Brute-force search

From Wikipedia, the free encyclopedia

In computer science, **brute-force search** or **exhaustive search**, also known as **generate and test**, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

A brute-force algorithm to find the divisors of a natural number n would enumerate all integers from 1 to the square root of n, and check whether each of them divides n without remainder. A brute-force approach for the eight queens puzzle would examine all possible arrangements of 8 pieces on the 64-square chessboard, and, for each arrangement, check whether each (queen) piece can attack any other.

While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions – which in many practical problems tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed.

This is the case, for example, in critical applications where any errors in the algorithm would have very serious consequences; or when using a computer to prove a mathematical theorem. Brute-force search is also useful as a baseline method when benchmarking other algorithms or metaheuristics. Indeed, brute-force search can be viewed as the simplest metaheuristic. Brute force search should not be confused with backtracking, where large sets of solutions can be discarded without being explicitly enumerated (as in the textbook computer solution to the eight queens problem above). The brute-force method for finding an item in a table — namely, check all entries of the latter, sequentially — is called linear search.

Contents

NETWORK-1 EXHIBIT 2001

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Implementing the brute-force search

Basic algorithm

In order to apply brute-force search to a specific class of problems, one must implement four procedures, *first*,*next*, *valid*, and *output*. These procedures should take as a parameter the data P for the particular instance of the problem that is to be solved, and should do the following:

1. *first* (P): generate a first candidate solution for P.
2. *next* (P, c): generate the next candidate for P after the current one c.
3. *valid* (P, c): check whether candidate c is a solution for P.
4. *output* (P, c): use the solution c of P as appropriate to the application.

The *next* procedure must also tell when there are no more candidates for the instance P, after the current one c. A convenient way to do that is to return a "null candidate", some conventional data value Λ that is distinct from any real candidate. Likewise the *first* procedure should return Λ if there are no candidates at all for the instance P. The brute-force method is then expressed by the algorithm

```
c ← first(P)
while' c ≠ Λ do
 if' valid(P,c) then output(P, c)
 c ← next(P,c)
end while
```

For example, when looking for the divisors of an integer n, the instance data P is the number n. The call *first*(n) should return the integer 1 if $n \geq 1$, or Λ otherwise; the call *next*(n,c) should return $c + 1$ if $c < n$, and Λ otherwise; and *valid*(n,c) should return **true** if and only if c is a divisor of n. (In fact, if we choose Λ to be $n + 1$, the tests $n \geq 1$ and $c < n$ are unnecessary.)The brute-force search algorithm above will call *output* for every candidate that is a solution to the given instance P. The algorithm is easily modified to stop after finding the first solution, or a specified number of solutions; or after testing a specified number of candidates, or after spending a given amount of CPU time.

Combinatorial explosion

The main disadvantage of the brute-force method is that, for many real-world problems, the number of natural candidates is prohibitively large. For instance, if we look for the divisors of a number as described above, the number of candidates tested will be the given number n. So if n has sixteen decimal digits, say, the search will require executing at least 10^{15} computer instructions, which will take several days on a typical PC. If n is a random 64-bit natural number, which has about 19 decimal digits on the average, the search will take about 10 years. This steep growth in the number of candidates, as the size of the data increases, occurs in all sorts of problems. For instance, if we are seeking a particular rearrangement of 10 letters, then we have 10! = 3,628,800 candidates to consider, which a typical PC can generate and test in less than one second. However, adding one more letter — which is only a 10% increase in the data size — will multiply the number of candidates by 11 — a 1000% increase. For 20

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

letters, the number of candidates is 20!, which is about 2.4×10^{18} or 2.4 quintillion; and the search will take about 10 years. This unwelcome phenomenon is commonly called the combinatorial explosion, or the curse of dimensionality.

Speeding up brute-force searches

One way to speed up a brute-force algorithm is to reduce the search space, that is, the set of candidate solutions, by using heuristics specific to the problem class. For example, in the eight queens problem the challenge is to place eight queens on a standard chessboard so that no queen attacks any other. Since each queen can be placed in any of the 64 squares, in principle there are $64^8 = 281,474,976,710,656$ possibilities to consider. However, because the queens are all alike, and that no two queens can be placed on the same square, the candidates are all possible ways of choosing of a set of 8 squares from the set all 64 squares; which means 64 choose 8 = 64!/56!/8! = 4,426,165,368 candidate solutions — about 1/60,000 of the previous estimate. Further, no arrangement with two queens on the same row or the same column can be a solution. Therefore, we can further restrict the set of candidates to those arrangements.

As this example shows, a little bit of analysis will often lead to dramatic reductions in the number of candidate solutions, and may turn an intractable problem into a trivial one.

In some cases, the analysis may reduce the candidates to the set of all valid solutions; that is, it may yield an algorithm that directly enumerates all the desired solutions (or finds one solution, as appropriate), without wasting time with tests and the generation of invalid candidates. For example, for the problem "find all integers between 1 and 1,000,000 that are evenly divisible by 417" a naive brute-force solution would generate all integers in the range, testing each of them for divisibility. However, that problem can be solved much more efficiently by starting with 417 and repeatedly adding 417 until the number exceeds 1,000,000 — which takes only 2398 (= 1,000,000 ÷ 417) steps, and no tests.

Reordering the search space

In applications that require only one solution, rather than all solutions, the expected running time of a brute force search will often depend on the order in which the candidates are tested. As a general rule, one should test the most promising candidates first. For example, when searching for a proper divisor of a random number n, it is better to enumerate the candidate divisors in increasing order, from 2 to n - 1, than the other way around — because the probability that n is divisible by c is $1/c$. Moreover, the probability of a candidate being valid is often affected by the previous failed trials. For example, consider the problem of finding a **1** bit in a given 1000-bit string P. In this case, the candidate solutions are the indices 1 to 1000, and a candidate c is valid if $P[c] = \mathbf{1}$. Now, suppose that the first bit of P is equally likely to be **0** or **1**, but each bit thereafter is equal to the previous one with 90% probability. If the candidates are enumerated in increasing order, 1 to 1000, the number t of candidates examined before success will be about 6, on the average. On the other hand, if the candidates are enumerated in the order 1,11,21,31...991,2,12,22,32 etc., the expected value of t will be only a little more than 2.More generally, the search space should be enumerated in such a way that the next candidate is most likely to be valid, *given that the previous trials were not*. So if the valid solutions are likely to be "clustered" in

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

some sense, then each new candidate should be as far as possible from the previous ones, in that same sense. The converse holds, of course, if the solutions are likely to be spread out more uniformly than expected by chance.

Alternatives to brute-force search

There are many other search methods, or metaheuristics, which are designed to take advantage of various kinds of partial knowledge one may have about the solution. Heuristics can also be used to make an early cutoff of parts of the search. One example of this is the minimax principle for searching game trees, that eliminates many subtrees at an early stage in the search. In certain fields, such as language parsing, techniques such as chart parsing can exploit constraints in the problem to reduce an exponential complexity problem into a polynomial complexity problem. In many cases, such as in Constraint Satisfaction Problems, one can dramatically reduce the search space by means of Constraint propagation, that is efficiently implemented in Constraint programming languages.The search space for problems can also be reduced by replacing the full problem with a simplified version. For example, in computer chess, rather than computing the full minimax tree of all possible moves for the remainder of the game, a more limited tree of minimax possibilities is computed, with the tree being pruned at a certain number of moves, and the remainder of the tree being approximated by a static evaluation function.

In cryptography

In cryptography, a *brute-force attack* involves systematically checking all possible keys until the correct key is found. This strategy can in theory be used against any encrypted data[1] (except a one-time pad) by an attacker who is unable to take advantage of any weakness in an encryption system that would otherwise make his or her task easier.

The key length used in the encryption determines the practical feasibility of performing a brute force attack, with longer keys exponentially more difficult to crack than shorter ones. Brute force attacks can be made less effective by obfuscating the data to be encoded, something that makes it more difficult for an attacker to recognise when he has cracked the code. One of the measures of the strength of an encryption system is how long it would theoretically take an attacker to mount a successful brute force attack against it.

References

1. Christof Paar, Jan Pelzl, Bart Preneel (2010). *Understanding Cryptography: A Textbook for Students and Practitioners* (http://www.crypto-textbook.com). Springer. p. 7. ISBN 3-642-04100-0.

See also

- How basic bruteforce tool can be used to crack command line application - Tool (http://www.worldofhacker.com/2013/09/basic-idea-of-creating-password.html)

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

- Brute force attack
- Big O notation

Retrieved from "http://en.wikipedia.org/w/index.php?title=Brute-force_search&oldid=636308630"

Categories: Search algorithms

- This page was last modified on 2 December 2014, at 12:55.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.