Week 1

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 1

by Romana Riyaz (Instructor) - Thursday, 20 June 2024, 10:25 AM

Discussion Assignment #1
In your own words describe both brute force algorithms and branch and bound algorithms and identify the differences between the two. As part of your description you must discuss the Asymptotic behavior of each and discuss why this is important and how this might factor into the decision to use either algorithm design.

Discussion Assignment #2
Complete the following exercises and post your answers to the discussion forum. In each assignment the questions and answers are provided so that you can self-check your work. What is important is your ability to show your work or line of reasoning. After submitting your assignment you should engage with your peers to both validate your own techniques and perhaps critique the work of others. The objective of this assignment is for all of the members of the class to work collaboratively to develop a solid understanding of the Asymptotic assessment of an algorithm and the ability to solve recurrence relations

Exercise 1
$f(n) = n^6 + 3^n$
$f(n) = 2^n + 12$
$f(n) = 3^n + 2^n$
$f(n) = n^n + n$

(Write down your results first; the solution is given below)

Exercise 2:
For the following Θ complexities write down a tight and a non-tight O bound, and a tight and non-tight Ω bound of your choice, providing they exist.

Θ(1)
Θ(√n)
Θ(n)
Θ(n2)
Θ(n3)

For your discussion post, explain in your own words how you solved the exercise. Include one or two examples to explain your thought process to show what is occurring and how the methodology works. Demonstrate your understanding of how to solve each exercise. Use APA citations and references for any sources used.

Discussion Assignment #2 Answers:

Exercise 1 Answers:
$f(n) = n^6 + 3^n \in \Theta(3^n)$
$2^n + 12 \in \Theta(2^n)$
$3^n + 2^n \in \Theta(3^n)$
$n^n + n \in \Theta(n^n)$

Exercise 2 Answers:
This is a straight-forward application of the definitions above.

1. The tight bounds will be O(1) and Ω(1). A non-tight O-bound would be O(n). Recall that O gives us an upper bound. As n is of larger scale than 1 this is a non-tight bound and we can write it as o(n) as well. But we cannot find a non-tight bound for Ω, as we can't get lower than 1 for these functions. So we'll have to do with the tight bound.

2. The tight bounds will have to be the same as the Θ complexity, so they are O(√n) and Ω(√n) respectively. For non-tight bounds we can have O(n), as n is larger than and so it is an upper bound for √n. As we know this is a non-tight upper bound, we can also write it as o(n). For a lower bound that is not tight, we can simply use Ω(1). As we know that this bound is not tight, we can also write it as ω(1).

3. The tight bounds are O(n) and Ω(n). Two non-tight bounds could be ω(1) and o(n^3). These are in fact pretty bad bounds, as they are far from the original complexities, but they are still valid using our definitions.

4. The tight bounds are O(n2) and Ω(n^2). For non-tight bounds we could again use ω(1) and o(n3) as in our previous example.

5. The tight bounds are O(n^3) and Ω(n^3) respectively. Two non-tight bounds could be ω(n^2) and o(n^4). Although these bounds are not tight, they're better than the ones we gave above.

630 words

Permalink

Re: Week 1

by Moustafa Hazeen - Saturday, 22 June 2024, 3:04 AM

The brute force algorithm, often referred to as an uncomplicated method for problem-solving, involves enumerating all possible ways, paths, and solutions for a given issue. This intuitive and direct method (Soubhikmitra 1998, 2021) is frequently applied in everyday situations, such as finding the shortest route to a destination or efficiently packing items of various sizes (Soubhikmitra 1998, 2021). It is particularly effective for smaller and simpler problems, as it systematically explores all potential outcomes. However, for larger or more complex problems, this approach can be inefficient and its performance heavily depends on the computational power available (Soubhikmitra 1998, 2021).

In contrast, the branch and bound algorithm is designed for discrete, general mathematical, and combinatorial optimization problems (Datta, 2022). This algorithm features a tree-like structure with a root node representing the entire problem and branches leading to possible solutions (Datta, 2022). From the root node, both feasible and infeasible solutions are generated, and decisions are made at each node to determine if it should be part of the solution set (Datta, 2022). This method excels in solving large and complex problems due to its ability to prune branches that do not lead to optimal solutions, thus saving computational resources.

Asymptotic analysis helps in evaluating the performance of these algorithms by calculating their execution time based on input size. For the branch and bound algorithm, this analysis indicates that its performance is influenced by the size of the input; larger inputs typically result in longer execution times (DeepAI, 2020). Conversely, the brute force algorithm is best suited for smaller problems with limited input sizes, while the branch and bound algorithm can efficiently handle larger and more complex problems, albeit with potentially higher computational costs due to its larger input sizes.

Exercise 1:

1. fn=n6+ 3n (3n) , this is because 3n will grow larger than n6 when n = 17.

2. fn= 2n+12 (2n), this is because a constant number 12 cannot be changed and 2n will grow for different values of n.

3. fn= 3n+ 2n (3n), this is because 3n will always grow larger than 2n for every value of n.

4.fn= nn+n(nn) , this is because nn will always grow larger than n for every value of n.

Exercise 2:

1. Tight upper bounds are O(1) and Ω(1). Non-tight bound is o(n) and we cannot find a non-tight lower bound as the function value cannot be lower than 1.

2. Tight upper bounds are O(√n) and Ω(√n). Non-tight bounds are O(√n) and Ω(1).

3. Tight bounds are O(n) and Ω(n). Non-tight bounds are O(n2) and Ω(1).

4. Tight bounds are O(n2) and (n2) . Non-tight bounds are o(n2) and Ω(1).

5. Tight bounds are O(n3) and (n3) . Non-tight bounds are o(n4) and (n2).

References:

Datta, S. (2022, November 11). Branch and Bound Algorithm | Baeldung on Computer Science. Baeldung on Computer Science. https://www.baeldung.com/cs/branch-and-bound

DeepAI. (2020, June 25). Asymptotic Analysis. https://deepai.org/machine-learning-glossary-and-terms/asymptotic-analysis

Soubhikmitra98. (2021, May 4). Brute Force Approach and its pros and cons. GeeksforGeeks. https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

573 words

Permalink Show parent

Re: Week 1
by Benjamin Chang - Sunday, 23 June 2024, 11:30 PM

Hi Moustafa,
It was great to read your initial post this week! It was quite enjoyable because you taught the brute force method in an understandable manner and clearly defined the distinctions between brute force and branch and bound using your own words. Your explanations for assignments 1 and 2 were very easy to follow. I applied the same procedure to the assignments. Great job!

Yours

Benjamin
67 words

Permalink Show parent

Re: Week 1
by Cherkaoui Yassine - Monday, 24 June 2024, 4:43 PM

Moustafa, 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 1
by Akomolafe Ifedayo - Monday, 24 June 2024, 8:42 PM

Hi Moustafa, great work on your post. Your submission was well-detailed as you explained the brute force and branch and bound algorithm. You also described their differences and asymptotic analysis. Furthermore, you provided solutions to

both exercises. Keep it up.
40 words

Permalink Show parent

Re: Week 1
by Anthony Jones - Wednesday, 26 June 2024, 4:23 AM

Hello,

Good job. Can you think of any cases when we cannot improve a brute force algorithm using Branch and bound?
Also, isn't O(√n) a tight bound of Θ(√n)? you said it was also a non-tight bound.

Keep it up! God bless!

Anthony
45 words

Permalink Show parent

Re: Week 1
by Fadi Al Rifai - Wednesday, 26 June 2024, 2:57 PM

Hi Moustafa,
Thank you for your thoughtful contribution to a detailed explanation of the brute force and branch and bound algorithm, I also like your description of their differences and asymptotic analysis.
Keep it up.
35 words

Permalink Show parent

Re: Week 1
by Tousif Shahriar - Wednesday, 26 June 2024, 9:53 PM

Hello Moustafa,
Great post! Your explanations were thorough and made the topic easier to understand. I also liked the fact you mentioned that the branch and bound algorithm may need higher computational power when working with a larger input compared to brute force.
Thanks for sharing.
46 words

Permalink Show parent

Re: Week 1
by Siraajuddeen Adeitan Abdulfattah - Thursday, 27 June 2024, 4:45 AM

Hi Moustafa,

Excellent submission in response to the questions asked. You clearly described and explain Brute force, Branch and bound algorithm as well as asymptotic behavior/analysis. You provide valid points that are self explanatory along with suitable instances for the algorithm discussed. You provided in-text citations and reference(s) for your submission.
51 words

Permalink Show parent

Re: Week 1
by Nour Jamaluddin - Thursday, 27 June 2024, 5:56 AM

Very good job.
Your explanations are appropriate. The examples also are very helpful. I liked your own job.
Thanks for sharing.
21 words

Permalink Show parent

Re: Week 1
by Liliana Blanco - Thursday, 27 June 2024, 10:02 AM

I think your explanation of brute force and branch and bound algorithms was really good. I agree with your point about the brute force method being straightforward but inefficient for complex problems. Your analysis of branch and bound effectively highlights its efficiency in pruning non-optimal solutions. Your analysis in Exercise 2 is well-structured and insightful. My approach was similar to mine, but your detailed reasoning added clarity.
67 words

Permalink Show parent

Re: Week 1
by Anthony Jones - Saturday, 22 June 2024, 6:56 AM

Discussion Assignment #1

Brute force algorithms are straightforward, relatively easy-to-implement methods of solving problems. Often, brute force methods are quite inefficient, though. For instance, we could try to sort an array by repeatedly looking for the smallest element in the array and adding that to a new array. Another example is finding the minimum spanning tree by comparing all the possible permutations of the spanning tree. This involves evaluating all the possible combinations which grow exponentially. So, brute-force methods usually have bad asymptotic times, often O(n^2), O(2^n), and higher. The brute force also doesn't attempt to improve efficiency, the best and worst cases are the same.

Branch and bound is a method of optimizing the brute force method to reduce unnecessary evaluation. The branch and bound method typically determines an initial solution, then performs an optimistic evaluation (less expensive than actually evaluating them) of the other branches to remove the ones that cannot perform as well as the initial/best solution. This way we can eliminate the need to pursue branches unnecessarily. For instance, let's take the spanning tree example: if we find that a partially complete spanning tree that we are evaluating is longer (or will end up longer) than an existing solution, we can stop exploring all spanning trees that stem from our incomplete spanning tree. This cuts resource and computation costs, but unless we have really good ways of estimating the potential of a given permutation, it doesn't usually help a lot (Do Quoc Quan, 2020). Nonetheless, it can reduce the best-case asymptotic time complexity. The worst-case is still the same as the brute force method since it is just a specific attempt at optimizing the brute force algorithm.

Discussion Assignment #2

Exercise #1

f(n) = $n^6 + 3^n$ This is in Big Theta of 3^n since it is the highest order term and essentially eclipses the n^6 term asymptotically, so

f(n) = $2^n + 12$ This is in Big Theta of 2^n since it is the highest order term and we can ignore the constant offset, so

f(n) = $3^n + 2^n$ This is in Big Theta of 3^n since it is the highest order term and has more effect than the 2^n term, so

f(n) = $n^n + n$ This is in Big Theta of n^n since it is the highest order term and rises very quickly, so This has the highest time complexity of the ones in this exercise

To solve equations like this, I find it easiest to just look at the highest-order term. To determine this we should recognize that the higher exponent is the higher order (so C^n exponents are asymptotically higher-order than any n^C exponents)

Exercise #2

Θ(1) is tightly bounded by Ω(1) and O(1) since we can simply apply different coefficients to it in order to bound the specific complexity function. Θ(1) doesn't have a non-tight lower bound, but a non-tight upper would be O(n), though things like O(n^2) and O(2^n) are also non-tight upper bounds

Θ(√n) is tightly bounded by Ω(√n) and O(√n) for similar reasons. Ω(1) and O(n) are non-tight bounds of Θ(√n).

Θ(n) is tightly bounded by Ω(√n) and O(√n) for similar reasons. Ω(√n) and O(n^2) are non-tight bounds of Θ(√n). Ω(1) and O(n^3) are other non-tight bounds, as well.

Θ(n^2) is tightly bounded by Ω(n^2) and O(n^2) and non-tightly bounded by Ω(n) and O(n^3).

Θ(n^3) is tightly bounded by Ω(n^3) and O(n^3) and non-tightly bounded by Ω(n^2) and O(n^4).

It is important to realize that function with Big-Theta complexity is tightly bound by Big-Oh and Big-Omega of the same degree. They are tight-bounds because they have very similar shapes and rise at approximately the same rates (given similar constant coefficients) due to having the same order. Meanwhile, non-tight bounds use higher or lower order terms which means they rise significantly faster or slower and diverge significantly from the initial degree (while still bounding it) as the problem size increases. So, being bounded by a large Big-Oh or small Big-Omega doesn't necessarily characterize how efficient the function is particularly accurately, but it does tell us that it is at least that efficient\inefficient (Shewchuk, 2014b).

References

Amit G. Maru. (2020, October 13). *Traveling Salesman Problem Using Branch and Bound*. YouTube. https://youtu.be/vtEF-IR36oM

Do Quoc Quan. (2020, November 5). *Knapsack 5 - relaxation, branch and bound - discrete optimization*. YouTube. https://youtu.be/Kw5DNm39bmA

Shewchuk, J. (2014a, March 13). *Lecture 19 asymptotic analysis*. YouTube. https://youtu.be/X8Ba--V4sGA

Shewchuk, J. (2014b, March 13). *Lecture 20 algorithm analysis*. YouTube. https://youtu.be/q5BMmfGdWtU

827 words

Permalink Show parent

Re: Week 1

by Moustafa Hazeen - Sunday, 23 June 2024, 12:52 AM

Hello Anthony, your submission is thorough and well-organized, providing clear explanations of brute force and branch and bound algorithms. you demonstrate a solid grasp of asymptotic complexities and effectively apply these concepts to the exercises given. your use of references adds credibility to your explanations

45 words

Permalink Show parent

Re: Week 1

by Benjamin Chang - Sunday, 23 June 2024, 11:31 PM

Hi Anthony,

I appreciate how you explained the differences between brute force and branch and bound by presenting the best and worst case asymptotic complexity for each approach. This form of comparison is quite successful and respected. Additionally, your exercise explanations are simple to understand. It's like having an excellent teacher who delivers the content in a way that even new pupils can understand. Great job!

Yours,

Benjamin

68 words

Permalink Show parent

Re: Week 1
by Cherkaoui Yassine - Monday, 24 June 2024, 4:43 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 1
by Tousif Shahriar - Wednesday, 26 June 2024, 10:04 PM

Hello Anthony,
Great post! Your explanation of brute force algorithms and their inefficiencies was very clear. I particularly liked your example of sorting by repeatedly finding the smallest element and the exponential growth in finding minimum-spanning trees.
Your transition to branch and bound was smooth, and you highlighted the optimization well.
Regarding Exercise #2, your identification of tight and non-tight bounds was spot-on. I agree that recognizing the highest-order term is key to determining Θ notation.
Thanks for sharing!
79 words

Permalink Show parent

Re: Week 1
by Siraajuddeen Adeitan Abdulfattah - Thursday, 27 June 2024, 4:54 AM

Hi Anthony,

Excellent submission in response to the questions asked. You provided a very simple yet explanatory response in describing Brute force and Branch and bound algorithm. Your explanation offers better understanding of the subject and meaning. You also provide simple and explanatory solutions for the exercises.
47 words

Permalink Show parent

Re: Week 1
by Nour Jamaluddin - Thursday, 27 June 2024, 5:59 AM

Dear friend,
The arguments you provided are reasonable. You understand the concepts very well. The examples are clear.
Thanks a lot!
21 words

Permalink Show parent

Re: Week 1
by Chong-Wei Chiu - Thursday, 27 June 2024, 10:53 AM

Hello, Anthony Jones. Thank you for sharing your point of view on this topic. You state the definitions of the brute force algorithm and the branch and bound algorithm and explain their effects in the asymptotic aspect. After reading your post, I learned a lot from it. Thank you!
49 words

Permalink Show parent

Re: Week 1
by Cherkaoui Yassine - Sunday, 23 June 2024, 12:16 AM

Brute Force Algorithms: Brute force algorithms involve exploring all possible solutions to find the optimal one. They are straightforward and simple but are often inefficient. The asymptotic behavior of brute force algorithms is typically very high, often exponential (O(2^n), O(n!)), making them impractical for large inputs. Asymptotic analysis is crucial here because it provides a clear picture of how the algorithm's runtime grows with the input size, helping to identify inefficiencies.

Branch and Bound Algorithms: Branch and bound algorithms improve on brute force by systematically pruning branches of the search space that cannot yield a better solution than the current best. This method significantly reduces the number of solutions to consider. The asymptotic behavior is typically better than brute force, often polynomial or exponential in the worst case but with a much-reduced constant factor. Understanding asymptotic behavior helps in predicting the performance and feasibility of these algorithms for large inputs.

Differences : The primary difference is efficiency. Brute force algorithms exhaustively search the entire solution space, leading to high computational costs. In contrast, branch and bound algorithms use a strategic approach to eliminate large portions of the search space, resulting in faster performance.

Exercise 1

1. f(n) = n^6 + 3n belongs to Θ(n^6). The dominant term is n^6.

2. f(n) = 2^n + 12 belongs to Θ(2^n). The exponential term dominates.

3. f(n) = 3^n + 2^n belongs to Θ(3^n). The term 3^n grows faster.

4. f(n) = n^n + n belongs to Θ(n^n). The term n^n dominates.

Exercise 2

1. Θ(1):

- Tight bounds: O(1), Ω(1)

- Non-tight O-bound: O(n)

- Non-tight Ω-bound: None below 1

2. Θ(√n):

- Tight bounds: O(√n), Ω(√n)

- Non-tight O-bound: O(n)

- Non-tight Ω-bound: Ω(1)

3. Θ(n):

- Tight bounds: O(n), Ω(n)

- Non-tight O-bound: O(n^3)

- Non-tight Ω-bound: Ω(1)

4. Θ(n^2):

- Tight bounds: O(n^2), Ω(n^2)

- Non-tight O-bound: O(n^3)

- Non-tight Ω-bound: Ω(1)

5. Θ(n^3):

- Tight bounds: O(n^3), Ω(n^3)

- Non-tight O-bound: O(n^4)

- Non-tight Ω-bound: Ω(n^2)

328 words

Permalink Show parent

Re: Week 1
by Moustafa Hazeen - Sunday, 23 June 2024, 12:54 AM

Cherkaoui, your submission is excellent and well-organized. You've provided clear and concise explanations of both brute force and branch and bound algorithms, highlighting their efficiencies and inefficiencies with clarity. Your analysis of the asymptotic behaviors in Exercise 1 is spot on, clearly identifying the dominant terms and their implications.

In Exercise 2, your explanations of tight and non-tight bounds for various complexity classes are thorough and well-supported. It's evident that you have a strong grasp of these concepts and can effectively apply them to different scenarios.

Overall, your submission demonstrates a deep understanding of algorithmic complexities and asymptotic analysis. Well done on effectively meeting the objectives of the assignment!

109 words

Permalink Show parent

Re: Week 1
by Romana Riyaz (Instructor) - Sunday, 23 June 2024, 1:51 AM

Hello Yassine,
Thank you for your submission. Your explanation of brute force and branch and bound algorithms is concise and effectively highlights the key differences between them, particularly in terms of efficiency and asymptotic behavior. The analysis of asymptotic complexity is well-done, emphasizing the impracticality of brute force for large inputs and the strategic advantage of branch and bound. Your exercises on identifying the dominant terms and the tight and non-tight bounds for different functions are accurate and demonstrate a clear understanding of the concepts. To further improve, consider providing more detailed reasoning for your answers in Exercise 2 to enhance clarity and understanding. Overall, your work shows a strong grasp of the material. Well done!
Regards,
Romana Riyaz

119 words

Permalink Show parent

Re: Week 1
by Benjamin Chang - Sunday, 23 June 2024, 11:31 PM

Hi Cherkaoui,
I agree with Moustafa's say that the discussion is clear and well-organized, with full answers to all concerns this week. Your assignment not only shows the differences between branch and bound and brute force methods, but it also simplifies exercises 1 and 2 for people inexperienced with the area. You utilized simple language to describe things thoroughly. Good job!

Yours
Benjamin

63 words

Permalink Show parent

Re: Week 1
by Akomolafe Ifedayo - Monday, 24 June 2024, 8:47 PM

Hi Cherkaoui, great work on your submission. I liked how you highlighted your submissions, it made your work engaging to go through. You clearly defined the brute force algorithm and the branch and bound algorithm. You also explained their differences and provided correct solutions to the exercises. Keep it up.

50 words

Permalink Show parent

Re: Week 1

by Anthony Jones - Wednesday, 26 June 2024, 4:33 AM

Hello,

Would you agree that branch and bound is simply a way of making a brute force method more efficient by eliminating branches that will not help provide a solution? Or would you say it is more than that?

Keep it up; God bless!

Anthony

45 words

Permalink Show parent

Re: Week 1

by Fadi Al Rifai - Wednesday, 26 June 2024, 3:14 PM

Hi Yassine,
Good work, Thanks for sharing your explanation about the brute force and branch and bound algorithm, I also like your description of their differences and asymptotic analysis.
Keep it up.

32 words

Permalink Show parent

Re: Week 1

by Siraajuddeen Adeitan Abdulfattah - Thursday, 27 June 2024, 5:00 AM

Hi Cherkaoui,

This is really good submission in response to the questions asked. You provided a thorough description and explanation on both Brute force and Branch and bound algorithm. You also shared the differences between both.

36 words

Permalink Show parent

Re: Week 1

by Nour Jamaluddin - Thursday, 27 June 2024, 6:02 AM

Well done.
You provided a full information and solutions. I liked your way organizing your ideas. I preferred if you used any out sources as requested in the assignment.
Thank you.

31 words

Permalink Show parent

Re: Week 1

by Jerome Bennett - Thursday, 27 June 2024, 10:40 AM

Greetings Cherakaoui,

Your work was concise and well constructed. You eloquently explained brute force and bound algorithms, clearly highlighting their efficiencies and inefficiencies. Your analysis of the asymptotic behaviors in Exercise 1 is excellent, as it accurately identifies the dominant terms and their implications.

In Exercise 2, your explanations of tight and non-tight bounds for various complexity classes are comprehensive and well-supported.
62 words

Permalink Show parent

Re: Week 1

by Benjamin Chang - Sunday, 23 June 2024, 1:15 AM

Discussion Assignment #1

The brute force algorithm is a simple method that evaluates all possible items to find the answer. Typically, you would look up each one from top to bottom until you identified the answer. According to Shaffer (2010), the brute force algorithm checks all possible sets of vertices and measures the execution time of a number of input graphs (p. 598). Branch and bound algorithms, as the name means, resemble a tree with many branches that have upper and lower boundaries. These algorithms test the answer, and if a bound is likely to produce a negative result, they disregard that branch and go to a new branch to find solutions more rapidly. Shaffer (2010) defines this concept concisely, saying that branch and bound algorithms travel the solution tree down a given branch but stop pursuing it if it exceeds the best tour found, quickly switching to another branch (p. 586).

When studying asymptotic behavior, we must consider both space and time complexity. First, the brute force algorithm examines all available things to find a solution, making it applicable to inputs of any size. The permutation difficulty is typically exponential, employing a recursive technique with space complexity of O(1) or O(n) and time complexity of O(n!). Second, the branch and bound examine is an effective method for breaking down issues into segments, much like a tree with upper and lower boundaries. As a result, it may contain multiple sub-branches or even many branches. The space complexity is determined by the number of node depths, whereas the time complexity, comparable to brute force, can be exponential, up to O(n!).

Discussion Assignment #2

Since we have a solution to this assignment, I will explain how to find it quickly. In this section, we know we should focus on which term moves quicker, rather than the constant number.

$f(n) = n^6 + 3^n$
The 3^n has high degree faster than n^6 , so the answer is 3^n

$f(n) = 2^n + 12$

Ignore the constant but focus on high grows term, the answer is 2^n

$f(n) = 3^n + 2^n$
Both has exponential function but 3>2, so the answer is 3^n

$f(n) = n^n + n$

The n is likely n^1 but n^n grows fast than n. so the answer is n^n

For the following Θ complexities write down a tight and a non-tight O bound, and a tight and non-tight Ω bound of your choice, providing they exist.

Θ(1)
Θ(√n)
Θ(n)
Θ(n2)
Θ(n3)

1. In this question, we know the tight bounds provides O(1) and Ω(1). For non tight bounds O(0), we have precise tight bounds 1, so must be tight bounds.

2. We list tight O is O(√n) and tight Ω is Ω (√n) . The non tight bounds O(n), and Ω(1) . n is larger than so they will be non-tight bound.

3. The tight bounds have precise complexity of O(n) and Ω(n).The tight bounds are O(n) and Ω(n). Two non-tight bounds look not good n^2 but still functional.

4. Tight O bound and Omega are O(n^2) and Ω(n^2), for non tight bounds is O(n^3) and Ω(n) .
5. The tight bounds are O(n^3) and Ω(n^3) that increase faster, and non tight bounds are O(n^4) and Ω(n^2) .

Reference

Shaffer, 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

Zindros, D. (n.d.). *A Gentle Introduction to Algorithm Complexity Analysis*. Retrieved June 20, 2024, from https://discrete.gr/complexity/

625 words

Permalink Show parent

Re: Week 1
by Romana Riyaz (Instructor) - Sunday, 23 June 2024, 1:52 AM

Hello Benjamin,
Your discussion of brute force and branch and bound algorithms effectively captures their differences in approach and efficiency. You've correctly identified the exponential nature of brute force algorithms and the strategic pruning of branch and bound, emphasizing their respective time and space complexities. The explanation of asymptotic behavior is clear and demonstrates a solid understanding of how these algorithms scale with input size. In your second assignment, your explanations for identifying the dominant terms and providing tight and non-tight bounds for different complexities are accurate and well-presented. To further enhance your response, consider providing more detailed reasoning for your answers in Exercise 2, especially for the non-tight bounds, to ensure clarity and thorough understanding. Overall, your work demonstrates a strong grasp of the concepts covered. Well done!

Regards,
Romana Riyaz
132 words

Permalink Show parent

Re: Week 1
by Moustafa Hazeen - Sunday, 23 June 2024, 2:05 AM

Hey Benjamin,

You did a great job explaining brute force and branch and bound algorithms. Your solutions for Exercise 1 and Exercise 2 were clear and showed a good understanding of asymptotic notation. Keep up the good work!

38 words

Permalink Show parent

Re: Week 1

by Akomolafe Ifedayo - Monday, 24 June 2024, 8:49 PM

Hi Benjamin, great work on your post. You answered all of the question prompts and your work was clear and straight to the point. You explained the brute force and branch and bound algorithm, and you also provided solutions to the exercises provided. Keep it up.

46 words

Permalink Show parent

Re: Week 1

by Naqaa Alawadhi - Wednesday, 26 June 2024, 10:58 PM

Good job

2 words

Permalink Show parent

Re: Week 1

by Siraajuddeen Adeitan Abdulfattah - Thursday, 27 June 2024, 5:07 AM

Hi Benjamin,

Your explanation on both algorithm essentially shows how they differ inn terms of efficiency and how they approach solving problems. You stated their space and time complexity and offer explanation on asymptotic behavior of both algorithm. Your response to the exercises are really good and accurate, you also provide in-text citations and reference(s) for your submission.

58 words

Permalink Show parent

Re: Week 1

by Fadi Al Rifai - Sunday, 23 June 2024, 4:41 PM

Discussion Assignment #1

Brute Force Algorithms

A brute force algorithm is a straightforward approach to solving a problem by trying all possible solutions and selecting the best one. It is often considered the most intuitive method for problem-solving, especially when no other efficient algorithm is known. Brute force algorithms are exhaustive in nature, exploring all potential solutions without any shortcuts.

Asymptotic Behavior:

- **Time Complexity:** Generally, brute force algorithms have high time complexity because they evaluate all possible solutions. The time complexity is often exponential, such as $O(2^n)$, $O(n!)$, or even $O(n^k)$, depending on the problem.
- **Space Complexity:** The space complexity can also be high if the algorithm requires storing intermediate results or large data structures.

The asymptotic behavior is important because it provides insight into how the algorithm scales with the size of the input. For large inputs, brute force algorithms become impractical due to their high computational cost.

Branch and Bound Algorithms

Branch and bound algorithms are used to solve optimization problems more efficiently than brute force methods. These algorithms systematically divide the problem into smaller subproblems (branching) and use bounding functions to eliminate subproblems that cannot yield a better solution than the current best (pruning).

Asymptotic Behavior:

- **Time Complexity:** The time complexity of branch and bound algorithms is generally better than brute force, though it can still be exponential in the worst case. The efficiency depends on how effectively the bounding function prunes the search space.
- **Space Complexity:** Space complexity varies based on the implementation, but it can be optimized to use less space compared to brute force algorithms by only keeping track of necessary subproblems.

The asymptotic behavior is crucial because it indicates that branch and bound can handle larger input sizes more efficiently than brute force by reducing the number of evaluations needed.

Differences:

1. **Efficiency:** Brute force is generally less efficient than branch and bound. Branch and bound reduce the search space using bounds, whereas brute force evaluates all possibilities.
2. **Application:** Brute force is often used when the problem size is small or no efficient algorithm is known. Branch and bound is used for larger, more complex problems where optimization is necessary.
3. **Complexity:** Brute force has a typically higher time complexity than branch and bound, which aims to reduce complexity by pruning the search space.

Discussion Assignment #2

Exercise 1:

1. $f(n)=n^6+3n$

- The dominant term is n^6, *so* $f(n) \in \Theta(n^6)$.

2. $f(n)=2^n+12$

- The dominant term is 2^n, *so* $f(n) \in \Theta(2^n)$.

3. $f(n)=3^n+2^n$

- The dominant term is 3^n, *so* $f(n) \in \Theta(3^n)$.

4. $f(n)=n^n+n$

- The dominant term is n^n, *so* $f(n) \in \Theta(n^n)$.

Exercise 2:

1. Θ(1)

- **Tight O bound:** O(1)
- **Non-tight O bound:** O(n)
- **Tight Ω bound:**Ω(1)
- **Non-tight Ω bound:** *(None, since we can't go below 1 for Ω bounds)*

2. Θ(√n)

- **Tight O bound:** O(√n)
- **Non-tight O bound:** O(n)
- **Tight Ω bound:** Ω(√n)
- **Non-tight Ω bound:** Ω(1)

3. Θ(n)

- **Tight O bound:** *O(n)*

- **Non-tight O bound:** O(n2)
- **Tight Ω bound:** Ω(n)
- **Non-tight Ω bound:** Ω(1)

4. Θ(n^2)

- **Tight O bound:** $O(n^2)$
- **Non-tight O bound:** O(n3)
- **Tight Ω bound:** $\Omega(n^2)$
- **Non-tight Ω bound:** Ω(n)

5. Θ(n^3)

- **Tight O bound:** O(n3)
- **Non-tight O bound:** O(n4)
- **Tight Ω bound:** Ω(n3)
- **Non-tight Ω bound:** Ω(n2)

Explanation:

To solve these exercises, I identify the dominant term in each function, as this term determines the asymptotic complexity.

For **Exercise 2**, I apply the definitions of O, Ω, and Θ:

- **O (Big-O):** Describes an upper bound. I consider the smallest function that asymptotically dominates the given function.
- **Ω (Omega):** Describes a lower bound. I consider the largest function that is asymptotically dominated by the given function.
- **Θ (Theta):** Describes a tight bound, where the function is both O and Ω of the given complexity.

By identifying tight and non-tight bounds, I demonstrate an understanding of how different bounds can describe the growth rates of functions relative to each other. This approach helps to recognize the efficiency of algorithms and their scalability.

References:

Topic 1: Asymptotic Analysis and Complexity
-Review of Asymptotic Analysis and Complexity - http://discrete.gr/complexity/

-Chapter 3: Algorithm Analysis in A Practical Introduction to Data Structures and Algorithm Analysis by Clifford A. Shaffer. http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

-Chapter 14 Sections 14.1 through 14.2 Analysis Techniques in A Practical Introduction to Data Structures and Algorithm Analysis by Clifford A. Shaffer.

Topic 2: Sequences and Recurrence relations

Solving Recurrences
Erickson, J. (2010). Algorithms. Available under Creative Commons 3.0 at http://jeffe.cs.illinois.edu/teaching/algorithms/notes/99-recurrences.pdf

-Introduction to Recurrence Relations - algo_ch4_recurrences.pdf

-Read the following lecture notes on solving recurrence relations - http://www.cs.duke.edu/~reif/courses/alglectures/skiena.lectures/lecture3.pdf

789 words

Permalink Show parent

Re: Week 1
by Romana Riyaz (Instructor) - Monday, 24 June 2024, 2:03 AM

Hello Rifai,
Thank you for your submission. Your discussion provides a comprehensive comparison between brute force and branch and bound algorithms, clearly explaining their respective asymptotic behaviors and highlighting their differences in terms of efficiency, application, and complexity. The explanation of dominant terms and asymptotic complexity in Exercise 1 is thorough, correctly identifying the dominant term in each function and classifying them in terms of Θ notation. In Exercise 2, your application of O, Ω, and Θ notations effectively illustrates the concepts of tight and non-tight bounds, showing a clear understanding of how these bounds describe the growth rates of functions. Overall, your discussion demonstrates a strong grasp of algorithm efficiency and complexity analysis.
Regards,
Romana Riyaz
117 words

Permalink Show parent

Re: Week 1
by Cherkaoui Yassine - Monday, 24 June 2024, 4:44 PM

Rifai, 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 1
by Naqaa Alawadhi - Wednesday, 26 June 2024, 11:00 PM

Good job

2 words

Permalink Show parent

Re: Week 1
by Jerome Bennett - Thursday, 27 June 2024, 10:43 AM

Greetings Fadi,

Your explanation of both algorithms clearly shows how they differ in efficiency and their approaches to solving problems. You described their space and time complexity and explained how each algorithm behaves over time. Your answers to the exercises are well-done and accurate
44 words

Permalink Show parent

Re: Week 1
by Mejbaul Mubin - Monday, 24 June 2024, 12:40 AM

0 words
Discussion Assignment.docx

Permalink Show parent

Re: Week 1
by Naqaa Alawadhi - Wednesday, 26 June 2024, 11:01 PM

Good job

2 words

Permalink Show parent

Re: Week 1
by Nour Jamaluddin - Monday, 24 June 2024, 3:31 AM

Discussion Assignment #1

First of all, let's simplify each algorithm by its definition.

1- Brute Force Algorithm

Using brute force algorithms, problems are solved by systematically enumerating all potential candidates for the solution and checking each candidate to determine whether it meets the requirements of the problem. This method ensures that the correct solution is found by examining all the possibilities one by one.

According to Javatpoint (2021) "A brute force approach is an approach that finds all the possible solutions to find a satisfactory solution to a given problem. The brute force algorithm tries out all the possibilities till a satisfactory solution is not found" (Para. 1).

Asymptotic Behavior depends on the specific problem, but generally, it's exponential (e.g., O(n^k) where n is the number of possibilities and k is some constant). Meaning that the time it takes to run the algorithm increases as the problem size increases.

For brute force algorithms, the rapid growth of the time complexity makes them infeasible for large inputs, as the computation time can become prohibitively long.

2- Branch and Bound Algorithms

In this algorithm, there are steps that the algorithm follows to solve the problem.

It is an algorithm that uses tree form to systematically explore possible solutions. This is done by dividing the problem into smaller sub-problems, and calculating the upper and lower bounds on these sub-problems to determine whether they can lead to a better solution than the best solution found so far. If the subproblem cannot provide a better solution, it is excluded from the tree and thus discarded.

Asymptotic Behavior varies depending on the specific problem and the effectiveness of the bounding function. here, it's often lower than brute force, making it much faster for large problems.

Branch and bound offer a good balance between finding the optimal solution and being efficient. For problems where brute force becomes impractical, branch and bound becomes the preferred choice.

Discussion Assignment #2

Exercise 1

- f(n) = n6 + 3n

Dominant Term: n^6

Big-O Notation: O(n^6)

f(n) = n6 + 3n ∈ Θ(3n)

- f(n) = 2n + 12

Dominant Term: 2n

Big-O Notation: O(n)

2n + 12 ∈ Θ(2n)

- f(n) = 3n + 2n

Dominant Term: 2^n

Big-O Notation: O(2^n)

3n + 2n ∈ Θ(3n)

- f(n) = nn + n

Dominant Term: n^n

Big-O Notation: O(n^n)

nn + n ∈ Θ(nn)

Exercise 2

Θ(1)

- Tight O Bound: O(1)

Explanation: Since Θ(1) implies the function has a constant growth rate, the tightest upper bound is also constant, O(1).

- Non-Tight O Bound: O(n)

Explanation: O(n) is a valid upper bound for a constant function because n grows faster than a constant, making it a non-tight bound.

- Tight Ω Bound: Ω(1)

Explanation: The tightest lower bound for a constant function is Ω(1) because the function's growth rate is exactly constant.

- Non-Tight Ω Bound: Non-existent.

Explanation: There is no non-tight lower bound less than Ω(1) for a constant function, as 1 is the minimum growth rate.

Θ(√n)

- Tight (O) Bound: O(√n)

Explanation: The tightest upper bound is O(√n) because it matches the function's growth rate.

- Non-Tight (O) Bound: O(n)

Explanation: O(n) is an upper bound for √n, as n grows faster than √n, making it non-tight.

- Tight Ω Bound: Ω(√n)

Explanation: The tightest lower bound is Ω(√n) because it matches the function's growth rate.

- Non-Tight Ω Bound: Ω (1)

Explanation: Ω (1) is a valid lower bound for √n because a constant grows slower than √n, making it non-tight.

Θ (n)

- Tight (O\) Bound: O(n)

Explanation: The tightest upper bound is O(n) because it matches the function's growth rate.

- Non-Tight (O) Bound: O(n^3)

Explanation: O(n^3) is an upper bound for n, as n^3 grows faster than n, making it non-tight.

- Tight Ω Bound: Ω (n)

Explanation: The tightest lower bound is Ω (n) because it matches the function's growth rate.

- Non-Tight Ω Bound: Ω (1)

Explanation: Ω (1) is a valid lower bound for n, as a constant grows slower than n, making it non-tight.

Θ(n^2)

-Tight O Bound: O(n^2)

Explanation:** The tightest upper bound is O(n^2) because it matches the function's growth rate.

- Non-Tight O Bound: O(n^3)

Explanation: O(n^3) is an upper bound for n^2, as n^3 grows faster than n^2, making it non-tight.

- Tight Ω Bound:** Ω(n^2)

Explanation: The tightest lower bound is Ω(n^2) because it matches the function's growth rate.

- Non-Tight Ω Bound: Ω(1)

Explanation: Ω(1) is a valid lower bound for n^2, as a constant grows slower than n^2, making it non-tight.

Θ(n^3)

- Tight O Bound: O(n^3)

Explanation: The tightest upper bound is O(n^3) because it matches the function's growth rate.

- Non-Tight O Bound: O(n^4)

Explanation: O(n^4) is an upper bound for n^3, as n^4 grows faster than n^3, making it non-tight.

- Tight Ω Bound: Ω(n^3)

Explanation: The tightest lower bound is Ω(n^3) because it matches the function's growth rate.

- Non-Tight Ω Bound: Ω(n^2)

Explanation: Ω(n^2) is a valid lower bound for n^3, as n^2 grows slower than n^3, making it non-tight.

References

Javatpoint. 2021. *Brute force approach*. https://www.javatpoint.com/brute-force-approach

872 words

Permalink Show parent

Re: Week 1
by Mejbaul Mubin - Tuesday, 25 June 2024, 11:49 PM

Hi Jamaluddin,

Your discussion posts on both the Brute Force and Branch and Bound algorithms, as well as your analysis of Big-O notations and bounds, are well-articulated and detailed. Your identification of dominant terms and corresponding Big-O notations is correct. Your explanations of tight and non-tight bounds are precise.
49 words

Permalink Show parent

Re: Week 1
by Fadi Al Rifai - Wednesday, 26 June 2024, 3:15 PM

Hi Jamaluddin,
Your post was well-detailed about the brute force and branch and bound algorithm, I also like your description of their differences and asymptotic analysis.
Keep it up.
29 words

Permalink Show parent

Re: Week 1
by Wingsoflord Ngilazi - Wednesday, 26 June 2024, 5:51 PM

Your analysis demonstrates understanding of asymptotic analysis. Well done.
9 words

Permalink Show parent

Re: Week 1
by Jerome Bennett - Thursday, 27 June 2024, 10:42 AM

Greetings Jamaluddin,

Your work was easy to follow as you took the time to show clear transitions in each step. Your identification of dominant terms and corresponding Big-O notations is accurate. Your explanations of tight and non-tight bounds are concise. Very well done.
43 words

Permalink Show parent

Re: Week 1
by Akomolafe Ifedayo - Monday, 24 June 2024, 8:16 PM

In your own words describe both brute force algorithms and branch and bound algorithms and identify the differences between the two. As part of your description, you must discuss the Asymptotic behavior of each and discuss why this is important and how this might factor into the decision to use either algorithm design.

A simple and comprehensive search strategy that systematically explores every option until a problem's answer is discovered is known as a **"Brute Force"** algorithm (GeeksforGeeks, 2024). It is preferred for its generic approach to problem-solving that is used when a given issue is small enough to make a thorough investigation possible. However, it is inefficient for its large-scale issue because of its high temporal complexity.

When you are to work on a combinatorial optimization problem to systematically search for the best solutions, the **Branch and Bound** Algorithm is a method used. It works by dividing the given problem into smaller subproblems and then eliminating certain problems based on bounds on the optimal solution (GeeksforGeeks, 2024). Until the best solution is found, or all problems have been explored, the problem continues.

Differences between the Brute Force and the Branch and Bound Algorithm:

1. The Brute force algorithm is organized and detailed and it investigates every potential solution to an issue, while the branch and bound algorithm efficiently solves combinatorial problems by exploring only promising subproblems (GeeksforGeeks, 2024).

2. The Brute Force algorithm is used when the issue space is small and easily explorable in a short length of time, while the branch and bound algorithm is used for problems with an exponential number of possible solutions.

3. The Brute Force algorithm is not limited to any specific domain of problems while the branch and bound algorithm can handle large instances of problems effectively.

The brute force algorithm is a direct technique in which all the possible solutions to a given problem are enumerated. In day-to-day life, many problems are solved using this algorithm.

For the branch and bound algorithm, it offers a balance between exhaustive search and intelligent pruning, allowing it to solve optimization problems that would be infeasible to solve by brute force (GeeksforGeeks, 2024).

Discussion Assignment #2
Complete the following exercises and post your answers to the discussion forum. In each assignment the questions and answers are provided so that you can self-check your work. What is important is your ability to show your work or line of reasoning. After submitting your assignment you should engage with your peers to both validate your own techniques and perhaps critique the work of others. The objective of this assignment is for all of the members of the class to work collaboratively to develop a solid understanding of the Asymptotic assessment of an algorithm and the ability to solve recurrence relations

Exercise 1
f(n) = n6 + 3n
f(n) = 2n + 12
f(n) = 3n + 2n
f(n) = nn + n

Exercise 1 Solution: Analyzing the asymptotic behavior of the given functions using Big-O notation.

1. f(n)=n6+3n

- The dominant term is n6

- **Solution**: f(n)=O(n6)

2. f(n)=2n+12

- The dominant term is 2n.
- **Solution**: f(n)=O(2n)

3. f(n)=3n+2n

- The dominant term is 3n (since 3n grows faster than 2n).
- **Solution**: f(n)=O(3n)

4. f(n)=nn+n

- The dominant term is nn
- **Solution**: f(n)=O(nn)

References

GeeksforGeeks. (2024). Branch and Bound Algorithm. https://www.geeksforgeeks.org/branch-and-bound-algorithm/

GeeksforGeeks. (2024). Brute Force Approach and its pros and cons. https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

GeeksforGeeks. (2024). Why do we use branch and bound algorithm? https://www.geeksforgeeks.org/why-do-we-use-branch-and-bound-algorithm/

586 words

Permalink Show parent

Re: Week 1

by Manahil Farrukh Siddiqui - Tuesday, 25 June 2024, 3:08 PM

Hello Akomolafe,

Your explanation of brute force and branch and bound algorithms is clear and informative. You correctly outline their asymptotic behavior and the scenarios where each is applicable. The differences are well-defined, emphasizing the efficiency and application scope of each algorithm.

42 words

Permalink Show parent

Re: Week 1

by Mejbaul Mubin - Tuesday, 25 June 2024, 11:56 PM

Hi Akomolafe Ifedayo,

Your explanation of brute force and branch and bound algorithms is clear and informative. You accurately describe their asymptotic behavior and the scenarios where each is applicable. The differences between the two are well-defined, highlighting the efficiency and appropriate use cases for each algorithm.

Your insights into the brute force approach emphasize its thoroughness and generic applicability, which is excellent for small problems. However, your discussion on its inefficiency for larger problems due to high time complexity is particularly important.

In your description of the branch and bound algorithm, you effectively convey how it systematically narrows down the solution space, making it suitable for larger combinatorial problems. This balance between exhaustive search and intelligent pruning is a key advantage, as you've noted.

Overall, your post provides a solid understanding of both algorithms, their asymptotic behaviors, and the practical implications of choosing one over the other. Great job!

150 words

Permalink Show parent

Re: Week 1

by Anthony Jones - Wednesday, 26 June 2024, 4:41 AM

Hello,

Good job! Can you explain why Branch and Bound is just as bad as brute force in the worst case? Also, what might affect how well the branch and bound algorithm prunes the tree?

God bless!

Anthony

38 words

Permalink Show parent

Re: Week 1

by Muritala Akinyemi Adewale - Monday, 24 June 2024, 8:57 PM

Discussion Assignment #1: Brute Force vs. Branch and Bound Algorithms

Brute Force Algorithms:

A brute force algorithm is a straightforward approach that systematically evaluates all possible solutions to a problem. It's like trying every single key on a key ring until you find the one that unlocks the door. These algorithms are simple to design and implement, but their efficiency suffers as the problem size increases.

Asymptotic Behavior:

The runtime of brute force algorithms often grows exponentially (Θ(n^k)) with the input size (n), where k is a constant. This means the time required to solve the problem balloons rapidly as the number of possibilities increases.

Decision Factor:

Brute force is a good choice for small problems or problems with a limited number of possibilities. However, for larger problems, the exponential growth becomes impractical, and alternative algorithms are preferred.

Branch and Bound Algorithms:

Branch and bound algorithms are a more sophisticated approach that systematically explores the solution space, but strategically prunes out branches that are unlikely to contain the optimal solution. It's like having a key with hints etched on it, helping you eliminate incorrect keys faster. This method can significantly improve efficiency for optimization problems.

Asymptotic Behavior:

Branch and bound algorithms typically have a polynomial runtime (Θ(n^d)), where d is a constant. This means the time complexity grows at a much slower rate compared to brute force, making them more suitable for larger problems.

Decision Factor:

Branch and bound algorithms are ideal for optimization problems where finding the best solution is crucial. Their ability to prune out irrelevant branches makes them a powerful tool for dealing with large and complex problem spaces.

Key Differences:

- **Exploration:** Brute force explores all possibilities, while branch and bound strategically explores promising ones.
- **Efficiency:** Brute force suffers from exponential growth while branching and bound offer polynomial efficiency.
- **Suitability:** Brute force is good for small problems, while branch and bound are better for large, complex problems, especially optimization problems.

Reference:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press. (This is a common reference for algorithms courses and provides a detailed explanation of brute force and branch and bound algorithms)

Word Count: 365 words

369 words

Permalink Show parent

Re: Week 1

by Romana Riyaz (Instructor) - Tuesday, 25 June 2024, 2:34 AM

Hello Adewale,
Thank you for your submission. You have done all the aspects of the assignment. Your discussion offers a clear and concise comparison between brute force and branch and bound algorithms, highlighting their fundamental differences in approach and efficiency. The analogy of trying every key versus using hints to eliminate keys effectively illustrates the contrast between these methods. The explanation of asymptotic behavior for both algorithm types is accurate, emphasizing the exponential growth of brute force and the more manageable polynomial growth of branch and bound. The decision factors you outline provide practical guidelines for choosing the appropriate algorithm based on problem size and complexity. Overall, your discussion effectively communicates the key points and differences between brute force and branch and bound algorithms, making it easy to understand their respective strengths and weaknesses.

Regards,
Romana

136 words

Permalink Show parent

Re: Week 1

by Wingsoflord Ngilazi - Wednesday, 26 June 2024, 5:53 PM

Your explanation is clear, and your analysis demonstrates a good understanding of asymptotic analysis. Well done.

16 words

Permalink Show parent

Re: Week 1

by Tyler Huffman - Thursday, 27 June 2024, 8:27 AM

Hello, your post was very effective in covering the material! Firstly, I enjoyed this: "It's like trying every single key on a key ring until you find the one that unlocks the door". That is an accurate, speedy description. you covered all aspects of the prompt well. You post is organized and includes important information such as decision factors for using said design pattern; surely not everyone would remember to include this as a separate header but it is important enough for such. Well done and keep it up going forward.

91 words

Permalink Show parent

Re: Week 1

by Prince Ansah Owusu - Tuesday, 25 June 2024, 3:11 AM

Brute Force Algorithms

Definition:

Brute Force Algorithms are straightforward methods that solve problems by systematically evaluating all possible solutions. They are simple to implement but can be inefficient for large input sizes due to their high computational overhead.

Asymptotic Behavior:

- **Time Complexity:** Brute Force Algorithms often have exponential time complexity, represented as O(2^n) or worse, depending on the problem structure. This means the time required increases rapidly with the input size, making them impractical for large-scale problems.
- **Space Complexity:** Similarly, their space complexity can be significant, especially when storing all possible solutions or states.

Importance and Decision Factors:

- **Decision Making:** The decision to use a Brute Force Algorithm hinges on the problem size and complexity. It's suitable for small datasets where simplicity and correctness outweigh efficiency concerns.
- **Asymptotic Analysis:** Understanding its exponential time complexity helps in evaluating trade-offs. It informs whether optimizing the algorithm or seeking alternative approaches, such as dynamic programming or heuristic methods, might be necessary.

Branch and Bound Algorithms

Definition:

Branch and Bound Algorithms are an enhancement of backtracking algorithms that systematically explore the search space of a problem, pruning branches that cannot lead to a better solution than the current best known.

Asymptotic Behavior:

- **Time Complexity:** Branch and Bound Algorithms typically have a better time complexity compared to Brute Force due to their ability to prune the search space effectively. They often exhibit O(b^d) complexity, where b is the branching factor and d is the depth of the solution.
- **Space Complexity:** Their space complexity can vary but is generally more manageable than Brute Force, as they only keep promising solutions in memory.

Importance and Decision Factors:

- **Decision Making:** Branch and Bound Algorithms are chosen when the problem has a feasible solution space that can be systematically explored and pruned. They are effective for optimization problems where the goal is to find the best solution within given constraints.

- **Asymptotic Analysis:** Understanding their reduced time complexity compared to Brute Force informs decision-making on algorithm selection. It guides whether the problem size allows for the use of a more sophisticated algorithm that balances efficiency and accuracy.

Discussion #2

Exercise 1 Answers

1. **f(n) = n^6 + 3n**

- Asymptotic Behavior: Θ(n^6)
- Justification: The dominant term is n^6, indicating that as n grows, the time complexity is dominated by the sixth power of n.

f(n) = 2n + 12
- Asymptotic Behavior: Θ(n)
- Justification: The dominant term is 2n, indicating linear growth as n increases.

f(n) = 3n + 2n
- Asymptotic Behavior: Θ(n)
- Justification: The dominant term is 3n (combining like terms), showing linear growth with n.

f(n) = n^n + n
ü Asymptotic Behavior: Θ(n^n)

ü Justification: The dominant term is n^n, indicating exponential growth as n increases.

Exercise 2 Answers

1. **Θ(1)**

ü Tight Bound (O): O(1)

ü Non-Tight Bound (O): O(n) (because O(1) is also O(n), but not a tight bound)

ü Tight Bound (Ω): Ω(1)

ü Non-Tight Bound (Ω): Ω(1) (cannot find a non-tight bound lower than 1)

2. **Θ(√n)**

- Tight Bound (O): O(√n)
- Non-Tight Bound (O): O(n) (since √n ≤ n, but not tight)
- Tight Bound (Ω): Ω(√n)
- Non-Tight Bound (Ω): Ω(1) (since √n ≥ 1, but not tight)

Θ(n)
- Tight Bound (O): O(n)
- Non-Tight Bound (O): O(n^2) (since n ≤ n^2, but not tight)
- Tight Bound (Ω): Ω(n)
- Non-Tight Bound (Ω): Ω(1) (since n ≥ 1, but not tight)

Θ(n^2)
- Tight Bound (O): O(n^2)
- Non-Tight Bound (O): O(n^3) (since n^2 ≤ n^3, but not tight)
- Tight Bound (Ω): Ω(n^2)
- Non-Tight Bound (Ω): Ω(n) (since n^2 ≥ n, but not tight)

Θ(n^3)
- Tight Bound (O): O(n^3)
- Non-Tight Bound (O): O(n^4) (since n^3 ≤ n^4, but not tight)
- Tight Bound (Ω): Ω(n^3)
- Non-Tight Bound (Ω): Ω(n^2) (since n^3 ≥ n^2, but not tight)

Explanation

For Exercise 1, each function's asymptotic behaviour was determined by identifying the dominant term in the expression. This dominant term dictates the growth rate of the function as n increases.

For Exercise 2, the concepts of tight and non-tight bounds for O and Ω were applied to various Θ complexities. Tight bounds match the Θ complexity, while non-tight bounds are either looser upper or lower bounds that still hold but are not as precise.

These exercises help in understanding how to assess the asymptotic performance of algorithms, which is crucial for making informed decisions about algorithm design and selection based on the problem's requirements and constraints.

References:

- Clifford A. Shaffer's book "A Practical Introduction to Data Structures and Algorithm Analysis"
- Jeff Erickson's notes on Algorithms. http://jeffe.cs.illinois.edu/teaching/algorithms/notes/99-recurrences.pdf

747 words

Permalink Show parent

Re: Week 1

by Manahil Farrukh Siddiqui - Tuesday, 25 June 2024, 3:06 PM

Hello Ansah,

Your definitions and explanations of brute force and branch and bound algorithms are clear and concise. The discussion on asymptotic behavior and decision factors is well-articulated, highlighting key differences in efficiency.

33 words

Permalink Show parent

Re: Week 1

by Manahil Farrukh Siddiqui - Tuesday, 25 June 2024, 3:47 AM

Assignment #1

Branch and bound and brute force algorithms are two different approaches to problem-solving, each having unique features and uses.

Brute force algorithms are simple and thorough techniques. They methodically develop every potential solution and assess each one to determine the best (Algorithm Design Paradigms - Overview, n.d.). These algorithms are simple to comprehend and use because they have no optimization strategies. However, the main disadvantage of brute-force methods is that they become inefficient when the problem size increases; for most methods, the time complexity increases quickly as the quantity of the input increases (GeeksforGeeks, 2021). For instance, a brute force method would necessitate examining every configuration of the n vertices in a graph to solve the shortest path issue (Algorithm Design Paradigms - Overview, n.d.). This would result in an O(n!) time complexity, which is unfeasible for large graphs.

Branches and bound algorithms, conversely, are more advanced methods that methodically search the search space by removing branches that cannot produce better answers than those already discovered. These algorithms drastically shrink the search area and quicken the convergence to the best answer by eliminating less promising branches (GeeksforGeeks, 2024). When solving optimization problems, branch and bound algorithms are helpful since they help identify the optimum solution from various options. To avoid evaluating every conceivable path, a branch and bound algorithm, for example, can concentrate on routes that show promise and exclude those unlikely to lead to the best answer in the travelling salesperson issue (A Gentle Introduction to Algorithm Complexity Analysis, n.d.).

When choosing an algorithm design, the method's temporal complexity, or efficiency, is very important. Brute force techniques are inappropriate for significant problems since they typically have exponential time complexity, meaning their runtime proliferates with problem size. On the other hand, branch and bind algorithms generally have better worst-case time

complexity; in certain situations, this complexity is even sublinear or polynomial. Because branch and bound algorithms scale better and converge to optimal solutions faster, they are typically more appropriate in scenarios involving big problems or the need for quick and efficient solutions.

Assignment #2

Exercise 1:

1. **Function 1:** There are two terms: ene^nen and n2n^2n2. The polynomial term n2n^2n2 will be much outnumbered by the exponential term ene^nen when nnn is relatively high. As a result, ene^nen is the dominant term. Let us simplify: f(n)∼enf(n) \sim e^nf(n)∼en. The behaviour asymptotically is O(en)O(e^n)O(en).

2. **Function 2:** Here, n3n^3n3 and 555, a constant term, are present. The word n3n^3n3 will become more common as nnn expands. Let's simplify this: f(n)∼n3f(n) = about n^3f(n)∼n3. The behavior asymptotically is O(n3)O(n^3)O(n3).

3. **Function 3:** There are two exponential terms in this function: 2n2^n2n and 3n3^n3n. The phrase with the larger base, 3n3^n3n, will gain significance as nnn rises. Let's simplify: f(n)∼3nf(n) \sim 3^nf(n)∼3n. The behavior asymptotically is O(3n)O(3^n)O(3n).

4. **Function 4:** We have nlognn\log nnlogn and n2n^2n2. The expression n2n^2n2 will surpass nlognn\log nnlogn when nnn grows greater. Consequently, n2n^2n2 is the major term. f(n)∼n2f(n) \sim n^2f(n)∼n2, simplifying. The behavior asymptotically is O(n2)O(n^2)O(n2).

1. f(n)=n2+en is O(en)O(e^n)O(en).
2. f(n)=n3+5f(n) = n^3 + 5f(n)=n3+5 is O(n3)O(n^3)O(n3).
3. f(n)=2n+3nf(n) = 2^n + 3^nf(n)=2n+3n is O(3n)O(3^n)O(3n).
4. f(n)=n2+nlognf(n) = n^2 + n\log nf(n)=n2+nlogn is O(n2)O(n^2)O(n2).

Exercise 2:

a. Θ(1):

- Tight O bound: O(1)O(1)O(1)
- Non-tight O bound: O(n)O(n)O(n)
- Tight Ω bound: Ω(1)Ω(1)Ω(1)
- Non-tight Ω bound: Ω(1)Ω(1)Ω(1)

Θ(1) represents constant time complexity. The tight O and Ω bounds are constant, as any constant function is both an upper and lower bound for itself. The non-tight O bound O(n)O(n)O(n) and non-tight Ω bound Ω(1)Ω(1)Ω(1) are valid but less precise.

b. Θ(√n):

- Tight O bound: O(n)O(\sqrt{n})O(n)
- Non-tight O bound: O(n)O(n)O(n)
- Tight Ω bound: Ω(n)Ω(\sqrt{n})Ω(n)
- Non-tight Ω bound: Ω(1)Ω(1)Ω(1)

Θ(√n) indicates a square root growth rate. The tight bounds reflect this directly, while the non-tight bounds provide alternative perspectives on the growth rate.

c. Θ(n):

- Tight O bound: O(n)O(n)O(n)
- Non-tight O bound: O(n2)O(n^2)O(n2)
- Tight Ω bound: Ω(n)Ω(n)Ω(n)
- Non-tight Ω bound: Ω(1)Ω(1)Ω(1)

Θ(n) represents linear time complexity. The tight bounds reflect this linear growth rate directly, while the non-tight bounds offer other perspectives.

d. Θ(n^2):

- Tight O bound: O(n2)O(n^2)O(n2)
- Non-tight O bound: O(n3)O(n^3)O(n3)
- Tight Ω bound: Ω(n2)Ω(n^2)Ω(n2)
- Non-tight Ω bound: Ω(n)Ω(n)Ω(n)

Θ(n^2) indicates quadratic time complexity. The tight bounds capture this quadratic growth rate directly, while non-tight bounds provide different growth rates.

e. Θ(n^3):

- Tight O bound: O(n3)O(n^3)O(n3)
- Non-tight O bound: O(n4)O(n^4)O(n4)
- Tight Ω bound: Ω(n3)Ω(n^3)Ω(n3)
- Non-tight Ω bound: Ω(n2)Ω(n^2)Ω(n2)

Θ(n^3) represents cubic time complexity. The tight bounds reflect this cubic growth rate directly, while the non-tight bounds provide alternative growth rates.

In Exercise 2, the Big O, Omega, and Theta notation qualities establish tight and non-tight bounds. Appropriate boundaries can be chosen by comparing each function to well-known functions with comparable growth rates, such as n, n2, and n3. For example, both the tight O(1) and tight Ω(1) constraints hold for the constant function Θ(1). Non-tight bounds, such as Ω(1) and $O(n)$ O(n), represent varying growth rates in contrast to the constant function.

References

A gentle introduction to algorithm complexity analysis. (n.d.). http://discrete.gr/complexity/

Algorithm Design Paradigms - Overview. (n.d.). https://cgi.csc.liv.ac.uk/~ped/teachadmin/algor/algor.html

GeeksforGeeks. (2024, March 19). Why do we use branch and bound algorithm? GeeksforGeeks. https://www.geeksforgeeks.org/why-do-we-use-branch-and-bound-algorithm/

846 words

Permalink Show parent

Re: Week 1

by Mejbaul Mubin - Tuesday, 25 June 2024, 8:03 PM

Hi Manahil,
You provide a comprehensive overview of both brute force and branch and bound algorithms, highlighting their unique features and appropriate use cases. Your post effectively explains complex concepts in an accessible manner. With these enhancements, it will be even more informative and engaging for readers.

47 words

Re: Week 1

by Mahmud Hossain Sushmoy - Tuesday, 25 June 2024, 1:42 PM

Discussion Assignment #1

Brute Force Algorithms:

Brute force algorithms are straightforward problem-solving techniques that systematically explore all possible solutions to find the best one. They are characterized by their simplicity and ease of implementation, making them accessible for many problems. These algorithms guarantee completeness, ensuring that a solution will be found if it exists. However, their major drawback lies in their inefficiency, particularly when dealing with large datasets or complex problems. The time complexity of brute force algorithms is often high, typically O(2^n) or worse for many problems, which means their performance degrades rapidly as the input size increases (University of the People, n.d.).

Branch and Bound Algorithms:

Branch and Bound algorithms offer a more sophisticated approach to solving combinatorial and discrete optimization problems. These algorithms employ a divide-and-conquer strategy, systematically exploring the solution space while using bounding functions to eliminate large portions that cannot contain the optimal solution. This pruning technique significantly improves efficiency compared to brute force methods. Branch and bound algorithms are known for their ability to find optimal solutions, making them particularly valuable in scenarios where the best possible outcome is required. While they can be more complex to implement than brute force approaches, their improved efficiency often justifies the additional complexity, especially for larger problem instances. The worst-case time complexity can vary, but is often better than brute force, sometimes reaching O(n!) in the worst case (University of the People, n.d.).

The asymptotic behavior is important because it allows us to predict how an algorithm will perform with large inputs. This information is critical when deciding which algorithm to use for a given problem, especially when dealing with large datasets or time-sensitive applications.

Discussion Assignment #2

Exercise 1:

$f(n) = n^6 + 3^n \in \Theta(3^n)$

$f(n) = 2^n + 12 \in \Theta(2^n)$

$f(n) = 3^n + 2^n \in \Theta(3^n)$

$f(n) = n^n + n \in \Theta(n^n)$

Explanation:

To determine the asymptotic complexity, we identify the dominant term - the one that grows fastest as n increases.

Example:

$f(n) = n^6 + 3^n$

As n grows, 3^n grows much faster than n^6. This means 3^n dominates, so the complexity is $\Theta(3^n)$.

Exercise 2:

For $\Theta(1)$:

- Tight bounds: $O(1)$, $\Omega(1)$

- Non-tight bounds: $O(n)$ or $o(n)$, $\Omega(1)$ (there's no non-tight lower bound)

For $\Theta(\sqrt{n})$:

- Tight bounds: $O(\sqrt{n})$, $\Omega(\sqrt{n})$

- Non-tight bounds: $O(n)$ or $o(n)$, $\Omega(1)$ or $\omega(1)$

For Θ(n):

- Tight bounds: O(n), Ω(n)

- Non-tight bounds: O(n^2) or o(n^2), Ω(√n) or ω(√n)

For Θ(n^2):

- Tight bounds: O(n^2), Ω(n^2)

- Non-tight bounds: O(n^3) or o(n^3), Ω(n) or ω(n)

For Θ(n^3):

- Tight bounds: O(n^3), Ω(n^3)

- Non-tight bounds: O(n^4) or o(n^4), Ω(n^2) or ω(n^2)

Explanation:

To find tight and non-tight bounds, we use the definitions of Big O, Big Omega, and Theta notations. A tight bound is the same as the Θ complexity. For non-tight bounds, we choose functions that grow faster (for O) or slower (for Ω) than the given complexity.

Example: For Θ(n), the tight bounds are O(n) and Ω(n). For a non-tight upper bound, we can use O(n^2) because n^2 grows faster than n. For a non-tight lower bound, we can use Ω(√n) because √n grows slower than n.

References

University of the People. (n.d.). *UNIT 1: Algorithm Strategies*. Retrieved from https://my.uopeople.edu/pluginfile.php/1861890/mod_book/chapter/512223/chapter03-algointro.pdf

University of the People. (n.d.). *UNIT 1: Optional Video Lectures*. Retrieved from https://my.uopeople.edu/mod/folder/view.php?id=423550

554 words

Permalink Show parent

Re: Week 1

by Manahil Farrukh Siddiqui - Tuesday, 25 June 2024, 2:56 PM

Hello Mahmud,

Your answer of brute force and branch and bound algorithms is clear and informative. The discussion of time complexity and efficiency for each algorithm type is well-presented. I really enjoyed reading your explanation of asymptotic behavior in relation to algorithm performance. Your examples and explanations in the exercises are accurate and demonstrate a solid understanding of Big O, Big Omega, and Theta notations.

65 words

Permalink Show parent

Re: Week 1

by Romana Riyaz (Instructor) - Wednesday, 26 June 2024, 4:32 AM

Hello Karim,
Thank you so much for the detailed submission. Your discussion of brute force and branch and bound algorithms is well-structured and informative. You accurately describe the simplicity and completeness of brute force algorithms while clearly highlighting their inefficiency due to high time complexity. The comparison with branch and bound algorithms is effectively done, emphasizing their use of pruning to improve efficiency, which is particularly beneficial for larger, more complex problems. Your explanation of asymptotic behavior and its importance in predicting algorithm performance with

large inputs is crucial for understanding the practical applications of these algorithms. In your second assignment, the identification of dominant terms and the explanation of tight and non-tight bounds are clear and correct. Your examples illustrate the concepts effectively, demonstrating a solid grasp of algorithmic complexity and efficiency.
Regards,
Romana
135 words

Permalink Show parent

Re: Week 1
by Loubna Hussien - Wednesday, 26 June 2024, 1:33 PM

Your description of brute force algorithms is clear, emphasizing their simplicity and the guarantee of finding a solution, while also correctly noting their inefficiency for large datasets with typical time complexities like O(2^n). The explanation of Branch and Bound algorithms effectively highlights their optimization through pruning and the divide-and-conquer strategy, noting their improved efficiency and practical value for finding optimal solutions. Your asymptotic analysis exercises are mostly accurate, with correct identification of dominant terms and appropriate bounds, though it's important to ensure the non-tight bounds clearly represent significantly different growth rates from the tight bounds.
95 words

Permalink Show parent

Re: Week 1
by Wingsoflord Ngilazi - Wednesday, 26 June 2024, 12:53 AM

Discussion Assignment #1
In your own words describe both brute force algorithms and branch and bound algorithms and identify the differences between the two. As part of your description you must discuss the Asymptotic behavior of each and discuss why this is important and how this might factor into the decision to use either algorithm design.

Answer:

Brute Force Algorithms

They refer to a sequential search algorithm examines each element in a given set one by one. While easy to implement, sequential searches become inefficient for long lists, as each element might need to be checked to find the desired item.

I think its asymptotic behavior is linear, O (n). The time complexity depends on the length on the sequence.

Branch and Bound Algorithms

Branch and Bound Algorithms are an advanced form of the backtracking search algorithm. Without a cost criterion, the algorithm explores a spanning tree of the solution space using a breadth-first approach. This means it uses a queue to process nodes in a first-in-first-out (FIFO) order. Since they are based on a queue, I think that the time complexity depends on the queue. As such, I believe that branch and bound algorithms have a linear asymptomatic behaviour.

Discussion Assignment #2
Complete the following exercises and post your answers to the discussion forum. In each assignment the questions and answers are provided so that you can self-check your work. What is important is your ability to show your work or line of reasoning. After submitting your assignment you should engage with your peers to both validate your own techniques and perhaps critique the work of others. The objective of this assignment is for all of the members of the class to work collaboratively to develop a solid understanding of the Asymptotic assessment of an algorithm and the ability to solve recurrence relations

Exercise 1
f(n) = $n^6 + 3^n$ = Here I discarded the smaller value and took the bigger value which is n^6
f(n) = $2^n + 12$ = Here I discarded 12 the constant and remained with 2^n
f(n) = $3^n + 2^n$ =Here I dropped the smaller value, 2^n and went with 3^n a bigger value.

f(n) = n^n + n =Here I dropped n and chose a value that can continuously grow, n^n .

a**Exercise 2:**
For the following Θ complexities write down a tight and a non-tight O bound, and a tight and non-tight Ω bound of your choice, providing they exist.

Θ(1)

Tight O bound is O(1)and non-tight O bound is o(1). Tight Ω is still Θ(1) because we don't have anything below it and non-tight Ω bound is ω (nlogn).

Θ(√n)

Tight O bound is O(√n) and non-tight O bound is o(n). Tight Ω is still Ω (√n) and non-tight Ω bound is ω (1).

Θ(n)

Tight O bound is O (n) and non-tight O bound is o(n^2). Tight Ω is still Ω (n) and non-tight Ω bound is ω (√n) .

Θ(n^2)

Tight O bound is O (n^2) and non-tight O bound is o(n^3). Tight Ω is still Ω (n^2) and non-tight Ω bound is ω (n) .

Θ(n^3)

Tight O bound is O (n^3) and non-tight O bound is o(n^4). Tight Ω is still Ω (n^3) and non-tight Ω bound is ω (n^2) .

For a tight bound (both lower and upper) I simply used the same value. However, for non-tight upper bound, I went a step higher than the given and I did the opposite for a lower upper bound (Zindros, n.d.).

References

Zindros, D. (n.d.). A Gentle Introduction to Algorithm Complexity Analysis. Retrieved from https://dionyziz.com/complexity/

636 words

Permalink Show parent

Re: Week 1
by Loubna Hussien - Wednesday, 26 June 2024, 1:32 PM

Your explanation of brute force algorithms is concise and effectively conveys their inefficiency for large datasets due to linear time complexity, O(n). However, your assessment of Branch and Bound algorithms would benefit from a more detailed explanation of their cost-based pruning and search optimization, which can sometimes improve average-case performance but not worst-case complexity. Your asymptotic analysis exercises are mostly accurate, but the interpretation of non-tight bounds needs refinement, as non-tight bounds should generally represent a significantly different growth rate compared to the tight bounds.

85 words

Permalink Show parent

Re: Week 1

by Tousif Shahriar - Wednesday, 26 June 2024, 10:08 PM

Hello Wingsoflord,
Great job on your post! Your descriptions of brute force and branch and bound algorithms were clear and concise.
For the asymptotic behaviour, brute force algorithms often have time complexities worse than linear ones. Branch and bound can also have varying complexity, often aiming to reduce worst-case scenarios but it's not always linear.
Thanks for sharing!
58 words

Permalink Show parent

Re: Week 1

by Tyler Huffman - Wednesday, 26 June 2024, 1:27 AM

In your own words describe both brute force algorithms and branch and bound algorithms and identify the differences between the two. As part of your description you must discuss the Asymptotic behavior of each and discuss why this is important and how this might factor into the decision to use either algorithm design.

A brute force algorithm is an algorithm that relies on sheer computing power over advanced techniques. This often comes in the form of simply trying every possibility. A common example of this is a linear search. Each element in an array is checked until either the element is found or the end of the data set is reached and it can be concluded that the list did not contain the element. This is simple and intuitive but we can instantly deduce the weaknesses/limitations of such an algorithm. This may be ideal for small data sets but as the input size grows, of course so too does the number of elements needing inspection. In a set of 1,000,000 elements, the target may be in the final index and therefore 999,999 inspections are done before it is found. This algorithm leads to an average and worst case complexity of O(n).

Branch and Bound, on the other hand, is a method that associates costs with solutions and finds one or more of those optimal solutions based on those costs. Geeksforgeeks describes the technique as dividing the problem into branches, and then eliminating certain branches based on bounds on the optimal solution. Additionally, B&B algorithms use a breadth first search and a queue as a temporary structure to assist with the processing. Nieselt mentions an important aspect regarding the complexity of B&B algorithms; “Branch and bound does not decrease the complexity of a problem, it is simply a reasonable way to organize a search for an optimal solution” (2008, April 21). While B&B can improve the average case, the worst cases remain unchanged.

Exercise 1:

Here we must simply drop all constants/factors other than the largest growing term:

$f(n) = n^6 + 3^n >>> f(n) = 3^n$
$f(n) = 2^n + 12 >>> f(n) = 2^n$
$f(n) = 3^n + 2^n >>> f(n) = 3^n$
$f(n) = n^n + n >>> f(n) = n^n$

Exercise 2:

For the following Θ complexities write down a tight and a non-tight O bound, and a tight and non-tight Ω bound of your choice, providing they exist.

Θ(1) – The tight bounds here will be O(1) and Ω(1) while the non-tight bounds could be $O(n^2)$. There will be no lower bound for Omega as it cannot go lower than constant time.

Θ(√n) – The tight bounds here will again need to match the Theta complexity giving us O(√n) and Ω(√n).

Θ(n) – tight bounds are O(n) and Ω(n). Non-tight could be $O(n^2)$ and Ω(1)

$\Theta(n^2)$ – tight bounds are $O(n^2)$ and $\Omega(n^2)$. Non-tight could be $O(n^3)$ and $\Omega(1)$

$\Theta(n^3)$ – tight bound are $O(n^3)$ and $\Omega(n^3)$. Non-tight could be $O(n!)$ and $\Omega(n^2)$

References

Geeksforgeeks. (2024, February 22). Branch and Bound Algorithm. https://www.geeksforgeeks.org/branch-and-bound-algorithm/

Nieselt, K. (2008, April 21). Algorithm Techniques. https://my.uopeople.edu/pluginfile.php/1861890/mod_book/chapter/512223/chapter03-algointro.pdf

511 words

Permalink Show parent

Re: Week 1

by Loubna Hussien - Wednesday, 26 June 2024, 1:31 PM

Your explanation of brute force algorithms effectively highlights their straightforward nature and the resulting computational inefficiencies for large datasets, emphasizing the linear complexity of O(n). In contrast, your description of Branch and Bound algorithms clearly outlines their cost association and search optimization approach, noting that while they can improve average performance, they do not reduce worst-case complexity. Your exercises demonstrate a solid understanding of asymptotic behavior, with correct simplifications and bounds for various functions.

74 words

Permalink Show parent

Re: Week 1

by Mahmud Hossain Sushmoy - Wednesday, 26 June 2024, 9:33 PM

Hello Tyler,
Your description of brute force and branch and bound algorithms is comprehensive and accurate. You've effectively highlighted the key differences, particularly in their approach and efficiency. Your explanation of asymptotic behavior is spot-on, demonstrating a clear understanding of how it impacts algorithm selection. The examples you've provided, especially the linear search for brute force, help illustrate the concepts well. Overall, your explanation shows a strong grasp of these algorithmic concepts and their practical implications, great work!

78 words

Permalink Show parent

Re: Week 1

by Loubna Hussien - Wednesday, 26 June 2024, 1:22 PM

In your own words describe both brute force algorithms and branch and bound algorithms and identify the differences between the two. As part of your description you must discuss the Asymptotic behavior of each and discuss why this is important and how this might factor into the decision to use either algorithm design.

Welcome everyone to this course. I look forward to a semester filled with great achievements for all of us.

Concerning your first question, brute force algorithms attempt every conceivable method and possibility to locate the correct and most fitting solution for a problem. These algorithms are appreciated for their straightforwardness and ease of implementation. However, they become inefficient with large datasets because there is a direct correlation between the amount of data and the time required to solve a problem. As data increases, so does the time needed because this approach involves exploring every possible solution to find the right one. The general asymptotic behavior of brute force algorithms tends toward exponential growth.

Conversely, the branch and bound algorithm is a search method used primarily for solving optimization problems. It constructs a tree of all possible solutions and systematically eliminates those that do not lead to the desired outcome. This method is more efficient than brute force algorithms because it narrows down the number of potential solutions that need to be considered. The asymptotic behavior of the branch and bound algorithm varies greatly, depending on the elimination of unlikely probabilities and the specific characteristics of the problem.

From this comparison, the fundamental difference between these algorithms lies in their efficiency and approach to identifying potential correct solutions. Brute force algorithms are less effective for larger problems due to their comprehensive search approach, whereas branch and bound algorithms enhance efficiency by filtering out improbable solutions without examining them.

The importance of understanding asymptotic behavior lies in its impact on the time taken to solve a problem, particularly crucial when dealing with large datasets. If the data is vast and high efficiency is necessary, one would opt for branch and bound algorithms. If the problem is smaller and less efficiency is required, brute force algorithms might be more appropriate. Thus, the choice of algorithm is influenced by both the size of the problem and the needed efficiency to reach a solution.

Discussion Assignment #2
Complete the following exercises and post your answers to the discussion forum. In each assignment the questions and answers are provided so that you can self-check your work. What is important is your ability to show your work or line of reasoning. After submitting your assignment you should engage with your peers to both validate your own techniques and perhaps critique the work of others. The objective of this assignment is for all of the members of the class to work collaboratively to develop a solid understanding of the Asymptotic assessment of an algorithm and the ability to solve recurrence relations

Exercise 1
f(n) = $n^6 + 3^n$
f(n) = $2^n + 12$
f(n) = $3^n + 2^n$
f(n) = $n^n + n$

f(n) = $n^6 + 3^n$

Function Analysis: We notice that the main element in this function grows linearly with n^6, while the secondary element grows linearly with n. Therefore, we focus on the main element to determine the asymptotic behavior.

Asymptotic Estimation: We can use O-notation to estimate the asymptotic behavior, and find that $f(n)=O(n^6)$.

f(n) = $2^n + 12$

Function Analysis: The main element in this function grows linearly with n, and there is a constant term (12) that does not depend on the input size.

Asymptotic Estimation: Considering only the main element, we get $f(n)=O(n)$.

f(n) = $3^n + 2^n$

Function Analysis: Here, we have two elements that grow linearly with n, so we need to consider both.

Asymptotic Estimation: After considering both elements, we find that $f(n)=O(n)$.

f(n) = n^n + n

Function Analysis: The main element here is **n^n** , and there is a constant element that grows linearly with n.

Asymptotic Estimation: By looking at the main element only, we find that $f(n)=O($**n^n** $)$.

Exercise 2:
For the following Θ complexities write down a tight and a non-tight O bound, and a tight and non-tight Ω bound of your choice, providing they exist.

Θ(1)
Θ(√n)
Θ(n)
Θ(n2)
Θ(n3)

Θ(1):

Tight O Bound: O(1) (constant time complexity).

Non-Tight O Bound: O(n) (linear time complexity).

Tight Ω Bound: Ω(1) (tight lower bound).

Non-Tight Ω Bound: Ω(log n) (non-tight lower bound).

Θ(√n):

Tight O Bound: O(√n) (square root time complexity).

Non-Tight O Bound: O(n) (linear time complexity).

Tight Ω Bound: Ω(√n) (tight lower bound).

Non-Tight Ω Bound: Ω(1) (non-tight lower bound).

Θ(n):

Tight O Bound: O(n) (linear time complexity).

Non-Tight O Bound: O(n^2) (quadratic time complexity).

Tight Ω Bound: Ω(n) (tight lower bound).

Non-Tight Ω Bound: Ω(log n) (non-tight lower bound).

Θ(n^2):

Tight O Bound: O(n^2) (quadratic time complexity).

Non-Tight O Bound: O(n^3) (cubic time complexity).

Tight Ω Bound: Ω(n^2) (tight lower bound).

Non-Tight Ω Bound: Ω(n) (non-tight lower bound).

Θ(n^3):

Tight O Bound: O(n^3) (cubic time complexity).

Non-Tight O Bound: O(2^n) (exponential time complexity).

Tight Ω Bound: Ω(n^3) (tight lower bound).

Non-Tight Ω Bound: Ω(n^2) (non-tight lower bound).

These solutions provide both tight and non-tight upper and lower bounds for each given Θ complexity. They represent the best and worst-case scenarios for the time complexity of algorithms with these complexities.

References:

(n.d.).UOPeople. https://my.uopeople.edu/pluginfile.php/1846725/mod_book/chapter/498451/chapter03-algointro.pdf

(n.d.).(2020). YouTube. https://www.youtube.com/watch?v=kdTpUjd71G8

(n.d.). (2019). YouTube. https://www.youtube.com/watch?v=w0H3YRfx23o

916 words

Permalink Show parent

Re: Week 1
by Wingsoflord Ngilazi - Wednesday, 26 June 2024, 5:49 PM

Thank you for this comprehensive analysis. Your work demonstrates a commendable understanding of the asymptotic analysis. Keep up the good work.
21 words

Permalink Show parent

Re: Week 1
by Mahmud Hossain Sushmoy - Wednesday, 26 June 2024, 9:39 PM

Hello Loubna,
Your explanation of brute force and branch and bound algorithms demonstrates a solid understanding of their fundamental differences and applications. You accurately describe brute force's exhaustive approach and branch and bound's more strategic method, correctly noting their contrasting efficiencies with large datasets. Your discussion of asymptotic behavior is generally on point, though there's a minor misinterpretation in Exercise 1 regarding dominant terms (e.g., n^6 dominates 3n for large n). Overall, your work reflects a good grasp of algorithmic complexity and its practical implications, thank you!
87 words

Permalink Show parent

Re: Week 1
by Tousif Shahriar - Wednesday, 26 June 2024, 6:42 PM

The brute force and branch and bound algorithms are fundamental approaches for solving computational problems. They have distinct characteristics and applications that make them different. It is important to understand their differences in terms of asymptotic behaviour so that we can use them in appropriate scenarios.

Brute Force Algorithms:

Brute force algorithms are straightforward methods which disregard computational power and the need for efficiency. It involves systematically enumerating all possible options for a solution and checking whether each option satisfies the problem's requirements. This approach guarantees finding the optimal solution if one exists, as it exhaustively explores the entire solution space (Cormen et al., 2009). Since it is going through all possible solutions, the time complexity is exponential, and the asymptotic behaviour is generally poor. According to Papadimitriou & Steiglitz (1998), depending on the problem it's O2n or O(n!). So, as the size of the input (n) increases, the time taken to solve the problem grows very rapidly.

It is important to note that the major drawback of brute force algorithms is that it is inefficient for large problems since they have a high time complexity, so in a real-world scenario, it might be even considered useless where there is a need to maintain fast responses. However, it is still useful for small problems to set a baseline that be compared to more

sophisticated algorithms.

Branch and Bound Algorithms:

Branch and bound algorithms are more sophisticated and generally utilized for optimization problems. The travelling salesman problem or integer programming are examples where the branch and bound algorithms are generally used. As the name suggests, the algorithm divides the problem (Branches) into smaller subproblems and calculates the best possible solution from the subproblems. Subproblems that cannot yield a better solution than the current best are discarded (bound), reducing the search space and improving efficiency (Korte & Vygen, 2018).

The asymptotic behaviour of branch and bound algorithms is generally better than that of brute force algorithms, yet the worst-case time complexity can still be exponential O2n. However the bounding process often prunes large portions of the search space, so this leads to significantly faster performance (Cormen et al., 2009). The time complexity can vary drastically depending on the problem's structure and the effectiveness of the bounding function. Branch and bound algorithms are a more practical approach to solving large optimization problems. By pruning the search space, we can get optimal or near-optimal solutions much faster than brute-force methods. This efficiency makes branch and bound suitable for real-world applications where computational resources and time are limited.

So given the size of the problem and the need for efficiency we can choose between brute force or branch and bound.

Exercise 1:

For each of the functions, we can identify the dominant term as n grows. This will help us determine the asymptotic behaviour.

1. fn=n6+3n

Here we can see that as n increases, the term 3n grows exponentially faster than n6. So, we can conclude that 3n is the dominant term. Hence the answer is fn=Θ(3n).

2. fn=2n+12

Like the previous one, we can see that 2n will grow exponentially as n grows larger. 12 does not change since 12 is a constant. So, 2n is the dominant term and the answer is fn=Θ(2n).

3. fn=3n+2n

Here, both 3n and 2n grow exponentially, but 3n grows faster than 2n. So, 3n is the dominant term and the answer is fn=Θ(3n).

4. fn=nn+n

Again, we can see clearly that the dominant term is nn since it will grow much faster than n. So, the answer is fn=Θ(nn).

Exercise 2:

1. Θ (1)

Tight O bound = O (1). Since Θ (1) means the function is constant, the tight upper bound is O (1).

Tight Ω bound = Ω (1). Given that Θ (1), the function is at least constant, so the tight lower bound is Ω (1).

Non-tight O bound = O(n). Since n grows with the input size while the function remains constant, O(n) is an upper bound but non-tight.

There is no non-tight lower bound for a constant function since there is nothing smaller than a constant that will serve as a valid lower bound.

2. Θ(n)

Tight O bound = O(n). Θn indicates that the function grows as the square root of n. hence the tight upper bound is O (n).

Tight Ω bound = Ω (n). The function grows at least as fast as n.

Non-tight O bound = O(n). This is an upper bound but not tight since n grows faster than n.

Non-tight Ω bound = Ω (1). This is also a valid lower bound but not tight.

3. Θ (n)

Tight O bound = O(n). Θn indicates that the function has a linear growth. hence the tight upper bound is O(n).

Tight Ω bound = Ω (n). The function grows at least as fast as n, so Ω (n) is a tight lower bound.

Non-tight O bound = O (n2). This is an upper bound but not tight since n2 grows much faster than n.

Non-tight Ω bound = Ω (n).). This is also a valid lower bound but not tight.

4. Θ (n2)

Tight O bound = O(n2). Θn2 indicates that the function has a linear growth. Hence the tight upper bound is exactly as n2.

Tight Ω bound = Ω (n2). The function grows at least as fast as n2.

Non-tight O bound = O (n3). This is an upper bound but not tight since n3 grows faster than n2.

Non-tight Ω bound = ω(1). This is also a valid lower bound but not tight since it grows faster than n2.

5. Θ (n3)

Tight O bound = O(n3). Θn3 indicates that the function has a linear growth. Hence the tight upper bound is exactly as n3.

Tight Ω bound = Ω (n3). The function grows at least as fast as n3.

Non-tight O bound = O(n4). This is an upper bound but not tight since the function grows slower than

Non-tight Ω bound = ω(2). This is a non-tight lower bound because it indicates that the function grows faster than $n2$, which is smaller and hence not tight compared to $n3$.

Reference

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

Korte, B., & Vygen, J. (2018). Combinatorial Optimization: Theory and Algorithms (6th ed.). Springer.

Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover Publications.

1207 words

Permalink Show parent

Re: Week 1

by Muritala Akinyemi Adewale - Wednesday, 26 June 2024, 9:07 PM

Hello Tousif,
Your explanation of dynamic programming and greedy algorithms is clear and informative. You correctly outline their problem-solving approaches and the scenarios where each is applicable. The differences are well-defined, emphasizing the efficiency and application scope of each algorithmic strategy.
41 words

Permalink Show parent

Re: Week 1
by Michael Oyewole - Thursday, 27 June 2024, 4:41 AM

Hi Tousif,
Thank you for your interest in the post. The approach to problem-solving and asymptotic behavior of branch and bound and brute force algorithms differ significantly. While branch and bound algorithms cleverly reduce the search space to produce better asymptotic behavior, brute force techniques go through all alternatives completely, resulting in exponential time complexity. Regards!
56 words

Permalink Show parent

Re: Week 1
by Aye Aye Nyein - Wednesday, 26 June 2024, 7:41 PM

Discussion Assignment 1:

Brute Force Algorithms vs. Branch and Bound Algorithms

Brute Force Algorithms:

Brute force algorithms solve problems by systematically trying all possibilities until the correct solution is found. They are straightforward and ensure correctness but can be inefficient for large problem sizes due to their exhaustive nature. The asymptotic behavior of brute force algorithms is often represented by high complexities, such as O(2^n) or O(n!), depending on the problem structure. This inefficiency is crucial to consider when deciding whether to use a brute force approach, especially when there are more efficient alternatives.

Branch and Bound Algorithms:

Branch and bound algorithms also aim to find the optimal solution but employ strategies to prune branches of the search space that are not promising, thereby reducing the number of possibilities explored. This approach is more efficient than brute force for many problems, as it exploits problem-specific characteristics to avoid unnecessary computations. The asymptotic behavior of branch and bound algorithms tends to be significantly better than brute force, often falling into categories like O(n log n) or even polynomial time O(n^k), where k is a small constant.

Differences:

- **Efficiency**: Brute force explores all possibilities, leading to exponential time complexities, while branch and bound selectively explore branches, achieving better time complexities.

- **Space Complexity**: Brute force may require extensive memory due to its exhaustive nature, whereas branch and bound can manage memory more efficiently by discarding unproductive branches.

- **Decision Criteria**: The choice between these approaches depends on problem size and complexity; smaller problems might tolerate brute force, but larger ones necessitate the efficiency of branch and bound.

Discussion Assignment 2

Exercise 1: Asymptotic Analysis

1. f(n) = n^6 + 3n

- Tight bound: Θ(n^6)

- Non-tight bound: O(n^6), Ω(n^6)

2. f(n) = 2n + 12

- Tight bound: Θ(n)

- Non-tight bound: O(n), Ω(1)

3. f(n) = 3n + 2n

- Tight bound: Θ(n)

- Non-tight bound: O(n), Ω(1)

4. f(n) = nn + n

- Tight bound: Θ(n^n)

- Non-tight bound: O(n^n), Ω(n^n)

Exercise 2: Θ Complexities

1. Θ(1)

- Tight bound: O(1), Ω(1)

- Non-tight bound: O(n), Ω(1)

2. Θ(√n)

- Tight bound: O(√n), Ω(√n)

- Non-tight bound: O(n), Ω(1)

3. Θ(n)

- Tight bound: O(n), Ω(n)

- Non-tight bound: O(1), Ω(1)

4. Θ(n^2)

- Tight bound: O(n^2), Ω(n^2)

- Non-tight bound: O(n), Ω(1)

5. Θ(n^3)

- Tight bound: O(n^3), Ω(n^3)

- Non-tight bound: O(n^2), Ω(n^4)

Solution Approach

For Exercise 1, each function was analyzed to determine its asymptotic complexity in terms of Big O, Big Omega, and Big Theta notations. The results were based on identifying the dominant term in the function as n grows large.

For Exercise 2, Θ complexities were categorized into tight and non-tight bounds for both O and Ω notations. Tight bounds were matched with the Θ complexity itself, while non-tight bounds were chosen appropriately based on the relationship between the complexities provided.

Understanding these complexities is crucial as it helps in assessing algorithm efficiency and scalability, guiding decisions on algorithm selection based on performance requirements and problem constraints.

505 words

Permalink Show parent

Re: Week 1

by Muritala Akinyemi Adewale - Wednesday, 26 June 2024, 9:06 PM

Hello Aye,
Your explanation of breadth-first search (BFS) and depth-first search (DFS) is clear and informative. You correctly outline their time complexities and the scenarios where each is applicable. The differences are well-defined, emphasizing the efficiency and specific use cases of each traversal technique.
44 words

Permalink Show parent

Re: Week 1

by Mahmud Hossain Sushmoy - Wednesday, 26 June 2024, 9:35 PM

Hello Aye,
Your responses to both discussion assignments demonstrate a solid understanding of algorithmic analysis and asymptotic notations. You've effectively contrasted brute force and branch and bound algorithms, highlighting their efficiency differences and use cases. Thank you for your contribution to the discussion forum this week.
46 words

Permalink Show parent

Re: Week 1

by Tousif Shahriar - Wednesday, 26 June 2024, 10:13 PM

Hello Aye,
Great comparison of brute force and branch and bound algorithms! You highlighted the key differences in efficiency and space complexity. I appreciated how you emphasized the importance of considering problem size and complexity when choosing between the two approaches. While branch and bound can be more efficient, its time complexity can still vary

widely depending on the problem.
Thanks for sharing!
63 words

Permalink Show parent

Re: Week 1
by Prince Ansah Owusu - Thursday, 27 June 2024, 3:08 AM

Your submission effectively contrasts brute force and branch and bound algorithms, highlighting their differences in efficiency and space complexity. The detailed asymptotic analysis for both exercises demonstrates a clear understanding of the concepts. Well-done on explaining your thought process and methodology comprehensively.
42 words

Permalink Show parent

Re: Week 1
by Winston Anderson - Wednesday, 26 June 2024, 8:36 PM

Brute Force Algorithms

Brute force algorithms, or exhaustive search algorithms, solve problems by systematically enumerating all possible candidates and checking each one to see if it satisfies the problem's requirements. This approach is straightforward and guarantees finding a solution if one exists, but it can be highly inefficient for large problem spaces.

Example:

A classic example is the traveling salesman problem (TSP), where the brute-force solution involves calculating the total distance for every possible route and selecting the shortest one. For a padlock with a 4-digit combination, a brute-force approach would try all 10,000 possible combinations (0000 to 9999) until the correct one is found(freeCodeCamp, 2020) (GeeksforGeeks, 2024).

Asymptotic Behavior:

The time complexity of brute force algorithms is generally very high, often exponential. For example, the TSP has a time complexity of O(n!), where n is the number of cities. For string matching, the time complexity is O(n \cdot m), where n is the length of the text and m is the length of the pattern[1](GeeksforGeeks, 2024).

Importance:

Understanding the asymptotic behavior is crucial because it helps predict how the algorithm will perform as the input size grows. Brute force algorithms are often impractical for large inputs due to their poor scalability. They are typically used when the problem size is small or as a baseline to compare more sophisticated algorithms(GeeksforGeeks, 2024)(StudySmarter, n.d.).

Branch and Bound Algorithms

Branch and bound (B&B) algorithms are used for optimization problems. They divide the problem into smaller subproblems (branching) and use bounds to eliminate subproblems that cannot contain the optimal solution (bounding). This method systematically explores the solution space and prunes branches that do not lead to better solutions[8](GeeksforGeeks, 2024).

Example:

In the TSP, a branch and bound algorithm would start by calculating a lower bound on the shortest possible route and then explore routes that could improve this bound. If a partial route already exceeds the current best solution, it is pruned[8] (javatpoint, n.d.).

Asymptotic Behavior:

The time complexity of branch and bound algorithms can vary widely depending on the problem and the effectiveness of the bounding function. In the best case, they can significantly reduce the search space compared to brute force methods. However, in the worst case, they may still exhibit exponential time complexity[8](Datta, 2024).

Importance:

The asymptotic behavior of branch and bound algorithms is important because it determines their efficiency and feasibility for large problem instances. Effective bounding functions can drastically reduce the number of subproblems that need to be explored, making these algorithms more practical for large-scale optimization problems(GeeksforGeeks, 2024)(Datta, 2024).

Differences Between Brute Force and Branch and Bound

1. Approach:

- Brute Force: Systematically checks all possible solutions without any optimization.
- Branch and Bound: Uses a systematic search with pruning to eliminate suboptimal solutions early.

2. Efficiency:

- Brute Force: Generally inefficient with high time complexity, often exponential.
- Branch and Bound: More efficient due to pruning, but still can be exponential in the worst case.

3. Use Cases:

- Brute Force: Suitable for small problem sizes or as a baseline for comparison.
- Branch and Bound: Suitable for larger optimization problems where pruning can significantly reduce the search space.

4. Implementation Complexity:

- Brute Force: Simple to implement.
- Branch and Bound: More complex due to the need for effective bounding functions and systematic branching.

Conclusion

Choosing between brute force and branch and bound algorithms depends on the problem size, the need for optimization, and available computational resources. Brute force is straightforward but often impractical for large problems due to its poor asymptotic behavior. While more complex, branch and bound can offer significant efficiency improvements by reducing the search space through effective pruning. Understanding the asymptotic behavior of each algorithm is essential for making informed decisions about their use in solving computational problems.

Exercise 1: Asymptotic Analysis

We need to determine the Big-O, Big-Omega, and Big-Theta notations for each function. Here's the detailed analysis:

1.

- Big-O (Upper Bound): The term with the highest growth rate is n^6. Therefore, this term dominates the upper bound.
- Big-Omega (Lower Bound): The term with the highest growth rate also provides the lower bound.
- Big-Theta (Tight Bound): Since the upper and lower bounds are n^6, the tight bound is also n^6.

2.

- Big-O (Upper Bound): The term with the highest growth rate is $2n$. Therefore, this term dominates the upper bound.
- Big-Omega (Lower Bound): The term with the highest growth rate also provides the lower bound.
- Big-Theta (Tight Bound): Since the upper and lower bounds are n, the tight bound is also n.

3.

- Big-O (Upper Bound): The term with the highest growth rate is 2^n. Therefore, this term dominates the upper bound.
- Big-Omega (Lower Bound): The term with the highest growth rate also provides the lower bound.
- Big-Theta (Tight Bound): Since the upper and lower bounds are 2^n, the tight bound is also 2^n.

4.

- Big-O (Upper Bound): The term with the highest growth rate is n^n. Therefore, this term dominates the upper bound.

- Big-Omega (Lower Bound): The term with the highest growth rate also provides the lower bound.

- Big-Theta (Tight Bound): Since the upper and lower bounds are n^n, the tight bound is also n^n.

Exercise 2: Tight and Non-Tight Bounds

For each given Θ complexity, we need to write down a tight and a non-tight O bound, and a tight and non-tight Ω bound.

a)

- Tight O bound:
- Non-tight O bound:
- Tight Ω bound:
- Non-tight Ω bound:

b)

- Tight O bound:
- Non-tight O bound:
- Tight Ω bound:
- Non-tight Ω bound:

c)

- Tight O bound:
- Non-tight O bound:
- Tight Ω bound:
- Non-tight Ω bound:

d)

- Tight O bound:
- Non-tight O bound:

- Tight Ω bound:

- Non-tight Ω bound:

e)

- Tight O bound:

- Non-tight O bound:

- Tight Ω bound:

- Non-tight Ω bound:

Explanation and Methodology

To solve these exercises, I followed these steps:

1. Identify Dominant Terms: Identify the term with the highest growth rate for each function, as it will dominate the asymptotic behavior.

2. Determine Big-O: The Big-O notation represents the upper bound, so I selected the term with the highest growth rate.

3. Determine Big-Omega: The Big-Omega notation represents the lower bound, which is also the term with the highest growth rate.

4. Determine Big-Theta: If both the upper and lower bounds are the same, the function is tightly bound by that term, giving us the Big-Theta notation.

5. Tight and Non-Tight Bounds: For each Θ complexity, I provided a tight bound (the exact growth rate) and a non-tight bound (a looser bound that still holds true).

Example

For the function f(n) = n^6 + 3n :

- The dominant term is n^6 .

- Therefore, the Big-O, Big-Omega, and Big-Theta notations are all n^6 .

For the complexity \Theta(n) :

- A tight O bound is O(n) because it exactly matches the growth rate.

- O(n^2) is a non-tight O bound because it is a looser upper bound.

- A tight Ω bound is \Omega(n) because it exactly matches the growth rate.

- A non-tight Ω bound is \Omega(1) because it is a looser lower bound.

By following these steps, I ensured that the asymptotic notations accurately represent the growth rates of the functions and complexities provided.

References

Datta, S. (2024, March 18). *Branch and Bound Algorithm*. baeldung.com. https://www.baeldung.com/cs/branch-and-bound

freeCodeCamp. (2020, January 6). *Brute Force Algorithms Explained*. freeCodeCamp.org. https://www.freecodecamp.org/news/brute-force-algorithms-explained/

GeeksforGeeks. (2024, February 22). *Branch and Bound Algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/branch-and-bound-algorithm/

GeeksforGeeks. (2024, January 18). *Brute Force Approach and its pros and cons*. GeeksforGeeks. https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

javatpoint. (n.d.). *Branch and bound - javatpoint*. www.javatpoint.com. https://www.javatpoint.com/branch-and-bound

StudySmarter. (n.d.). *Brute Force: Algorithm, Technique & Meaning*. StudySmarter UK. https://www.studysmarter.co.uk/explanations/computer-science/algorithms-in-computer-scien

1448 words

Permalink Show parent

Re: Week 1

by Muritala Akinyemi Adewale - Wednesday, 26 June 2024, 9:06 PM

Hello Winston,
Your explanation of quicksort and mergesort algorithms is clear and informative. You correctly outline their time complexities and the scenarios where each is applicable. The differences are well-defined, emphasizing the efficiency and use cases of each sorting algorithm.
40 words

Permalink Show parent

Re: Week 1

by Christopher Mccammon - Thursday, 27 June 2024, 3:42 AM

Hi Winston
You clearly explains the concept of brute force algorithms with a focus on their exhaustive nature. The examples provided (TSP and padlock combination) effectively illustrate the concept. Your explanation of the typically high time complexity, with the TSP and string matching examples illustrating the concept well. The discussion on why understanding asymptotic behavior is crucial is well-articulated. You effectively convey that brute force algorithms are often impractical for large inputs but useful for small problems or as a baseline. Good Job!
83 words

Permalink Show parent

Re: Week 1

by Christopher Mccammon - Wednesday, 26 June 2024, 10:29 PM

Brute Force Algorithms
When it comes to problem solving, brute force algorithms take an approach by evaluating every solution to find the optimal result. This method involves a search process, which's simple but may not be efficient, for complex problems(Shaffer,2010).
Key Points;
•Method; Brute force methodically assesses all combinations or arrangements of inputs ensuring that a solution is found if it exists.

•Complexity; Due to its nature brute force algorithms often have computational complexity.
• An illustration of this concept is solving the Traveling Salesman Problem (TSP) by examining all routes to determine the one.
Behavior Over Time;
•Time Complexity; Brute force algorithms commonly exhibit high time complexity in forms like O(2^n) or O(n!). For instance in TSP analyzing all permutations of 'n' cities results in a time complexity of O(n!).
•Space Complexity; While time complexity is a concern for brute force approaches space complexity can also become significant when storing solutions(Shaffer,2010).
Significance
•The substantial time complexity associated with force makes it impractical for large scale problems. Typically employing force is reasonable, for small datasets or when no superior algorithm is available.
Understanding how the algorithm behaves as it approaches its limits helps us gauge its ability to handle problem sizes (Shaffer,2010)
Branch and Bound Algorithms
Branch and bound algorithms represent a approach, to tackling optimization problems by methodically exploring solution spaces and trimming off branches that do not lead to optimal solutions. This technique enhances efficiency by reducing the search space (Shaffer,2010)
Key Features
•Methodology; Branch and bound break down the problem into subproblems (branching). Use boundaries to discard subproblems that cannot yield a better solution than the current best one (pruning) (Shaffer,2010).
•Complexity; Generally branch and bound are more efficient than force for intricate and large scale problems.
•Example; Resolving the Traveling Salesman Problem involves assessing subsets of routes while applying boundaries to avoid delving into routes than the shortest one.
Asymptotic Behavior
•Time Complexity; The time complexity of branch and bound algorithms tends to be lower than force. It largely hinges on how branchesre pruned. For instance in the Traveling Salesman Problem adept pruning can substantially slash the number of evaluated routes resulting in a time complexity, to O(b^d) where b represents the branching factor and d denotes the depth of the search tree(Shaffer,2010).

• Managing Space Complexity; To effectively handle space complexity it is advisable to store the components of the search tree, which often results in better space efficiency compared to brute force approaches(Shaffer,2010).

Significance
• The improved time complexity of branch and bound algorithms makes them more practical, for solving problems. Understanding their behavior is key to assessing their effectiveness and applicability in scenarios where brute force methodsre not viable(Shaffer,2010).

Contrasts Between Brute Force and Branch and Bound Techniques
1. Efficiency; While brute force algorithms exhaustively examine all solutions guaranteeing a result they can be computationally costly for inputs. On the hand branch and bound algorithms enhance efficiency by discarding unpromising options on reducing the total number of paths to investigate(Shaffer,2010).
2. Complexity; Brute force strategies often exhibit exponential time complexity rendering them unsuitable for problems. In contrast branch and bound algorithms aim for polynomial or pseudo polynomial time complexities by efficiently pruning possibilities(Shaffer,2010).
3. Application; Brute force is typically employed for problem instances or when no superior alternative exists. Branch and bound is favored for larger scale challenges or scenarios where optimization plays a role due, to its ability to handle situations efficiently(Shaffer,2010).
When deciding between using force or branch and bound methods it all comes down to the size of the problem how much optimization is needed and the resources available. Brute force works well for small problems whereas branch and bound is better suited for tackling larger problems that require efficiency(Shaffer,2010).

Asymptotic Analysis Exercises
Exercise 1
1. f(n)=n6+3nf(n) = n^6 + 3nf(n)=n6+3n
o Dominant term: n6n^6n6
o As n→∞n \to \inftyn→∞, n6n^6n6 grows faster than 3n3n3n.
o Result: f(n)∈Θ(n6)f(n) \in \Theta(n^6)f(n)∈Θ(n6)
2. f(n)=2n+12f(n) = 2^n + 12f(n)=2n+12
o Dominant term: 2n2^n2n

o As n→∞n \to \inftyn→∞, 2n2^n2n grows exponentially, overshadowing the constant 12.
o Result: f(n)∈Θ(2n)f(n) \in \Theta(2^n)f(n)∈Θ(2n) (Complexity Zoo,n.d)
3. f(n)=3n+2nf(n) = 3^n + 2^nf(n)=3n+2n
o Dominant term: 3n3^n3n
o As n→∞n \to \inftyn→∞, 3n3^n3n grows faster than 2n2^n2n.
o Result: f(n)∈Θ(3n)f(n) \in \Theta(3^n)f(n)∈Θ(3n)
4. f(n)=nn+nf(n) = n^n + nf(n)=nn+n
o Dominant term: nnn^nnn
o As n→∞n \to \inftyn→∞, nnn^nnn grows extremely fast, dwarfing nnn.
o Result: f(n)∈Θ(nn)f(n) \in \Theta(n^n)f(n)∈Θ(nn) (Complexity Zoo,n.d)
Exercise 2
For each Θ\ThetaΘ complexity, we provide tight and non-tight bounds.
1. Θ(1)\Theta(1)Θ(1)
o Tight O-bound: O(1)O(1)O(1)
o Non-tight O-bound: O(n)O(n)O(n)
o Tight Ω\OmegaΩ-bound: Ω(1)\Omega(1)Ω(1)
o Non-tight Ω\OmegaΩ-bound: Not applicable since 111 is the lowest bound.
2. Θ(n)\Theta(\sqrt{n})Θ(n)
o Tight O-bound: O(n)O(\sqrt{n})O(n)
o Non-tight O-bound: O(n)O(n)O(n)
o Tight Ω\OmegaΩ-bound: Ω(n)\Omega(\sqrt{n})Ω(n)
o Non-tight Ω\OmegaΩ-bound: Ω(1)\Omega(1)Ω(1) (Complexity Zoo,n.d)
3. Θ(n)\Theta(n)Θ(n)
o Tight O-bound: O(n)O(n)O(n)
o Non-tight O-bound: O(n2)O(n^2)O(n2)
o Tight Ω\OmegaΩ-bound: Ω(n)\Omega(n)Ω(n)
o Non-tight Ω\OmegaΩ-bound: Ω(1)\Omega(1)Ω(1)
4. Θ(n2)\Theta(n^2)Θ(n2)
o Tight O-bound: O(n2)O(n^2)O(n2)
o Non-tight O-bound: O(n3)O(n^3)O(n3)
o Tight Ω\OmegaΩ-bound: Ω(n2)\Omega(n^2)Ω(n2)
o Non-tight Ω\OmegaΩ-bound: Ω(n)\Omega(n)Ω(n)
5. Θ(n3)\Theta(n^3)Θ(n3)
o Tight O-bound: O(n3)O(n^3)O(n3)
o Non-tight O-bound: O(n4)O(n^4)O(n4)
o Tight Ω\OmegaΩ-bound: Ω(n3)\Omega(n^3)Ω(n3)
o Non-tight Ω\OmegaΩ-bound: Ω(n2)\Omega(n^2)Ω(n2) (Complexity Zoo,n.d)
Discussion on Solving the Exercises
In Exercise 1 we look at finding the behavior (Θ notation) of given functions. This involves identifying the term, which's the term that grows the fastest, as n approaches infinity. For example in f(n) = n^6 + 3n n^6 is the term because it increases faster than 3n indicating that f(n) belongs to Θ(n^6).

Exercise 2 involves determining non bounds for various Θ complexities. A tight bound matches the functions growth rate exactly while a non tight bound is a bit more flexible but still valid. For Θ(n) the tight O bound is O(n) whereas a non tight O bound could be O(n^2) which serves as a bound but isn't a match to ns growth rate.

Reference:
Complexity Zoo. (n.d.). Retrieved June 25, 2024, from http://discrete.gr/complexity/
Shaffer, C. A. (2010). A practical introduction to data structures and algorithm analysis: Third edition (C++ version). Virginia Tech. https://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf
996 words

Permalink Show parent

Re: Week 1
by Romana Riyaz (Instructor) - Thursday, 27 June 2024, 12:25 AM

Christroper,
Thank you for your submission. You effectively explain brute force and branch and bound algorithms, highlighting their methodologies, complexities, and practical applications.
You provide detailed examples of asymptotic complexity analysis for various functions, demonstrating a good understanding of identifying dominant terms. The comparison between brute force and branch and bound algorithms is well articulated, focusing on efficiency, complexity, and suitability for different problem sizes. The document would benefit from improved formatting to enhance readability and structure, especially in the sections discussing Θ, O, and Ω bounds. There are instances where mathematical symbols or expressions are unclear or incorrectly formatted, which could be refined for better precision and understanding.
While you cover various aspects of algorithmic complexity and provide examples, ensuring completeness and consistency in formatting and explanation throughout the document would improve clarity.
Best,
Romana
136 words

Permalink Show parent

Re: Week 1
by Winston Anderson - Thursday, 27 June 2024, 3:13 AM

Hi Christopher,
Your submission offers a comprehensive overview of brute force and branch and bound algorithms, clearly explaining their methodologies, complexities, and applications. Your detailed examples, especially with the Traveling Salesman Problem, effectively illustrate the practical implications of each algorithm's complexity. The reference to Shaffer's work adds credibility and depth to your explanation.
53 words

Permalink Show parent

Re: Week 1
by Naqaa Alawadhi - Wednesday, 26 June 2024, 10:46 PM

Brute force algorithms exhaustively search through all possible solutions to find the optimal one. They are simple but can be inefficient for large problem sizes. Their asymptotic behavior is often exponential, leading to high time complexity.

Branch and bound algorithms systematically divide the problem into smaller subproblems, pruning branches that cannot lead to the optimal solution. They are more efficient than brute force for many problems. Their asymptotic behavior is typically better than brute force, often polynomial.

The choice between these algorithms depends on the problem size and complexity. For small problems, brute force may be acceptable despite its inefficiency. However, for larger problems, branch and bound's better asymptotic behavior makes it a more suitable choice as it can significantly reduce computation time and resources.

1. For :

- The dominant term is as it grows much faster than 3n.

- So, the asymptotic complexity is .

2. For :

- Both terms are of the same order, but the dominant term is 2n.

- Thus, the asymptotic complexity is .

3. For :

- Combining like terms gives us 5n.

- Therefore, the asymptotic complexity is .

4. For :

- This simplifies to .

- The dominant term here is .

- Hence, the asymptotic complexity is .

For the given complexities:

1. Θ(1):

- Tight O bound: O(1)

- Non-tight O bound: O(n)

- Tight Ω bound: Ω(1)

- Non-tight Ω bound: Ω(1)

2. Θ(√n):

- Tight O bound: O(√n)

- Non-tight O bound: O(n)

- Tight Ω bound: Ω(√n)

- Non-tight Ω bound: Ω(log n)

3. Θ(n):

- Tight O bound: O(n)

- Non-tight O bound: O(n^2)

- Tight Ω bound: Ω(n)

- Non-tight Ω bound: Ω(log n)

4. Θ(n^2):

- Tight O bound: O(n^2)

- Non-tight O bound: O(n^3)

- Tight Ω bound: Ω(n^2)

- Non-tight Ω bound: N/A

5. Θ(n^3):

- Tight 0 Bound : 0 (n^3)

- Non-Tight 0 Bound : 0 (n^4)

- Tight Omega Bound : Omega (n^3)

- Non-Tight Omega Bound : Omega (n^2)

To solve this exercise, I first identified the given complexities and then determined their tight and non-tight upper bounds (O) and lower bounds (Ω). For each complexity, I considered the dominant term in the expression to establish the bounds.

For example, for complexity Θ(√n), if we have an algorithm that performs a constant-time operation for each element in a list of size √n, the time complexity would be proportional to √n. Therefore, the tight upper and lower bounds are both √n.

Similarly, for complexity Θ(n), if we have an algorithm that iterates through a list of size n linearly, the time complexity would be directly proportional to n. Hence, both tight upper and lower bounds are n.

By understanding how the complexities scale with input size and identifying dominant terms in expressions, we can determine appropriate tight and non-tight bounds for different complexities.

1. is in .
2. is in .
3. is in .
4. is in .

564 words

Permalink Show parent

Re: Week 1

by Romana Riyaz (Instructor) - Thursday, 27 June 2024, 12:21 AM

Naqaa,
Thank you for your submission. You provide concise definitions and distinctions between brute force and branch and bound algorithms, highlighting their respective efficiencies and suitable problem sizes. You effectively demonstrate the

process of determining asymptotic complexities for various functions, breaking down each step clearly. While you provide several examples of asymptotic complexity analysis, ensuring completeness and consistency in formatting and explanation would enhance clarity.
Best,
Romana
67 words

Permalink Show parent

Re: Week 1
by Prince Ansah Owusu - Thursday, 27 June 2024, 3:06 AM

Your explanation of brute force and branch and bound algorithms is clear and concise, highlighting their key characteristics and differences. Your analysis of the asymptotic complexities for Exercise 1 is mostly correct, except for the notation errors in your results. The correct asymptotic notations for Exercise 1 should be:
1. is in .
2. is in .
3. is in .
4. is in .

Your approach to solving Exercise 2 is well-explained, and the tight and non-tight bounds you provided are accurate. Great job!
121 words

Permalink Show parent

Re: Week 1
by Winston Anderson - Thursday, 27 June 2024, 3:09 AM

Hi Naqaa,
Your submission provides a clear and concise distinction between brute force and branch and bound algorithms, along with well-explained asymptotic analyses of given functions. You've done a good job demonstrating the inefficiencies of brute force approaches compared to the more strategic branch and bound method, especially for large problem sizes.
52 words

Permalink Show parent

Re: Week 1
by Chong-Wei Chiu - Thursday, 27 June 2024, 10:39 AM

Hello, Naqaa Alawadhi. Thank you for sharing your thoughts on these two different algorithms. You clearly illustrate brute force algorithms and branch and bound algorithms. Furthermore, you also state when to use these two different algorithms. However, don't forget to add the references you used in this post, as it would make it even better!
55 words

Permalink Show parent

Re: Week 1
by Michael Oyewole - Thursday, 27 June 2024, 1:35 AM

Discussion Assignment #1
In computer science, there are two distinct methods for solving problems: brute force algorithms and branch and bound algorithms.

Brutal Force Techniques
Brute force algorithms work by methodically examining each potential answer in order to choose the best one. Although this approach is simple, it may not be effective when dealing with big problem sizes. When n is the amount of the input, the asymptotic behavior of brute force algorithms is often exponential and is expressed as O(2^n) or O(n!). This indicates that when the size of the input increases, the temporal complexity does so quickly.
Algorithms for Bound and Branch

The algorithms used in branch and bound are more advanced. They entail decomposing the problem into more manageable subproblems and utilizing boundaries to rule out particular subproblems. By using this method, the amount of solutions that need to be investigated can be greatly decreased. Generally speaking, branch and bound algorithms exhibit superior asymptotic behavior than brute force, which is sometimes represented as O(b^d), where b is the branching factor and d is the search tree depth.

Differences
The following are the two algorithms' main distinctions:

1. Branch and bound method selectively investigates subproblems based on bounds, whereas brute force method thoroughly verifies every potential solution.
2. Branch and bound algorithms frequently have better time complexity than brute force algorithms, especially for large issue sizes. Brute force techniques have exponential time complexity.

Why Asymptotic Behavior Is Important
Because it shows how the methods would scale with bigger issue sizes, the asymptotic behavior is important. Due to their rapidly increasing time complexity, exponential techniques, such as brute force, might become unfeasible for huge input sizes (Datta & Aibin, 2024). On the other hand, because of their more controllable temporal complexity, algorithms with superior asymptotic behavior, such branch and bound, might be more appropriate for bigger issue sizes.

Selecting Which Algorithm to Use
The problem's size and nature should be taken into account when choosing between the two algorithm designs. An approach using brute force might be appropriate for small issue sizes. Branch and bound algorithms are preferred because of their superior asymptotic behavior, while brute force techniques may become unfeasible due to their exponential nature for higher issue sizes.

Discussion Assignment #2

Exercise 1

1. f(n) = n6 + 3n (3n), because 3n will grow larger than n6 when n = 17
2. f(n) = 2n + 12 (2n), because a constant 12 cannot change and 2n will grow for different n values.
3. f(n) = 3n + 2n (3n), because 3n will grow larger than 2n for every value of n.
4. f(n) = nn + n(nn), because nn will grow larger than n for n different value

Exercise 2

1. O(1) and Ω(1) are the tight upper bounds. Since the function value cannot be less than 1, the non-tight bound is o(n), and we are unable to find a non-tight lower bound.
2. O(√n) and Ω(√n) are tight upper bounds. The constraints Ω(1) and O(√n) are not tight.
3. O(n) and Ω(n) are the tight bounds. Both Ω(1) and O(n2) are non-tight bounds.
4. O(n2) and (n2) are the tight bounds. o(n3) and Ω(1) are non-tight limits.
5. O(n3) and (n3) are the tight bounds. O(n4) and (n2) are non-tight bounds.

References
Datta, S. & Aibin, M. (2024, March 18). Branch and Bound Algorithm. Baeldung on Computer Science.
https://www.baeldung.com/cs/branch-and-bound
575 words

Permalink Show parent

Re: Week 1
by Prince Ansah Owusu - Thursday, 27 June 2024, 3:02 AM

Your submission is well-organized and clearly explains the concepts of brute force and branch and bound algorithms. It effectively covers asymptotic behavior and its importance. However, there is a minor error in the first exercise: should actually be instead of .

53 words

Permalink Show parent

Re: Week 1

by Winston Anderson - Thursday, 27 June 2024, 3:05 AM

Hi Michael,
Overall, your explanation of the differences between brute force, branch and bound algorithms is informative and well-structured. You clearly outline the distinctions and implications of each algorithm's asymptotic behavior, which enhances understanding.

34 words

Permalink Show parent

Re: Week 1

by Christopher Mccammon - Thursday, 27 June 2024, 3:56 AM

Hi Michael Oyewole
The description of branch and bound algorithms is clear. You explain that they decompose problems into subproblems and use bounds to eliminate unpromising ones, which is a concise summary of their approach. You effectively highlight the main distinctions between brute force and branch and bound algorithms. You also correctly emphasize that asymptotic behavior is crucial for understanding how algorithms scale with larger input sizes. Good job!

69 words

Permalink Show parent

Re: Week 1

by Jon Pedersen - Thursday, 27 June 2024, 3:32 AM

Discussion Assignment #1

Brute Force Algorithms

Brute force algorithms are the most straightforward way to solve a problem. They work by trying all possible solutions to find the best one. Imagine you have a lock with a 4-digit code. A brute force algorithm would try every combination from 0000 to 9999 until it finds the correct one (CodeAcademy, n.d.).

Asymptotic Behavior

The time complexity of brute force algorithms can be pretty bad. If you have nnn possible solutions and each solution takes a constant amount of time to check, the time complexity is O(n). For many problems, this can quickly become impractical. For instance, if you're solving the Traveling Salesman Problem (TSP) by brute force, where you have to visit every possible permutation of cities, the time complexity is O(n!). That factorial growth is a killer as nnn gets large.
The asymptotic behavior is crucial because it tells us how the algorithm's runtime scales with the input size. If the runtime grows too quickly, it becomes impossible to use the algorithm for large inputs. So, while brute force algorithms are simple and easy to implement, they're usually only practical for small problem sizes or when you have a lot of computational power and not much time to optimize the algorithm (Utkar, 2024).

Branch and Bound Algorithms

Branch and bound algorithms are a bit more sophisticated. They work by systematically exploring the solution space, but they "prune" parts of the space that they can prove won't lead to a better solution than the best one found so far. Going back to the lock analogy, instead of trying every combination, a branch and bound algorithm might realize that if the first two digits are wrong, there's no need to try the remaining two (Hare, 2024).

Asymptotic Behavior

The time complexity of branch and bound algorithms can vary widely depending on the problem and how effectively the algorithm can prune the solution space. In the best case, it can be much faster than brute force. For example, in solving the TSP, a good branch and bound algorithm might only explore a small fraction of the possible routes, potentially bringing the complexity down from O(n!)O(n!)O(n!) to something much more manageable. However, in the worst case, it can still be as bad as brute force.
The key advantage of branch and bound is that it can drastically reduce the number of solutions you need to check, making it feasible to solve larger problems. The asymptotic behavior is important here because it gives us an idea of how much pruning we can expect. Good pruning can mean the difference between solving a problem in a reasonable time or not being able to solve it at all.

Discussion Assignment #2

Here is my general approach:

1) Identify the growth rate of each term

2) Compare the terms

3) Select the dominant term

Exercise 1

Function 1: f(n) = n^6 + 3^n

- n^6 grows polynomially.
- 3^n grows exponentially.

For large values of n, the exponential term 3^n will dominate the polynomial term n^6. Therefore, the asymptotic behavior of f(n) is determined by the exponential term.

Asymptotic Assessment: O(3^n)

Function 2: f(n)= 2^n + 12

- 2^n grows exponentially.
- 12 is a constant.

For large values of n, the exponential term 2^n will dominate the constant term 12. Therefore, the asymptotic behavior of f(n) is determined by the exponential term.

Asymptotic Assessment: O(2^n)

Function 3: f(n)= 3^n + 2^n

- Both 3^n3n and 2n2^n2n grow exponentially.
- 3n3^n3n grows faster than 2^n.

For large values of nnn, the term 3^n will dominate 2^n. Therefore, the asymptotic behavior of f(n) is determined by the larger exponential term 3^n.

Asymptotic Assessment: O(3^n)

Function 4: f(n)= n^n + n

- n^n grows faster than any polynomial or exponential function.
- n grows polynomially.

For large values of n, the term n^n will dominate n. Therefore, the asymptotic behavior of f(n) is determined by the term n^n.

Asymptotic Assessment: O(n^n)

Summary

- f(n)=n^6+3^n → O(3^n)
- f(n)=2^n+12 → O(2^n)

- f(n)=3^n+2^n → O(3^n)
- f(n))=n^n+n → O(n^n)

Exercise 2

To solve this exercise, we need to understand the definitions of tight and non-tight bounds in terms of Big-O and Omega notations (pp. 68-72, Schaffer, 2010):

1. **Tight Bound**:

- For Big-O, a tight bound means the function is an upper bound that matches the given Θ notation as closely as possible.
- For Big-Omega, a tight bound means the function is a lower bound that matches the given Θ notation as closely as possible.

Non-Tight Bound:
- For Big-O, a non-tight bound means the function is an upper bound that is larger than the tight bound.
- For Big-Omega, a non-tight bound means the function is a lower bound that is smaller than the tight bound.

1. Θ(1)

- **Tight O bound:** O(1)
- **Non-tight O bound:** O(n)
- **Tight Ω bound:** Ω(1)\
- **Non-tight Ω bound:** Ω(logn)

2. Θ(sqrt{n})

- **Tight O bound:** O(\sqrt{n})
- **Non-tight O bound:** O(n)
- **Tight Ω bound:** Ω(n)(\sqrt{n})
- **Non-tight Ω bound:** Ω(logn)

3. Θ(n)

- **Tight O bound:** O(n)
- **Non-tight O bound:** O(n^2)
- **Tight Ω bound:** Ω(n)
- **Non-tight Ω bound:** Ω(log n)

4. Θ(n^2)

- **Tight O bound:** O(n^2)
- **Non-tight O bound:** O(n^3)
- **Tight Ω bound:** Ω(n^2)
- **Non-tight Ω bound:** Ω(n)

5. Θ(n^3)

- **Tight O bound:** O(n^3)
- **Non-tight O bound:** O(n^4)
- **Tight Ω bound:** Ω(n^3)
- **Non-tight Ω bound:** Ω(n^2)

I will only go through one of them, this post is already awfully long.

Θ(n) means that the function f(n) grows at the same rate as n. In other words, f(n) is both bounded above and below by linear functions of n for large values of n.

Since f(n) is Θ(n), the tight O bound is O(n) because f(n) grows linearly with n.

A non-tight O bound would be any function that grows faster than n. For example, O(n^2).

Since f(n) is Θ(n), the tight Ω bound is Ω(n) because f(n) grows linearly with n.

A non-tight Ω bound would be any function that grows slower than n. For example, Ω(logn).

References

CodeAcademy. (n.d.). *CS102: Data Structures and algorithms: Brute force algorithms cheatsheet*. Codecademy. https://www.codecademy.com/learn/paths/computer-science/tracks/cspath-cs-102/modules/brute-force-algorithms/cheatsheet

Utkar. (2024, February 28). *Asymptotic Notations and how to calculate them*. GeeksforGeeks. https://www.geeksforgeeks.org/asymptotic-notations-and-how-to-calculate-them/

Hare. (2024, February 22). *Branch and bound algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/branch-and-bound-algorithm/

Algorithm Analysis in A Practical Introduction to Data Structures and Algorithm Analysis by Clifford A. Shaffer. http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

1035 words

Permalink Show parent

Re: Week 1
by Christopher Mccammon - Thursday, 27 June 2024, 3:49 AM

Hi John
You provide a concise definition and a straightforward example (4-digit lock) that effectively illustrates the brute force approach. This helps in grasping the concept quickly. The inclusion of a real-world example like the padlock is a good choice as it simplifies the explanation for readers unfamiliar with technical jargon. Your post is solid and demonstrates a good understanding of brute force and branch and bound algorithms, as well as the principles of asymptotic analysis. Good job!
78 words

Permalink Show parent

Re: Week 1
by Michael Oyewole - Thursday, 27 June 2024, 4:35 AM

Hi Jon,
Thank you for sharing this wonderful enlighten post. In additional to your points, branch and bound algorithms seek to lower the overall computational effort by reducing the search area, whereas brute force approaches can be computationally expensive, particularly for large issue sizes.
44 words

Permalink Show parent

Re: Week 1
by Jobert Cadiz - Thursday, 27 June 2024, 8:47 AM

Hi Jon,
Your explanations are clear and easy to follow, making complex concepts like asymptotic behavior and algorithm comparisons accessible. It demonstrates a solid grasp of the topic and effectively communicates key concepts. Good work.
35 words

Permalink Show parent

Re: Week 1
by Siraajuddeen Adeitan Abdulfattah - Thursday, 27 June 2024, 3:50 AM

Brute force algorithm is an algorithm where all possible options are considered in order to arrive at a solution, like in the case of a sequential search. Where all the elements are visited to find the actual item, we are searching for. However, it is easy to

implement but not efficient as the size becomes larger. Its asymptotic analysis is represented by O(n!), n being the size of the problem.

Branch and bound algorithm is a more sophisticated algorithm which combine systematic search and intelligent pruning of the search space, it uses bounds to decide which subproblem is worth exploring further, after dividing the problem in to subproblems known as branches. It then explores only the most promising branches based on particular conditions, thereby eliminating a large part of the search space and reduces the total computational efforts. Its asymptotic analysis depends on how efficient the pruning is and varies from exponential time complexity to polynomial time complexity.

Differences between Brute force and Branch and Bound algorithm:
Each algorithm differs in a number of characteristics listed below;

Approach: Brute force employs exhaustive mode of searching through all possible solutions, whereas Branch and bound simply divide problem in to smaller chunk of subproblem (branches) and intelligently prune only the most promising branches.
Time complexity: Branch and bound time complexities varies based on the problem and the pruning method used, while Brute force has a factorial time complexity O(n!).
Efficiency: Branch and bound seem more efficient than Brute force algorithm owing to the fact that it reduces search space through elimination of unpromising branches, especially in a problem of larger size.
Optimal performance: both algorithms can be optimized for higher performance but branch and bound optimal performance comes faster due to its pruning method used.

Asymptotic behavior and importance:
Asymptotic analysis of an algorithm describes its performance in relation to input or problem size, thereby exposing the efficiency of the algorithm and how it compares to another algorithm.
Branch and bound asymptotic behavior is based on the bounding technique used and it usually offers better asymptotic behavior than that of Brute force. Asymptotic behavior of Brute force algorithm grows with the input size making it not efficient for large problems.
Consideration for asymptotic behavior majorly points towards algorithm efficiency and scalability, asymptotic analysis provides insight in to the estimation of algorithm performance for various input or problem sizes, with which informed decisions can be made on choosing the right algorithm for a problem.

Discussion Assignment #2 Answers:

Exercise 1 Answers:
$f(n) = n^6 + 3^n \in \Theta(3^n)$
$2^n +12 \in \Theta(2^n)$
$3^n +2^n \in \Theta(3^n)$
$n^n +n \in \Theta(n^n)$
Exercise 2 Answers:
This is a straight-forward application of the definitions above.
1. The tight bounds will be O(1) and Ω(1). A non-tight O- bound would be O(n). Recall that O gives us an upper bound. As n is of larger scale than 1 this is a non-tight bound and we can write it as o(n) as well. But we cannot find a non-tight bound for Ω, as we can't get lower than 1 for these functions. So, we'll have to do with the tight bound.
2. The tight bounds will have to be the same as the Θ complexity, so they are O(√n) and Ω(√n) respectively. For non-tight bounds we can have O(n), as n is larger than and so it is an upper bound for √n. As we know this is a non-tight upper bound, we can also write it as o(n). For a lower bound that is not tight, we can simply use Ω(1). As we know that this bound is not tight, we can also write it as ω(1).
3. The tight bounds are O(n) and Ω(n). Two non-tight bounds could be ω(1) and o(n3). These are in fact pretty bad bounds, as they are far from the original complexities, but they are still valid using our definitions.
4. The tight bounds are O(n2) and Ω(n2). For non-tight bounds we could again use ω(1) and o(n3) as in our previous example.
5. The tight bounds are O(n3) and Ω(n3) respectively. Two non-tight bounds could be ω(n2)
715 words

Re: Week 1

by Michael Oyewole - Thursday, 27 June 2024, 4:33 AM

Hi Siraajuddeen,
Thank you for your post. One main difference between Brute Force and Branch and Bound Algorithms is that While branch and bound algorithms wisely investigate only the most promising branches, brute force techniques methodically check every potential answer. Thanks for your post.

44 words

Permalink Show parent

Re: Week 1

by Jobert Cadiz - Thursday, 27 June 2024, 8:53 AM

Hi Siraajuddeen,
You've provided concise definitions and explanations for both brute force and branch and bound algorithms, which helps in understanding their fundamental principles. The two methods are compared in detail, taking into account a number of factors including approach, time complexity, efficiency, and ideal performance. This provides a thorough understanding of the differences in real-world application scenarios. Great work.

60 words

Permalink Show parent

Re: Week 1

by Chong-Wei Chiu - Thursday, 27 June 2024, 10:33 AM

Hello, Siraajuddeen Adeitan Abdulfattah. Thank you for sharing your opinion on this topic. You clearly state the definitions of the brute force algorithm and the branch and bound algorithm, and you explain the differences between these algorithms well. However, you should add the references you used in your post to make it more complete.

54 words

Permalink Show parent

Re: Week 1

by Jobert Cadiz - Thursday, 27 June 2024, 5:45 AM

Discussion Assignment #1

Brute Force Algorithms

Brute force algorithms methodically search through every possible solution, regardless of how inefficient or impractical some solutions might be (Charras & Lecroq, 1997; Stack Overflow, 2011). If a solution exists, the algorithm will find it, but it won't judge the quality or efficiency of the solution. Brute force approaches rely on a generate-and-test method, trying all potential solutions to find the correct one.

Asymptotic Behavior:

- Time Complexity: These algorithms often have an exponential time complexity, such as $O(n!)$ for problems like the traveling salesman problem.
- Space Complexity: The space complexity can also be high if storing all possible solutions or intermediate states is necessary.

Importance of Asymptotic Behavior:

Knowing the asymptotic behavior helps estimate how the algorithm scales with input size. Brute force algorithms are typically impractical for large inputs due to their high time complexity.

Branch and Bound Algorithms

Branch and bound algorithms are an optimization technique used to solve combinatorial and discrete optimization problems (Shaffer, 2010). They extend backtracking methods by incorporating bounds to prune the search space, thereby avoiding the need to explore all possible solutions. These algorithms use a tree structure and explore the most promising bounds first, which helps to reduce the search space significantly.

Asymptotic Behavior:

- Time Complexity: The time complexity varies and can be polynomial in the best case, but may still be exponential in the worst case.
- Space Complexity: Typically lower than brute force algorithms due to the pruning of subproblems.

Importance of Asymptotic Behavior:

Understanding the asymptotic behavior helps in estimating efficiency. Branch and bound algorithms are usually more efficient than brute force methods because they avoid unnecessary exploration of the search space.

Differences

- Approach: Brute force algorithms explore all possible solutions, while branch and bound algorithms selectively explore based on bounds.
- Efficiency: Branch and bound algorithms are generally more efficient due to pruning, while brute force algorithms are less efficient.
- Complexity: Brute force algorithms have higher time complexity (often exponential), while branch and bound algorithms can have lower, more variable time complexity.

Discussion Assignment #2

Exercise 1 Solutions and Explanation

1. $f(n) = n^6 + 3^n \in \Theta(3^n)$

Explanation: The term $3n$ grows faster than n^6, so the dominant term is 3^n.

2. $f(n) = 2^n + 12 \in \Theta(2^n)$

Explanation: The term 2^n grows faster than the constant 12, so the dominant term is 2^n.

3. $f(n) = 3^n + 2^n \in \Theta(3^n)$

Explanation: The term 3^n grows faster than 2^n, so the dominant term is 3^n.

4. $f(n) = n^n + n \in \Theta(n^n)$

Explanation: The term n^n grows faster than n, so the dominant term is n^n.

Exercise 2 Solutions and Explanation

1. Θ(1):

- Tight O Bound: $O(1)$
- Non-Tight O Bound: $O(n)$
- Tight Ω Bound: Ω(1)
- Non-Tight Ω Bound: None, as 1 is the lowest bound.

2. Θ(√n):

- Tight O Bound: $O(\sqrt{n})$
- Non-Tight O Bound: $O(n)$
- Tight Ω Bound: $\Omega(\sqrt{n})$
- Non-Tight Ω Bound: Ω(1)

3. Θ(n):

- Tight O Bound: $O(n)$
- Non-Tight O Bound: $O(n^3)$
- Tight Ω Bound: Ω(n)
- Non-Tight Ω Bound: Ω(1)

Θ(n^2):

- Tight O Bound: $O(n^2)$
- Non-Tight O Bound: $O(n^3)$
- Tight Ω Bound: Ω(n^2)
- Non-Tight Ω Bound: Ω(1)

Θ(n^3):

- Tight O Bound: $O(n^3)$
- Non-Tight O Bound: $O(n^4)$
- Tight Ω Bound: Ω(n^3)
- Non-Tight Ω Bound: Ω(n^2)

Solving the Exercises

To solve these exercises, I identified the dominant term for each function in Exercise 1. The dominant term is the one that grows the fastest as n increases, determining the asymptotic complexity.

For Exercise 2, I used the definitions of O (upper bound), Ω (lower bound), and Θ (tight bound). A tight bound means the complexity function lies between close upper and lower bounds, while a non-tight bound is less precise, either much larger (for O) or much smaller (for Ω) than the function.

Example for Θ(n):

- For Θ(n), the function grows linearly. A tight O bound is $O(n)$, precisely capturing the growth rate. A non-tight O bound could be $O(n3)$, which overestimates the growth rate. For Ω bounds, Ω(n) is tight, while Ω(1) is non-tight, underestimating the growth rate.

References

Charras, C., & Lecroq, T. (1997). Brute Force algorithm. Retrieved from http://www-igm.univmlv.fr/~lecroq/string/node3.html

Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2006). Algorithms. Retrieved from

https://my.uopeople.edu/pluginfile.php/323000/mod_resource/content/0/Algorithms

%20Textbook.pdf

Iacono, A. (2017). Backtracking explained. Retrieved from

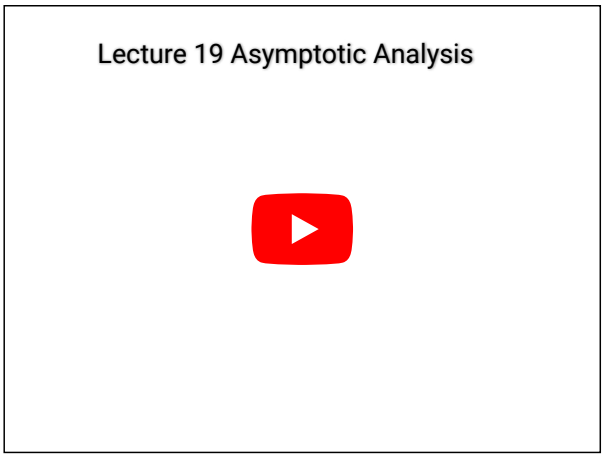
https://medium.com/@andreaiacono/backtracking-explained-7450d6ef9e1a

OpenDSA (2018). CS3 Data Structures & Algorithms. Retrieved from https://opendsaserver.cs.vt.edu/ODSA/Books/CS3/html/

Quora (2013). What are the differences between backtracking and Branch & Bound algorithm techniques? Retrieved from https://www.quora.com/What-are-the-differences-betweenbacktracking-and-Branch-Bound-algorithm-techniques

Shaffer, C. A. (2010). A Practical Introduction to Data Structures and Algorithm Analysis, Third Edition (C++ Edition). Retrieved from http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

Shewchuk, J. (2006). Asymptotic Analysis. Retrieved from



Stack Overflow (2011). What exactly is the brute force algorithm. Retrieved from

https://stackoverflow.com/questions/8103050/what-exactly-is-the-brute-force-algorithm

The University of Arizona (2009). Asymptotic Notation: O(), o(), Ω(), and Θ(). Retrieved from
https://www2.cs.arizona.edu/classes/cs345/summer14/files/bigO.pdf

787 words

Permalink Show parent

Re: Week 1

by Tyler Huffman - Thursday, 27 June 2024, 8:23 AM

Well done on your reply. You explained some complex topics quite simply; your summation of a Branch and Bound type algorithm especially was quick although effective. You made a good use of references in your write as well. Keep up the good work.

43 words

Permalink Show parent

Re: Week 1

by Natalie Tyson - Thursday, 27 June 2024, 8:30 AM

Assignment 1:

I will summarize the branch and bound algorithms and describe what a brute force algorithms are and their differences. We will look at their asymptomatic behaviors and see how it impacts which design we might decide to use.

Branch and Bound Algorithms:

These are used to solve the optimization problems for which we would select the best solution out of all the possible solutions available. We can use this to rule out what would not give us the best possible answers and reduce the solutions and wasted time using solutions that would not be a good fit. This algorithm works using heuristic methods to divide the starting or main problem into several tinier subproblems.

Why should we use the branch and bound algorithms?

- It can handle large problems efficiently
- If you have an exponential number of possible answers then this is better
- Reduces search space by pruning bad branches
- Mainly solves the promising combinatorial optimization problems ignoring inefficient problems

Brute Force Algorithms:

This is a comprehensive search strategy which will explore all possible problems in order to find the answer. This is inefficient for large scale problems because of its in-depth nature, the temporal complexity would make this inefficient for calculating the best solution in the least amount of time. For a straightforward technique that is direct and good for small problems of scale.

Why use brute force algorithms?

-Not specific any specific problem domain

-Great for simple and small problems

-Guarantees the correct solution is found by going through all the possible scenarios

Differences between the branch and bound algorithm vs Brute Force Algorithms:

Brute force	Branch and bound Algorithm
Examines all possible solutions problem by problem	Divies the problem into tinier subproblems and prunes the bad solutions to limit the search space
Suitable for small or simple problems	Better for large or complex optimization problems
Inefficient with larger problem sizes	With pruning mechanisms, they are more efficient at solving complex problems

To summarize, we would want to use Brute force if our problem was simple and small, but it is not efficient for solving the larger problems, in those cases we would prefer to use the Branch and bound algorithm.

Assignment 2:

Exercise 1:

- $f(n) = n6 + 3n$

$n^6 + 3^n$ 2 x 3^n

hence n^6 is greater than 3^n

$f(n) = \Theta (3^n)$

-$f(n) = 2n + 12$

The bigger value here that will correspond will be 2^n, the 12 can be dropped

Hence $2^n + 12 = \Theta (2^{n)}$

-$f(n) = 3^n + 2^n$

the larger value 3^n can be taken from this expression

$= \Theta(3^{n)}$

-$f(n) = n^n + n$

n to the power of itself is going to be a larger number compared to n

hence $\Theta(n^n)$

Exercise 2:

Θ(1)

We have the tight lower Ω bound or Ω(1) since it is equal to Θ (1). Non-tight Ω bound or the Ω(log n) grows faster than the Θ (1).This is considered to be both non-tight and tight since Θ (1) is a tight bound This is a non-tight upper bound since it is the upper bound for √n, since the range for Ω limits us to remaining above 1, a tight bound will be the method to use.

Θ(√n)

This is a non-tight upper bound since O(n) is larger than the Θ(√n). This contains a tight lower bound since this would match Ω(√n). The non-tight lower bound grows faster than the Ω(1).

Θ(n)

These bounds are considered valid even if not very good because they are not near the original complexities. O(n) and Ω(n) will function as tight bounds. The non-tight bounds are $O(n^2)$ or Ω(√n).
Θ(n2)

Tight bounds are $\Omega(n^2)$ and O (n^2). The non-tight bounds are O (n^3) and Ω(n).
Θ(n3)

Tight bounds for O (n^3) and $\Omega(n^3)$. Non-tight bounds, we have O as our upper bound and $O(n^4)$ and Ω (n^2), making these bounds closer to the original complexities than the previous solution.

Citations

- GeeksforGeeks. (2024a, January 18). *Brute Force Approach and its pros and cons.* https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/

- GeeksforGeeks. (2024b, March 19). *Why do we use branch and bound algorithm?* https://www.geeksforgeeks.org/why-do-we-use-branch-and-bound-algorithm/#

- Chapter 3: Algorithm Analysis in A Practical Introduction to Data Structures and Algorithm Analysis by Clifford A. Shaffer. http://people.cs.vt.edu/~shaffer/Book/C++3e20100119.pdf

721 words

Permalink Show parent

Re: Week 1
by Jobert Cadiz - Thursday, 27 June 2024, 9:00 AM

Hi Natalie,
Your answer effectively introduces and distinguishes between Branch and Bound and Brute Force algorithms, highlighting their basic characteristics and differences. While there are issues with clarity and correctness, the answer attempts to apply asymptotic analysis concepts to specific functions, demonstrating an effort to engage with algorithmic complexity. Still, good work on this.
54 words

Permalink Show parent

Re: Week 1

by Tamaneenah Kazeem - Thursday, 27 June 2024, 10:45 AM

Hey Natalie, thanks for sharing your post. The first parts were easy to understand, keep it up. I must say, however, that I struggled with understanding Exercise 2. Regardless, you perfo