more precise. Specifically, stating that an NP-complete problem is one that is both in NP and as hard as any problem in NP. The statement "A problem is considered NP-complete if every other NP-complete problem can be solved in polynomial time" is misleading. It should state that a problem is NP-complete if any problem in NP can be reduced to it in polynomial time.
77 words

Permalink Show parent

Re: Week 7

by Tamaneenah Kazeem - Wednesday, 7 August 2024, 12:34 AM

Describe in your own words what a NP complete problem is and why a NP complete program is considered to be 'hard.'

Include one or two examples to explain your thought process to show what is occurring and how the methodology works.

NP complete problems (NPCs) can be defined as problems which fall in-between the categories of NP-hard and NP. These types of problems, NP-hard and NP, have 2 main characteristics:

- NPs are problems that when given a solution, their accuracy can be validated in polynomial time. This ensures verifying the solution relatively quickly.
- On the other hand, NP-hard problems are those without an efficient solution algorithm. That is, there is no polynomial time algorithm that is known to solve it. If a solution is found for an NP-complete problem, then that can be used to solve all other NP-hard problems.

An NP-complete problem is considered "hard" because, there are currently no polynomial-time algorithms discovered to solve any NP-complete problem. If we were to find a polynomial-time algorithm for any NP-complete problem, it would change computer science as we know it by providing polynomial-time solutions to all NP problems. However, till today, such an algorithm has not been found, and it is widely believed that no such algorithm exists. So, P≠NP.

Example: Travelling Salesman Problem (TSP)

A classic example of an NP-complete problem is the Travelling Salesman Problem (TSP).

The goal of TSP is for a salesman to visit each city exactly once and then return to the starting city, minimizing the total travel distance.

- **Verification**: If provided with a specific route, we can quickly compute the total distance and check if the route visits each city exactly once and returns to the starting point, ensuring it meets the criteria.
- **Implication**: If an efficient (polynomial-time) solution to TSP were found, it could be used to solve other NP problems by transforming them into an instance of TSP. Unfortunately, such a solution has not been found yet.

References:

Team, S. E. (n.d.). *NP Complete*. Retrieved from StudySmarter: https://www.studysmarter.co.uk/explanations/computer-science/theory-of-computation/np-complete/

Team, S. E. (n.d.). *NP Hard Problems*. Retrieved from StudySmarter: https://www.studysmarter.co.uk/explanations/computer-science/theory-of-computation/np-hard-problems/

348 words

Permalink Show parent

Re: Week 7

by Michael Oyewole - Wednesday, 7 August 2024, 1:21 AM

A subclass of NP (nondeterministic polynomial time) issues is known as NP-complete problems. A class of computing issues known as NP problems is those that can be verified by a deterministic computer in polynomial time and solved in polynomial time by a non-deterministic machine (Geeksforgeeks, 2024). If every other NP problem can be reduced to issue L in polynomial time, then problem L is NP-complete. Every problem in NP can be solved in polynomial time if any NP-complete problem can be solved in that amount of time. The most difficult issues in the NP set are those that are NP-complete.

If a decision problem L satisfies both of the following two criteria, it is NP-complete:
1. NP contains L. (NP-complete problems have no known efficient solution, but any solution may be tested fast).
2. All NP problems can be reduced to L in polynomial time (the definition of reduction is given below).

If an issue follows Property 2 above and need not follow Property 1, then it is NP-Hard. Consequently, if a problem is both NP and NP-hard, it is NP-complete (Geeksforgeeks, 2024).
One example of a problem that we are unable to solve in polynomial time is circuit satisfiability. A boolean circuit, which consists of several AND, OR, and NOT gates connected by wires, is the input for this problem. We'll assume that the circuit is free of loops, which means it doesn't contain any flip-flops or delay lines. The circuit receives a set of m boolean (TRUE/FALSE) values, denoted by the values x1,..., xm (Geeksforgeeks, 2024). There is just one boolean value in the output.
Since we can compute the output of a k-input gate in O(k) time, we can use depth-first search to calculate the circuit's output in polynomial (really, linear) time given particular input values.

Reference
Geeksforgeeks. (2024 May 15). Introduction to NP-Complete Complexity Classes. https://www.geeksforgeeks.org/introduction-to-np-completeness/

310 words

Permalink Show parent

Re: Week 7

by Winston Anderson - Wednesday, 7 August 2024, 3:17 AM

Hi Michael,
The initial definition of NP-complete problems is mostly correct. The explanation could be more structured and concise, separating the definitions of NP, NP-complete, and NP-hard more clearly. The example provided is good.

34 words

Permalink Show parent

Re: Week 7

by Muritala Akinyemi Adewale - Wednesday, 7 August 2024, 3:12 AM

NP-Complete Problems: The Everest of Computational Complexity

What is an NP-Complete Problem?

An NP-complete problem is a decision problem (one with a yes/no answer) that is both:

- **NP (Non-deterministic Polynomial Time):** This means that if a proposed solution is given, it can be verified in polynomial time to determine if it's correct.
- **NP-hard:** This implies that any problem in the NP class can be transformed (reduced) into this problem in polynomial time. In essence, it's at least as hard as any other problem in NP.

Why are NP-Complete Problems Hard?

NP-complete problems are considered "hard" because no one has yet found a deterministic algorithm that can solve them in polynomial time. This means that as the problem size grows, the time it takes to solve the problem grows exponentially.

Examples of NP-Complete Problems:

- **Traveling Salesman Problem:** Given a list of cities and their distances, find the shortest possible route that visits each city exactly once and returns to the starting city.
- **The Satisfiability Problem (SAT):** Given a Boolean formula, determine if there exists an assignment of truth values to the variables that makes the formula true.

The Challenge of NP-Completeness:

Imagine trying to find the shortest possible route between cities (Traveling Salesman Problem). A brute-force approach would involve calculating the distance of every possible route, which grows exponentially with the number of cities. Even for a relatively small number of cities, this becomes computationally infeasible.

While verifying a proposed solution (checking if a given route is indeed the shortest) can be done efficiently, finding the optimal solution itself remains a daunting challenge.

The Importance of Understanding NP-Completeness:

Recognizing NP-complete problems is crucial in algorithm design. When faced with such a problem, it's often practical to seek approximate solutions or heuristics rather than pursuing an exact, optimal solution that might be computationally intractable.

Reference:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

323 words

Permalink Show parent

Re: Week 7

by Winston Anderson - Wednesday, 7 August 2024, 3:20 AM

Hi Muritala,
The definitions of NP and NP-hard are accurate and clearly explained. The explanation is well-structured, clear, and easy to understand.
The examples provided (Traveling Salesman Problem and SAT) are appropriate and relevant.
The explanation of why NP-complete problems are hard and their importance is well-articulated.
47 words

Permalink Show parent

Re: Week 7

by Wingsoflord Ngilazi - Wednesday, 7 August 2024, 4:40 AM

Describe in your own words what a NP complete problem is and why a NP complete program is considered to be 'hard.'

Include one or two examples to explain your thought process to show what is occurring and how the methodology works. Use APA citations and references for any sources used.

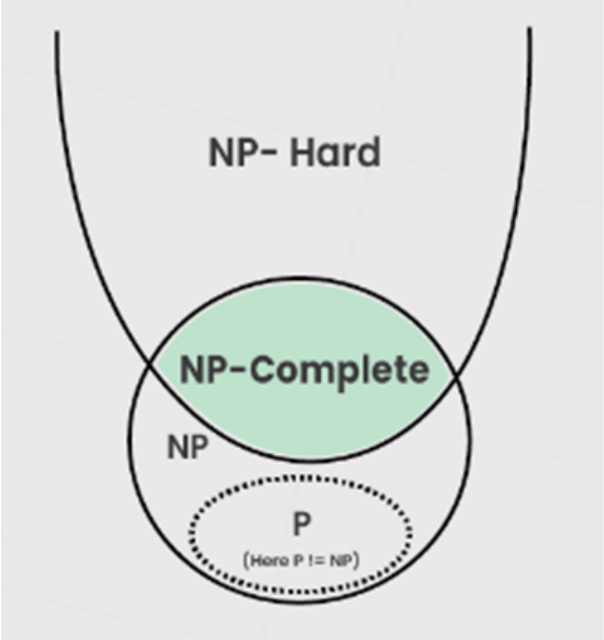
An NP-complete problem is a type of computational problem that is both in the class NP (nondeterministic polynomial time) and as hard as any problem in NP, meaning that any NP problem can be transformed into it in polynomial time (Kaye, 2000). These problems are significant in theoretical computer science because if one NP-complete problem can be solved in polynomial time, then every problem in NP can also be solved in polynomial time, effectively proving P=NP (GeeksforGeeks, 2024).

A decision problem P is NP-complete if it meets the following two criteria:

i. P is in NP (Any solution to NP-complete problems can be quickly verified, but no efficient solution is known).

ii. Every problem in NP is reducible to P in polynomial time

It is important to note that a problem is NP-hard if it satisfies the second criterion but not necessarily the first. Therefore, a problem is NP-complete if it is both in NP and NP-hard. Reduction is a process in computational theory where one problem is transformed into another problem in a way that a solution to the second problem can be used to solve the first problem. This transformation must be done in polynomial time, ensuring that solving the second problem is no harder than solving the first problem. The following diagram shows the place of NP-Complete problems (GeeksforGeeks, 2024).



Why NP-Complete Problems are Considered Hard

No polynomial-time algorithm has been discovered for any NP-complete issue, despite substantial study. This suggests that we are currently unable to solve these issues fast (in polynomial time (Kaye, 2000). Additionally, every problem in NP might potentially be solved in polynomial time if a polynomial-time algorithm for any NP-complete problem were to be found. This would represent a significant advancement in computer science (Kaye, 2000.

Below are popular problems that can be classified as NP-Complete:

i. Travelling Salesman Problem (TSP):

Problem

Determine the shortest path that visits each city precisely once and returns to the initial city given a list of cities and their respective distances.

Why it is hard

The problem is difficult because, for big datasets, brute force solutions are unfeasible due to the fact that the number of viable routes rises factorially with the number of cities.

Solution

Although approximation methods and dynamic programming are frequently employed, the precise polynomial-time solution is still unknown.

ii. Knapsack Problem:

Problem

Given a set of items, each with a weight and value, decide how many of each item to put in a collection so as to maximize the total value and keep the overall weight within a certain range.

Why it is hard

Solving by brute force requires a large amount of computing power due to the exponential number of possible possibilities.

Solution

To obtain approximate answers or efficiently solve smaller cases, strategies such as greedy algorithms and dynamic programming are employed.

References

GeeksforGeeks. (2024, May 15). *Introduction to NP-Complete Complexity Classes*. https://www.geeksforgeeks.org/introduction-to-np-completeness/

Kaye, R. (2000). Minesweeper is NP-complete. *Mathematical Intelligencer*, *22*(2), 9-15.

542 words

Permalink Show parent

Re: Week 7

by Romana Riyaz (Instructor) - Wednesday, 7 August 2024, 10:31 AM

Hello Nglazi,
Thank you for your submission. Your explanation of NP-complete problems is clear and thorough, effectively outlining the characteristics and significance of these problems in computational theory. You accurately describe the criteria for a problem to be classified as NP-complete and highlight the concept of polynomial-time reduction. The examples of the Travelling Salesman Problem (TSP) and the Knapsack Problem illustrate well why NP-complete problems are considered difficult, emphasizing the exponential growth of possible solutions and the limitations of brute force methods. The use of APA citations strengthens your argument by providing credible sources. Overall, your explanation is well-organized and insightful, providing a solid understanding of NP-complete problems and their complexity.

Regards,
Romana
113 words

Permalink Show parent

Re: Week 7
by Wingsoflord Ngilazi - Wednesday, 7 August 2024, 3:50 PM

Dear Prof,
Thank you for the positive feedback.
8 words

Permalink Show parent

Re: Week 7
by Tyler Huffman - Wednesday, 7 August 2024, 6:54 AM

Describe in your own words what a NP complete problem is and why a NP complete program is considered to be 'hard.'

Nondeterministic polynomial time (or NP) is a complexity class for decision problems; decision problems are a set of problems that only have two output possibilities: "yes" or "no". As expected, NP-complete is a subclass. NP-complete has a few additional requirements that must be met for a problem to be considered NP-complete. In more technical terms, an NP problem must additionally be what is called "NP-Hard" to be NP-complete. Here is how Britannica puts it: "If a problem is NP and all other NP problems are polynomial-time reducible to it, the problem is NP-complete" (2024). These are problems for which we can nearly be sure there is no efficient, polynomial based solution. As we see based on these requirements, problems that fall within the NP-complete category have no currently known efficient algorithm to solve them. The traveling salesman problem is an example of this; Britannica states "The only known general solution algorithm that guarantees the shortest path requires a solution time that grows exponentially with the problem size (i.e., the number of cities)" (n.d.).

This is indeed the reason an NP-complete problem is considered "hard" in the computer science sense of the word. There is simply no efficient algorithm to solve them and therefore for large data sets, significant computational resources are required.

References

Britannica. (2024, June 28). NP-complete problem. https://www.britannica.com/science/NP-complete-problem

Britannica. (n.d.). Traveling salesman problem. https://www.britannica.com/science/traveling-salesman-problem
248 words

Permalink Show parent

Re: Week 7
by Tamaneenah Kazeem - Thursday, 8 August 2024, 4:55 AM

Hi Tyler.
Your explanation of NP-complete problems is clear but can be more concise. NP-complete problems are decision problems in the NP class that are also NP-hard, meaning every problem in NP can be reduced to them in polynomial time. They are considered "hard" because no efficient algorithm is known to solve them. This makes them computationally intensive for large datasets, as could be exemplified by the traveling salesman problem.

Overall, well done.
73 words

Permalink Show parent

Re: Week 7

by Tousif Shahriar - Wednesday, 7 August 2024, 1:39 PM

An NP-complete problem is a fascinating class of computational problems that sit at the heart of computer science. These problems are both in NP (nondeterministic polynomial time) and as hard as any problem in NP. In simpler terms, NP-complete problems are those for which a proposed solution can be verified quickly (in polynomial time), but finding the solution in the first place may take an impractically long time. This dual nature of ease and difficulty makes them particularly intriguing (Garey & Johnson, 1979).

These problems are considered hard because they encapsulate the difficulty of all problems in NP. If someone could find a polynomial-time algorithm to solve one NP-complete problem, it would imply that every problem in NP could also be solved in polynomial time. This "holy grail" of computer science, known as the P vs NP problem, remains unsolved and is one of the seven Millennium Prize Problems (Sipser, 2012). The sheer uncertainty and the profound implications of solving any NP-complete problem add to their complexity and allure.

We can consider the following NP-Complete problems:

1. The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is a classic example of an NP-complete problem. Imagine a salesperson trying to visit a list of cities, each once, and return home, all while minimizing travel distance. While it's easy to verify a given route's total distance, figuring out the shortest possible route is computationally intense (Cormen et al., 2009). This reminds me of planning a road trip: it's straightforward to check if your planned route hits all the spots, but finding the optimal route can be mind-boggling when considering all possible city permutations.

Methodology:

Verification: Given a route, sum the distances between consecutive cities and check if the total distance is minimal.

Finding the Solution: Evaluate all possible permutations of cities to find the route with the shortest distance, which can take factorial time in the worst-case scenario.

2. The Knapsack Problem

Another well-known NP-complete problem is the Knapsack Problem. Suppose you have a backpack, and you need to fill it with the most valuable items without exceeding its weight limit. While verifying that a chosen set of items fits the weight limit and calculating their total value is easy, determining the best combination of items is not (Kleinberg & Tardos, 2006). This problem often comes to mind when packing for a trip: you want to maximize what you bring without overloading your bag.

Methodology:

Verification: Given a subset of items, sum their weights and values to ensure they fit within the knapsack's capacity and check if the total value is maximal.

Finding the Solution: Examine all possible subsets of items to identify the combination with the highest value that fits within the capacity, which can take exponential time in the worst-case scenario.

Conclusion

NP-complete problems represent some of the most challenging problems in computational theory. They are characterized by the ease of verifying solutions but the difficulty of finding them. The Traveling Salesman Problem and the Knapsack Problem are prime examples that illustrate both the complexity and the computational intensity inherent to NP-complete problems.

Reference

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman.

Kleinberg, J., & Tardos, É. (2006). Algorithm Design. Pearson.

Sipser, M. (2012). Introduction to the Theory of Computation (3rd ed.). Cengage Learning.

577 words

Permalink Show parent

Re: Week 7
by Wingsoflord Ngilazi - Wednesday, 7 August 2024, 4:02 PM

Your submission is concise, and it clearly demonstrates a good grasp of the NP-Complete problem. Well done.
17 words

Permalink Show parent

Re: Week 7
by Jerome Bennett - Wednesday, 7 August 2024, 5:11 PM

CS 3304 Discussion Forum Unit 7

An NP-complete problem is a type of problem in computer science that is both in NP (nondeterministic polynomial time) and as hard as any problem in NP. I will try to explain in simpler terms below:

NP (Nondeterministic Polynomial time): This is a class of problems for which a given solution can be verified as correct or incorrect in polynomial time (i.e., the time taken to check the solution is reasonable and scales nicely with the size of the input). However, finding the solution itself may not be easy or quick (Shaffer, 2009).

NP-Complete: These problems are the hardest problems within NP. If you can find a polynomial-time solution to one NP-complete problem, you can translate that solution to solve all other NP problems in polynomial time as well. Essentially, solving one NP-complete problem efficiently would mean you could solve all NP problems efficiently (Shaffer, 2009).

The reason NP-complete problems are considered hard is that no one has found a polynomial-time algorithm to solve any of them. If such an algorithm existed, it would revolutionize fields that rely on complex problem-solving, like cryptography, scheduling, and optimization. However, algorithms such as “Kookaburra” have been developed to help solve them, or at least, optimize the process (The Lead, 2017).

Example: The Traveling Salesman Problem (TSP)

Imagine you're a salesman who needs to visit a number of cities. The goal is to find the shortest possible route that visits each city exactly once and returns to the starting city. This is known as the Traveling Salesman Problem (TSP), a classic NP-complete problem.

Here's how the problem works:

Input: A list of cities and the distances between each pair of cities.

Objective: Find the shortest possible route that visits each city once and returns to the starting point.

For a small number of cities, you could list all possible routes and compare their lengths to find the shortest one. However, as the number of cities increases, the number of possible routes grows factorially (i.e., extremely fast). For example, with 10 cities, there are over 3 million possible routes to check.

Methodology in Practice:

Brute Force Approach: Check all possible routes and pick the shortest one. This works for small numbers of cities but becomes impractical as the number of cities grows due to the exponential increase in possibilities (Kuo, 2024).

Heuristics and Approximation: Since finding the exact solution is often impractical, we use heuristic methods (like the nearest neighbor, genetic algorithms, or simulated annealing) to find a good-enough solution more quickly (Kuo, 2024).

Verification: Once you have a proposed route, you can easily check if it's valid (it visits each city once and returns to the start) and calculate its total distance to see if it's the shortest found so far. This verification step is polynomial time (Kuo, 2024).

The TSP exemplifies the difficulty of NP-complete problems: while checking a solution is straightforward, finding the solution is not. Despite numerous attempts, no polynomial-time algorithm has been discovered to solve NP-complete problems, which is why they are considered "hard."

Reference

Shaffer, C. (2009). Lecture 21: NP-Hard Problems. A Practical Introduction to Data Structures and Algorithm Analysis. https://my.uopeople.edu/pluginfile.php/1861950/mod_book/chapter/512270/Jeff_nphardRA7.pdf

Kuo, M (2024). Algorithms for the Travelling Salesman Problem. https://www.routific.com/blog/travelling-salesman-problem

The Lead. (2017). 'Missing puzzle piece' to help solve the infamous Travelling Salesman Problem. https://phys.org/news/2017-04-puzzle-piece-infamous-salesman-problem.html#google_vignette

553 words

Permalink Show parent

Re: Week 7
by Michael Oyewole - Thursday, 8 August 2024, 1:16 AM

Hi Jerome,
Thank you for your post. My contribution is that since NP-complete programs cannot be solved in polynomial time, that is, the amount of time needed to solve the problem grows exponentially with the size of the input, they are regarded as difficult programs.
45 words

Permalink Show parent

Re: Week 7
by Jobert Cadiz - Thursday, 8 August 2024, 8:22 AM

Hi Jerome,
Thanks for sharing your post. You clearly define NP and NP-complete, making the complex concepts more accessible. The Traveling Salesman Problem (TSP) is an excellent example to illustrate the difficulty of NP-complete problems. ou cover

different approaches to solving NP-complete problems, including brute force, heuristics, and approximation methods. Great work, keep it up.

55 words

Permalink Show parent

Re: Week 7

by Aye Aye Nyein - Wednesday, 7 August 2024, 7:26 PM

An NP-complete problem is a category of computational problems characterized by their difficulty and their relationship with NP (nondeterministic polynomial time) problems. These problems are notable for being both within NP and for being as hard as any problem in NP.

Here is a breakdown of why NP-complete problems are considered "hard":

1. **NP Problems**: These are problems for which if given a proposed solution, it can be verified quickly (in polynomial time). However, finding that solution initially might be challenging (Garey & Johnson, 1979).

2. **NP-complete Problems**: These problems are a special subset of NP problems with two crucial properties:

- **Verification**: Solutions can be verified quickly.

- **Hardness**: Any problem in NP can be transformed into an NP-complete problem in polynomial time.

This implies that if you can solve one NP-complete problem efficiently, you can solve all NP problems efficiently (Garey & Johnson, 1979).

NP-complete problems are challenging because no polynomial-time algorithm is known to solve all of them efficiently. Most experts believe such an algorithm does not exist. This issue is closely related to the P vs. NP question, which explores whether problems that can be quickly verified can also be solved quickly (Garey & Johnson, 1979).

Examples of NP-complete Problems:

1. **The Traveling Salesman Problem (TSP)**:

- **Description of the problem**: Find the shortest route that visits each city precisely once and returns to the beginning city, given a set of cities and the distances between each pair.

- **Why It is Hard**: It is not possible to analyze all alternative routes because the number of potential routes rises factorially with the number of cities (n!). Nonetheless, it is not too difficult to determine whether a particular path is the shortest (Garey & Johnson, 1979).

2. **The Knapsack Problem**:

- **Description of the problem**: Given a set of items with specific weights and values, and a knapsack with a weight limit, determine the maximum value of items that can be packed without exceeding the weight limit.

- **Why It is Hard**: The number of possible combinations of items grows exponentially with the number of items, making it challenging to find the optimal combination. Nonetheless, verifying a combination's total value and weight is computationally easy (Garey & Johnson, 1979).

Methodology:

To determine if a problem is NP-complete, the general approach involves:

1. **Demonstrating that the Problem is in NP**: Show that a proposed solution can be checked quickly.

2. **Proving that the Problem is NP-Hard**: Show that any problem in NP can be transformed into the problem in question in polynomial time. This process, known as polynomial-time reduction, involves converting a known NP-complete problem into the problem under consideration (Garey & Johnson, 1979).

In summary, NP-complete problems are considered difficult because they represent a class of problems where finding a solution is computationally intensive, and solving one efficiently implies a breakthrough for all NP problems.

Reference:

Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman and Company.

502 words

Permalink Show parent

Re: Week 7

by Michael Oyewole - Thursday, 8 August 2024, 1:13 AM

Hi Aye,
Thanks for sharing with us. A problem P is called the NP-complete if it satisfies the following two scenarios: L is in the set of NP, and every issue that belongs to NP can be reduced to problem L in a polynomial time. Thanks!

46 words

Permalink Show parent

Re: Week 7

by Liliana Blanco - Wednesday, 7 August 2024, 11:26 PM

An NP-complete problem is a cornerstone in the study of computational complexity theory. These problems exhibit two key characteristics. First, any NP-complete problem belongs to NP, meaning that if a solution is proposed, it can be verified as correct within polynomial time. This implies that the time required to check the solution increases in a way that can be described by a polynomial function relative to the input size. Second, NP-complete problems are as difficult as the toughest problems in NP. Informally, a problem is NP-hard if every other problem in NP can be transformed (or reduced) into it in polynomial time. Thus, if an algorithm is found that solves an NP-hard problem in polynomial time, it would also solve all NP problems in polynomial time.

NP-complete problems are considered 'hard' because, despite extensive research, no polynomial-time algorithm has been discovered to solve any of these problems. This leads to a fundamental question in computer science: does P equal NP? In other words, can every problem that can be verified in polynomial time also be solved in polynomial time? This remains one of the most critical and unresolved questions in theoretical computer science.

The Traveling Salesman Problem (TSP) is a well-known example of an NP-complete problem. This problem involves finding the shortest possible route that visits a list of cities exactly once and returns to the original city, given the distances between each pair of cities. The challenge lies in the factorial growth of possible routes with an increasing number of cities, making it impractical to evaluate every route for larger city sets. This problem is detailed by Dasgupta, Papadimitriou, and Vazirani in their comprehensive guide on algorithms.

Another instance is the Boolean Satisfiability Problem (SAT), which entails determining whether there exists an interpretation that fulfills a specified Boolean formula. For instance, given a formula such as (A or B) and (not A or C), is it possible to assign truth values to A, B, and C in a manner that renders the formula true? SAT was the initial problem to be formally demonstrated as NP-complete. In both of these instances, the process of verifying a proposed solution is direct and can be accomplished quickly. However, discovering the solution itself requires an amount of time that increases exponentially with the size of the input.

References: Dasgupta, S., Papadimitriou, C.H., & Vazirani, U.V. (2006). Algorithms. Berkeley, CA: University of California Berkeley, Computer Science Division. Available at http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

406 words

Permalink Show parent

Re: Week 7

by Michael Oyewole - Thursday, 8 August 2024, 1:10 AM

Hi Liliana
Thanks for the post. An NP-complete problem is the hardest problem in the set of NP problems; an NP-complete problem is a problem that is included in the sets of NP-complete and NP-hard problems. Thanks for sharing your views.
41 words

Permalink Show parent

Re: Week 7

by Prince Ansah Owusu - Thursday, 8 August 2024, 4:50 AM

Your submission provides a thorough explanation of NP-complete problems and includes detailed examples. It effectively captures the complexity and significance of these problems. However, the citation for Dasgupta et al. is a bit informal.
34 words

Permalink Show parent

Re: Week 7

by Christopher Mccammon - Thursday, 8 August 2024, 9:02 AM

Hi Liliana
Your explanation effectively communicates the essence of NP-complete problems, focusing on their definition, significance, and the challenge they present in computational complexity. You clearly articulate that NP-complete problems can be verified in polynomial time, and you introduce the crucial aspect of polynomial-time reductions that make these problems the hardest within NP. The examples provided are pertinent and illustrate the complexity associated with NP-complete problems well. Good job!
69 words

Permalink Show parent

Re: Week 7

by Prince Ansah Owusu - Thursday, 8 August 2024, 2:29 AM

Definition of NP-Complete Problems

An NP-Complete problem is a class of computational problems that are both NP-Hard and in NP. To understand this, we must define two key concepts:

- **NP (Non-deterministic Polynomial time):** This class includes problems for which a solution, if given, can be verified in polynomial time (Shaffer, 2020). In other words, if a solution is proposed, its correctness can be quickly checked.
- **NP-Hard:** These problems are as difficult as the hardest problems in NP. Solving one NP-Hard problem efficiently implies that all NP-Hard problems can be solved efficiently (Dasgupta, Papadimitriou, & Vazirani, 2008). NP-Hard

problems are not required to be in NP and may not have solutions that are verifiable in polynomial time.

An NP-Complete problem is both in NP and NP-Hard. This means that an NP-Complete problem can be verified quickly (in polynomial time) and is as challenging as the hardest problems in NP. The significance of NP-Completeness is that if an efficient (polynomial-time) algorithm is found for one NP-Complete problem, it would imply that all NP-Complete problems can be solved efficiently (Shaffer, 2020).

Why NP-Complete Problems Are Considered 'Hard'

NP-Complete problems are considered 'hard' because:

1. **No Known Polynomial-Time Solutions:** Despite extensive research, no polynomial-time algorithms are known for solving NP-Complete problems. As the size of the problem increases, the time required to find an exact solution grows exponentially, making it impractical for large instances (Dasgupta, Papadimitriou, & Vazirani, 2008).
2. **Intractability:** If an efficient solution were found for any NP-Complete problem, it would imply that all NP problems could be solved efficiently. Since no such solution is known, NP-Complete problems are often addressed with approximation algorithms or heuristics (Shaffer, 2020).

Examples

1. **Traveling Salesman Problem (TSP):** The TSP involves finding the shortest route that visits a set of cities exactly once and returns to the starting point. Although verifying a given route's length is polynomial-time feasible, finding the shortest route requires an exhaustive search, which becomes computationally infeasible as the number of cities increases (Dasgupta, Papadimitriou, & Vazirani, 2008).
2. **Knapsack Problem:** This problem involves selecting a subset of items, each with a weight and value, to maximize total value without exceeding the weight capacity of the knapsack. While checking whether a given set of items fits within the capacity and meets a value criterion is polynomial-time feasible, finding the optimal combination requires an exhaustive search, which becomes impractical for large numbers of items (Shaffer, 2020).

Conclusion

NP-Complete problems are pivotal in computational complexity theory as they represent some of the most challenging problems in computer science. Understanding their properties aids in developing efficient algorithms and approximation techniques, even though exact solutions remain elusive for large instances.

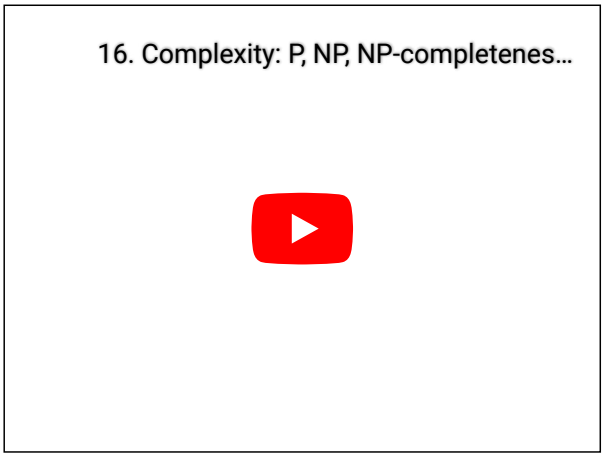
References

Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2008). *Algorithms*. McGraw-Hill.

Shaffer, C. A. (2020). *A Practical Introduction to Data Structures and Algorithm Analysis*. CRC Press.

Lecture 21: NP-Hard Problems. Retrieved from pdf

Reductions and NP-Completeness Lecture Notes. Retrieved from



.

482 words

Permalink Show parent

Re: Week 7
by Tamaneenah Kazeem - Thursday, 8 August 2024, 4:59 AM

Hello Prince
Your explanation of NP-Complete problems is clear and thorough. The examples provided are excellent and help illustrate the complexity and challenges of NP-Complete problems effectively. Great job citing and referencing sources in the correct format!
37 words

Permalink Show parent

Re: Week 7
by Liliana Blanco - Thursday, 8 August 2024, 8:34 AM

I really enjoyed your explanation on NP-complete issues. Your explanations of the Traveling Salesman Problem and the Knapsack Problem made the concepts of NP and NP-Hard readily apparent and showed why these issues are so challenging. I agree that the difficulty of locating polynomial-time solutions is highlighted by NP-complete problems, which are essential to computational complexity theory. Your investigation validates my views regarding the significance of these issues in developing effective algorithms.
72 words

Permalink Show parent

Re: Week 7
by Siraajuddeen Adeitan Abdulfattah - Thursday, 8 August 2024, 2:45 AM

NP-complete is a computational type of problem which belongs to the class called NP (nondeterministic polynomial time). It is considered hard because of the resources (computational) required to solve it, which is quite significant. For example, a problem X is NP complete if there is an NP problem Y, such that Y is reducible to X in polynomial time. NP-complete problems are as hard as NP problems, a problem is NP-complete if it is a part of both NP and NP-Hard problem (Geeksforgeeks, 2023.). A typical example is the traveling salesman problem, which shows that given a list of cities and the distances between each pair of cities, as well as the task of finding the shortest possible route to each city once and return to the starting city. This shows that it is difficult to find an optimal solution in polynomial time, thereby making the traveling salesman problem an NP-complete problem.

Reference:

Geeksforgeeks, (Jan,2023). Difference between NP hard and NP complete problem. Retrieved from:
https://www.geeksforgeeks.org/difference-between-np-hard-and-np-complete-problem/
166 words

Permalink Show parent

Re: Week 7
by Prince Ansah Owusu - Thursday, 8 August 2024, 4:48 AM

Your submission provides a clear explanation of NP-complete problems and offers a good example with the traveling salesman problem. However, it could benefit from a bit more detail on why NP-complete problems are considered 'hard' and a more formal APA citation.
41 words

Permalink Show parent

Re: Week 7
by Tamaneenah Kazeem - Thursday, 8 August 2024, 5:04 AM

Your explanation of NP-complete problems effectively highlights their complexity and the substantial resources required to solve them. However, some points could be clearer and more concise. For instance, clarify the difference between NP and NP-complete with more precision.
Overall, great work!
41 words

Permalink Show parent

Re: Week 7
by Chong-Wei Chiu - Thursday, 8 August 2024, 10:34 AM

Hello, Siraajuddeen Adeitan Abdulfattah. Thank you for sharing your opinion about NP complete problems. In your post, you illustrate the basic concepts of NP complete problems with brief descriptions. However, I think you can focus on your example and explain the reasons why it is an NP complete problem and the hardest part of this problem.
56 words

Permalink Show parent

Re: Week 7
by Christopher Mccammon - Thursday, 8 August 2024, 5:51 AM

In the realm of computer science NP problems stand out as a concept that highlights the boundaries of computational efficiency. These problems are renowned, for their complexity and the formidable challenge they pose to researchers and professionals. To understand NP problems one must first grasp NP (polynomial time) problems, where a proposed solution can be swiftly verified within a polynomial timeframe.
NP complete problems possess two characteristics. Initially they fall under NP category indicating that any solution can be verified efficiently in time. Additionally they are just as challenging as any problem in NP since every NP issue can be transformed into an NP problem without difficulty in polynomial time. This transformation ability implies that finding a solution for NP problems would also lead to solutions for all NP issues(Khan Academy,n.d).
The intrinsic complexity of NP problems stems from the absence of known algorithms, for resolving them and the prevailing belief that such algorithms might not exist. As the size of these problems increases the time needed to solve them tends to grow. For example consider the Traveling Salesman Problem (TSP) which involves determining the route visiting a series of cities before returning to the starting point(Khan Academy,n.d)..
While it's easy to confirm a route determining the efficient one becomes difficult as the number of cities increases due, to the numerous possible routes. In a vein the Knapsack Problem involves selecting items to maximize value without exceeding a weight limit. While it's simple to check if a set of items fits within the weight limit finding the best combination requires effort (Karp, 1972).
To tackle NP problems various strategies are employed. Approximation algorithms aim to provide solutions that're close to optimal within time offering practical but not necessarily perfect results. Heuristic methods offer satisfactory solutions without guaranteeing optimality. Exponential time algorithms can precisely solve NP problems. Become impractical for large inputs due to their high computational requirements (GeeksforGeeks, 2021).

In summary NP complete problems highlight the challenges in complexity and algorithm efficiency. These issues play a role in defining what is computationally achievable. Despite their significance effective solutions remain challenging to find leading investigations into approximation techniques and heuristics well, as exploring exponential algorithms.

References:
GeeksforGeeks. (2021). What is NP-complete problem? https://www.geeksforgeeks.org/what-is-np-complete-problem/
Khan Academy. (n.d.). NP-completeness. https://www.khanacademy.org/computing/computer-science/algorithms/np-completeness/a/np-completeness
Karp, R. M. (1972). Reducibility among combinatorial problems. In Complexity of Computer Computations (pp. 85-103). Springer. https://doi.org/10.1007/978-1-4684-2001-2_9
395 words

Permalink Show parent

Re: Week 7
by Jobert Cadiz - Thursday, 8 August 2024, 8:20 AM

Hi Christopher,
Thanks for your post. Your explanation of NP problems, NP-complete problems, and their complexity is generally clear and accessible. Including examples like the Traveling Salesman Problem (TSP) and the Knapsack Problem helps illustrate the concepts. Good work.
39 words

Permalink Show parent

Re: Week 7
by Muritala Akinyemi Adewale - Thursday, 8 August 2024, 8:26 AM

Your engaging analysis of solution verification versus discovery offers a fresh perspective. The chess game problem example vividly demonstrates the rapid expansion of move possibilities and the ensuing computational difficulties. Your focus on these challenges helps clarify their significance in computational complexity. Thanks for sharing such a thought-provoking and well-articulated overview!
51 words

Permalink Show parent

Re: Week 7
by Liliana Blanco - Thursday, 8 August 2024, 8:36 AM

Your discussion of NP-complete problems was quite clear and comprehensive, in my opinion. Your illustrations effectively highlight the difficulty of these issues. My analysis and yours are very similar in that they highlight the importance of NP-complete problems in computational complexity.
41 words

Permalink Show parent

Re: Week 7
by Jobert Cadiz - Thursday, 8 August 2024, 6:44 AM

Discussion Forum Unit 7

What is an NP-Complete Problem?

An NP-Complete problem is a type of problem that is both in the class of problems known as NP (nondeterministic polynomial time) and is as difficult as the toughest problems in NP.

1. **NP (Nondeterministic Polynomial time):** A problem is in NP if, once you have a potential solution, you can quickly check if it's correct. This checking can be done in polynomial time, which means the time grows at a manageable rate as the problem size increases.
2. **NP-Complete: A problem is NP-Complete if:**

- It is in NP (you can verify solutions quickly).
- Any problem in NP can be converted into this problem in polynomial time.

This means that NP-Complete problems are the most challenging within the NP class. If you could solve one NP-Complete problem efficiently (in polynomial time), you could solve all NP problems efficiently. Conversely, if no efficient solution is possible for NP-Complete problems, then it's likely that no NP problem can be solved efficiently.

Why is an NP-Complete Problem Considered 'Hard'?

According to GeeksforGeeks (2024), NP-Complete problems are considered hard because:

1. No Known Fast Solution: There is no known way to solve NP-Complete problems quickly (in polynomial time). Finding a solution can take a very long time as the problem size grows.
2. Conversion from Any NP Problem: The fact that any problem in NP can be turned into an NP-Complete problem shows that solving an NP-Complete problem is as hard as solving the hardest problems in NP. This complexity makes NP-

Complete problems particularly tough to handle.

Examples of NP-Complete Problems

1. **Traveling Salesman Problem (TSP):**

- Problem Statement: Given a list of cities and the distances between them, find the shortest route that visits each city exactly once and returns to the starting city.
- Why it's NP-Complete: Finding the shortest route is difficult and no fast (polynomial-time) method is known to solve this problem. Checking if a given route is correct is easy, but finding the best one is very hard.

Subset Sum Problem:

- Problem Statement: Given a set of numbers and a target number, decide if there is a subset of numbers that adds up to the target number.
- Why it's NP-Complete: It's easy to check if a given subset adds up to the target, but finding such a subset from a large set is hard. No quick solution is known for all cases.

Thought Process and Methodology

To understand why NP-Complete problems are hard, we can think about:

1. Examining how one may turn another difficulty into a problem is one technique to understand why a problem is difficult. For instance, demonstrating that the TSP can be derived from an already-existing hard problem demonstrates how difficult the TSP is.
2. NP-Complete problems are challenging because, although a solution can be quickly checked, it takes time to identify the optimal solution from a large number of options. It is challenging to solve NP-Complete issues because of this distinction.

To sum up, NP-Complete problems are challenging because they have no known polynomial-time solutions, and they can capture the complexity of all NP problems. This puts them at the center of research on computational theory and optimization.

References

GeeksforGeeks. (2024, May 15). *Introduction to NP-Complete Complexity Classes*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-np-completeness/

536 words

Permalink Show parent

Re: Week 7

by Muritala Akinyemi Adewale - Thursday, 8 August 2024, 8:27 AM

You've clearly captured the essence of the trade-off between verifying and finding solutions. The chess game problem example is particularly effective in showcasing the exponential growth of possible moves and the computational difficulties that arise. Your discussion effectively highlights the importance of these problems in understanding computational complexity. Thanks for the clear and insightful explanation!

55 words

Permalink Show parent

Re: Week 7
by Christopher Mccammon - Thursday, 8 August 2024, 8:51 AM

Hi Jobert
Your explanation of NP-Complete problems is clear and informative. You effectively outline the key concepts of NP and NP-Completeness, providing a solid foundation for understanding why NP-Complete problems are considered particularly challenging. The use of examples like the Traveling Salesman Problem and the Subset Sum Problem helps illustrate these concepts well. good job
55 words

Permalink Show parent

Re: Week 7
by Natalie Tyson - Thursday, 8 August 2024, 7:22 AM

We were given in our reading a breakdown of the different complexity classes in our reading for unit 7. NP stands for nondeterministic polynomial time, which unlike the P class problems that are made up of quicker decision problems, are not easy or quick to solve in both theory and practice.

NP-complete problems have two features that set them apart:

Hard to solve: Computationally it is difficult to solve these problems, if you are able to find an efficient way to solve one of these problems with the right polynomial-time formula, then it should work on all of the NP problems as well.
Polynomial Time Verification: If a solution was given for the problems, we can use a deterministic algorithm in order to verify if it's correct. If you want to check if a possible solution meets the requirements efficiently, it is possible to do so quickly.

Solving an NP - complete problem would be defined as 'hard' because it consumes more memory and time in order to solve the problem. There is no efficient algorithm which is able to solve these types of problems.

Let's look at some examples of NP complete problems.

Subset Sum Problem
Description - This problem involves locating a subset of numbers from a given set which will when added produce a target sum

Example: If you have a set of numbers {2, 12, 4, 5} and a target sum of 11. Determine whether a subset of these numbers sums up to 11. We have {2, 4, 5} which meet this requirement.

Why is it so hard? There are 2^n possible s subsets of a set with n numbers. As the size of the set increases, the number of subsets grows exponentially, making this computationally harder to solve as the set gets larger. There is no polynomial-time solution for this problem.

Traveling Salesman Problem (TSP)
Description - Problem asks the shortest possible route that visits a set of cities only once and returns to the original city.

Example: We have 3 cities: A, B,and C and we want to find the most optimal route. We need to consider all computations and we have 3! = 6 permutations for the routes. As the city number grows, so will the route number grow factorially. Making the optimal route computation intensive as it grows.
Why is it so hard? This is supposed to evaluate for the shortest possible route by evaluating all possible routes. There is no polynomial-time algorithm that can solve the TSP problems, finding the solution quickly in polynomial time would not be possible.

Summary

NP-complete problems can verify whether a solution is the correct one or not in polynomial time. It is easy to check the solutions or if someone were to give us a subset for the sum problem as well as verify the route for the TSP. Finding the solution quickly and efficiently is the real challenge for these problems that quickly consume more resources as they scale in size.

Citation

- GeeksforGeeks. (2023, October 3). P, NP, CoNP, NP hard and NP complete: Complexity Classes. https://www.geeksforgeeks.org/types-of-complexity-classes-p-np-conp-np-hard-and-np-complete/#
- Schaffer, C.A. (2011). A Practical Introduction to Data Structures and Algorithms Analysis (3.1 ed.). Blacksburg, VA: Virginia Tech University, Department of Computer Science. Available at http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

539 words

Permalink Show parent

Re: Week 7
by Jobert Cadiz - Thursday, 8 August 2024, 8:18 AM

Hi Natalie,
Your explanation of NP-complete problems is thorough and covers important aspects. You clearly define NP-complete problems and highlight their characteristics, particularly the difficulty in solving and polynomial-time verification. The inclusion of concrete examples effectively illustrates the concepts and demonstrates why these problems are challenging. Great work!

48 words

Permalink Show parent

Re: Week 7
by Muritala Akinyemi Adewale - Thursday, 8 August 2024, 8:26 AM

Your explanation of the balance between verifying solutions and finding them is spot-on. The chess game problem example effectively highlights the exponential growth in potential moves and the subsequent computational challenges. Your discussion emphasizes the crucial role these problems play in understanding computational complexity. Thanks for providing such a clear and informative overview!

53 words

Permalink Show parent

Re: Week 7
by Liliana Blanco - Thursday, 8 August 2024, 8:37 AM

Your explanation of NP-complete problems is excellent and easy to follow. I like how you described the key features of these problems and provided practical examples like the Subset Sum Problem and the Traveling Salesman Problem. It's cool to see how we both highlighted the challenges in finding efficient solutions while acknowledging the ease of verifying them.

57 words

Permalink Show parent

Re: Week 7
by Christopher Mccammon - Thursday, 8 August 2024, 8:56 AM

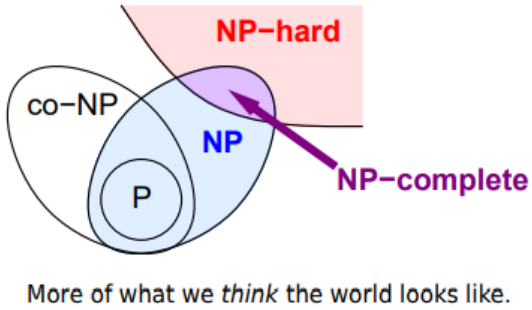
Hi Natalie
Your explanation of NP-Complete problems offers a solid foundation on the topic. You effectively describe what NP-Complete problems are, outlining their characteristics and challenges. The inclusion of well-known examples, like the Subset Sum Problem and the Traveling Salesman Problem (TSP), helps in illustrating why these problems are considered difficult. Good job!

53 words

Permalink Show parent

Re: Week 7

by Chong-Wei Chiu - Thursday, 8 August 2024, 10:23 AM



More of what we *think* the world looks like.

NP (Nondeterministic Polynomial time) complete problems are a kind of NP problem. This means they cannot be solved in polynomial time. According to the graph, we know that NP complete problems are also NP-hard problems, which means they are harder to solve than general NP problems.

I think the reason why NP complete problems are considered to be hard can be attributed to two factors: the diversity of solutions and the lack of a general solution for NP complete problems.

Diversity of solutions: NP complete problems have a vast number of possible solutions, and the number of these solutions usually increases sharply as the input size increases. You need to verify these solutions to determine whether they are the correct answer to the problem you want to solve. If your algorithm is not efficient, this issue can become a disaster. The computational requirements can become extremely large, making the solution set too large to verify, and finding the correct answer becomes impractical.

General solution for NP complete problems: Another issue is that a general solution for these NP complete problems has not yet been found. This means that even though we know these NP-complete problems can be transformed into one another, we still cannot solve these problems in polynomial time.

Reference:

Schaffer, C.A. (2011). A Practical Introduction to Data Structures and Algorithms Analysis (3.1 ed.). Blacksburg, VA: Virginia Tech University, Department of Computer Science. Available at http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

Dasgupta, S., Papadimitriou, C.H., & Vazirani, U.V. (2006). Algorithms. Berkeley, CA: University of California Berkeley, Computer Science Division. Available at http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

258 words

Permalink Show parent