<https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life>

**Which are the 10 algorithms every computer science student must implement at least once in life?**

UpdateCancel

Answer Wiki

14 Answers

[Matthew Mirman](https://www.quora.com/profile/Matthew-Mirman)

[Matthew Mirman](https://www.quora.com/profile/Matthew-Mirman), CMU - SCS for undergrad & masters

44.5k Views

**Note**: nobody is holding any students down and forcing them to implement any algorithms, but if somebody was, I'd imagine it should be these.  
  
**Note**: some of these are more useful than others, and some I included because they make you learn so many more algorithms

* In place merge sort & heap sort with linear time heap building.
* Breadth first traversals of graphs & trees.
* Dijkstra's algorithm w/ fibonacci heap
* Kruskal's algorithm w/ union find by rank
* deterministic quick select
* polymorphic type inference
* 2D voronoi diagram generation
* 2D convex hull generation
* Knapsack & related DP problems
* [Ford–Fulkerson algorithm](http://en.wikipedia.org/wiki/Ford%E2%80%93Fulkerson_algorithm) for max flow.

[Updated Nov 27, 2012](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Matthew-Mirman) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=VOWBROMv0yh)

**Related Questions**

[More Answers Below](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life#MoreAnswers)

* [What are algorithmic problems that every cse student must solve?](https://www.quora.com/What-are-algorithmic-problems-that-every-cse-student-must-solve)
* [What are the top Stack Overflow questions every computer science student must see?](https://www.quora.com/What-are-the-top-Stack-Overflow-questions-every-computer-science-student-must-see)
* [What are the 5 most important CS courses that every computer science student must take?](https://www.quora.com/What-are-the-5-most-important-CS-courses-that-every-computer-science-student-must-take)
* [Have there been any new brilliant computer science algorithms in last 10 years?](https://www.quora.com/Have-there-been-any-new-brilliant-computer-science-algorithms-in-last-10-years)
* [What are the 10 artificial intelligence (AI) algorithms that every computer science student must be aware of?](https://www.quora.com/unanswered/What-are-the-10-artificial-intelligence-AI-algorithms-that-every-computer-science-student-must-be-aware-of)

[Peteris Erins](https://www.quora.com/profile/Peteris-Erins)

[Peteris Erins](https://www.quora.com/profile/Peteris-Erins), Data science intern at Twitter, Cambridge graduate

61.9k Views • Upvoted by

[Stan Hanks](https://www.quora.com/profile/Stan-Hanks), building "the Internet" since 1981, now building web-scale companies,

[Edwin Khoo](https://www.quora.com/profile/Edwin-Khoo), Graduate student,

[Igor Markov](https://www.quora.com/profile/Igor-Markov), EECS Prof at Michigan - currently at Google

Coming from an intensive programming contest background, having read Introduction to Algorithms, etc., but then moving on to doing real-life projects, I find that some of the more interesting and powerful algorithms in a problem solving setting find little use in the industry.  
  
I'm thus going to present algorithms that carry over to engineering, not just research and TopCoder. Another goal was for these algorithms to be useful and widely applicable hammers in their own right, not just as an idea. The benefits of most "algorithms" will be hard to explain them to someone who has not yet been introduced to them, but I tried my best of giving a hint for their role.

1. **Naive Bayes.** Classification is a natural first step to machine learning and the Naive Bayes classifier combines that with the trending concept of Bayesian inference. Despite its simplicity, can address key problems like spam classification.
2. **Map-Reduce.** A simple and visual way of leveraging multiple computers to solve a single task. More on its different uses here: [MapReduce Patterns, Algorithms, and Use Cases](http://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/).
3. **Hill Climbing.** Introduces optimization over a non-exhaustible domain and leads to a whole array of powerful methods like gradient-descent, simulated annealing. Optimization allows us to do decision making, it's important.
4. **Binary search.** Introduces logarithmic complexity and how the structure of data can change complexity characteristics of operations.
5. **Exponentiation (powers) by halving.** Great example of a recursive attack. You would be surprised how many problems can be described as exponentiation (for an associative operation) of *something*: a number, a matrix and more. Again it's magical that associativity of an operation allows you to turn a linear factor into a logarithmic one.
6. **Floyd-Warshall.** Graphs as matrices, shortest paths, negative cycles, dynamic programming in **five** lines of code.
7. **Inductive inference.** [Inductive inference](http://en.wikipedia.org/wiki/Inductive_inference). This idea on how to find programs that do certain things without having a human write them I believe is the key to the future of AI and simulating and understanding the human mind.
8. **Markov Chain Monte Carlo.** Approximate integral, expected value computation based on a conditional specification of the probability distribution. This gives us a general way of doing data analysis.
9. **Shor's algorithm.** Non-trvial example of a quantum algorithm that shows a significant complexity improvement. Reveals how quantum computing works.
10. **PageRank.** How to iteratively solve for tightly related variables to extract metrics on graphs.

[Written Jan 26, 2013](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Peteris-Erins) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=FAXqDfSrM1H)

Anonymous

Anonymous

104.9k Views • Upvoted by

[Stan Hanks](https://www.quora.com/profile/Stan-Hanks), building "the Internet" since 1981, now building web-scale companies

More important than algorithms(just problems #$!%), the techniques/concepts residing at the base of such algorithms is more important.  
  
There are broadly 4 ways in which classification of algorithms can be done.

1. ***Classification by purpose***

Each algorithm has a goal, for example, the purpose of the Quick Sort algorithm is to sort data in ascending or descending order. But the number of goals is infinite, and we have to group them by kind of purposes.  
  
2.  ***Classification by implementation***

* **Recursive** or **iterative**  
  A recursive algorithm is one that calls itself repeatedly until a certain condition matches. It is a method common to functional programming.   
  For example, the towers of hanoi problem is well understood in recursive implementation. Every recursive version has an iterative equivalent iterative, and vice versa.
* **Logical** or **procedural**  
  An algorithm may be viewed as controlled logical deduction.   
  A logic component expresses the axioms which may be used in the computation and a control component determines the way in which deduction is applied to the axioms.
* **Serial** or **parallel** Algorithms are usually discussed with the assumption that computers execute one instruction of an algorithm at a time. This is a serial algorithm, as opposed to parallel algorithms, which take advantage of computer architectures to process several instructions at once. **Sorting algorithms** can be parallelized efficiently.
* **Deterministic** or **non-deterministic**  
  Deterministic algorithms solve the problem with a predefined process whereas non-deterministic algorithm must perform guesses of best solution at each step through the use of heuristics.

3.   ***Classification by design paradigm***

* **Divide and conquer**  
  A divide and conquer algorithm repeatedly reduces an instance of a problem to one or more smaller instances of the same problem (usually recursively), until the instances are small enough to solve easily. One such example of divide and conquer is merge sorting. The binary search algorithm is an example of a variant of divide and conquer called **decrease and conquer algorithm**, that solves an identical subproblem and uses the solution of this subproblem to solve the bigger problem.
* **Dynamic programming**  
  The shortest path in a weighted graph can be found by using the shortest path to the goal from all adjacent vertices.   
  When the optimal solution to a problem can be constructed from optimal solutions to subproblems, using **dynamic programming avoids recomputing solutions** that have already been computed.   
  - The main difference with the "divide and conquer" approach is, subproblems are independent in divide and conquer, where as the overlap of subproblems occur in dynamic programming.   
  - Dynamic programming and **memoization** go together. The difference with straightforward recursion is in caching or memoization of recursive calls. Where subproblems are independent, this is useless. By using memoization or maintaining a table of subproblems already solved, dynamic programming reduces the exponential nature of many problems to polynomial complexity.
* **The greedy method**  
  A greedy algorithm is similar to a dynamic programming algorithm, but the difference is that solutions to the subproblems do not have to be known at each stage. Instead a "greedy" choice can be made of what looks the best solution for the moment.   
  The most popular greedy algorithm is finding the minimal spanning tree as given by **Kruskal**.
* **Linear programming**  
  The problem is expressed as a set of linear inequalities and then an attempt is made to maximize or minimize the inputs. This can solve many problems such as the maximum flow for directed graphs, notably by using the simplex algorithm.   
  A complex variant of linear programming is called **integer programming**, where the solution space is restricted to all integers.
* **Reduction** also called **transform and conquer**  
  Solve a problem by transforming it into another problem. A simple example:**finding the median** in an unsorted list is first translating this problem into sorting problem and finding the middle element in sorted list. The main goal of reduction is finding the simplest transformation possible.
* **Using graphs**  
  Many problems, such as playing chess, can be modeled as problems on graphs. A graph exploration algorithms are used.   
  This category also includes the search algorithms and backtracking.
* **The probabilistic and heuristic paradigm**

1. **Probabilistic**   
   Those that make some choices randomly.
2. **Genetic**   
   Attempt to find solutions to problems by mimicking biological evolutionary processes, with a cycle of random mutations yielding successive generations of "solutions". Thus, they emulate reproduction and "survival of the fittest".
3. **Heuristic**   
   Whose general purpose is not to find an optimal solution, but an approximate solution where the time or resources to find a perfect solution are not practical.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
You can look at [Algorithms Repository](http://algorithm.daqwest.com/)  
  
1. **Searching and sorting algorithms** -   
Sorting algorithms include [Quicksort](http://algorithm.daqwest.com/search?search=Quicksort) [, Merge sort](http://algorithm.daqwest.com/search?search=Merge+sort)[, Heapsort](http://algorithm.daqwest.com/search?search=Heapsort)[, Bubble sort](http://algorithm.daqwest.com/search?search=Bubble+sort)[,](http://algorithm.daqwest.com/search?search=Selection+sort)[Insertion sort](http://algorithm.daqwest.com/search?search=Insertion+sort)[, Radix sort](http://algorithm.daqwest.com/search?search=Radix+sort). Other imp soting algorithms are [Topological sort](http://algorithm.daqwest.com/?search=Topological+sort)[, Counting sort](http://algorithm.daqwest.com/?search=Counting+sort)[, Shell sort](http://algorithm.daqwest.com/?search=Shell+sort)  
A comprehensive list can be found [here](http://algorithm.daqwest.com/#Sequence).  
Important searching algorithms include [breadth](http://algorithm.daqwest.com/search?search=Breadth-first+search)/ [depth](http://algorithm.daqwest.com/search?search=Depth-first+search) first  search, binary search etc.  
  
2. **Dynamic Programming** -- To name a few DP problems, [Longest Common Subsequence problem](http://algorithm.daqwest.com/search?search=Longest+common+subsequence+problem), [Knapsack](http://algorithm.daqwest.com/search?search=Knapsack+problem), [travelling salesman problem](http://algorithm.daqwest.com/search?search=Traveling+salesman+problem) etc. A list of dynamic  programming algorithms can be found [here](http://algorithm.daqwest.com/#Dynamic).  
  
3. **Graph algorithms** -- Important graph algorithms are [Dijkstra](http://algorithm.daqwest.com/search?search=Dijkstra&#x27;s%20algorithm), [Prim,](http://algorithm.daqwest.com/search?search=Prim&#x27;s%20algorithm) [Kruskal](http://algorithm.daqwest.com/search?search=Kruskal&#x27;s%20algorithm), [Bellman-Ford](http://algorithm.daqwest.com/search?search=Bellman%C3%A2%C2%80%C2%93Ford+algorithm). A comprehensive list can be found [here](http://algorithm.daqwest.com/#Graph).  
  
  
Good luck !!!

[Updated Feb 22, 2014](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answers/1867566) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=3nQn5CGGkAm)

[Alket Cecaj](https://www.quora.com/profile/Alket-Cecaj)

[Alket Cecaj](https://www.quora.com/profile/Alket-Cecaj), Java developer

4k Views

There are many depending on the purpose of application .Here are some of the most famous :  
  
1- clustering :is the problem of  grouping the individuals in a population together by their similarity of attributes. A very famous clustering algorith is for example k-means,  
  
2- classification algorithms. Classification tries  to predict, for each individual  
in a population, which of a (small) set of classes this individual belongs to. A classification task, given a new individual, determines which class that individual belongs to. It also mya assign a probability to this association. An example is KNN (or k neares neighbour )  
  
3-dimension reduction algorithm for reducing dimensions of a data set .It tries to take a large set of data and replace it with a smaller set of data that contains much of the important information in the larger set. For example you can use the reduced data set for undestanding it better and visualize it in 2D dimensions.   
  
4- principal components analysis for finding structure in your dataset,   
  
5- collaborative filtering for building recommendation systems. It is a problem of similarity matching. For example finding people who are similar to you in  
terms of the products they have liked or have purchased or finding products that are similar in terms .   
  
6- association rules or co-occurrence grouping for market basket analysis.  A common question in this case is : what items are commonly purchased together?   
For example, analyzing purchase records from a supermarket may uncover that  
beer is purchased together with chips  frequently .  
  
7- regression method for predicting the value of a certain value. For example : How much will a given customer use a certain service? The quantity to be predicted here is service usage, and a model could be generated by  
looking at other, similar individuals in the population and their historical usage.  
  
8- LDA algorithm for sentiment analysis and text mining but also for many other applications   
  
9- Dijkstra's Shortest Path for finding the shortest way from a node to another in a graph.  
  
10- Link prediction to predict connections between data items, by  
suggesting that a link should exist, and estimating the strength of the  
link. For example in social networking it tries to guess if   you and John  
share 10 friends, maybe you and John could be friens also in real life.

[Written Feb 20](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Alket-Cecaj) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=FjFvyFaApod)

[Vladimir Novakovski](https://www.quora.com/profile/Vladimir-Novakovski)

[Vladimir Novakovski](https://www.quora.com/profile/Vladimir-Novakovski), IOI 2001, ACM World Finals 2003 and 2004

13.1k Views •

[Vladimir](https://www.quora.com/profile/Vladimir-Novakovski) has [60+ answers](https://www.quora.com/profile/Vladimir-Novakovski/answers/Computer-Programming) in

[Computer Programming](https://www.quora.com/topic/Computer-Programming).

This excludes some important parts of CS, but at least half of the 10 should probably come from this list: [What were the "16 standard algorithms" that Neal Wu's coaches "drilled into his brain" in preparation for the International Olympiad in Informatics?](https://www.quora.com/What-were-the-16-standard-algorithms-that-Neal-Wus-coaches-drilled-into-his-brain-in-preparation-for-the-International-Olympiad-in-Informatics)

[Written Nov 19, 2012](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Vladimir-Novakovski) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=4ohbSvviKkU)

[Sachin Gupta](https://www.quora.com/profile/Sachin-Gupta-6)

[Sachin Gupta](https://www.quora.com/profile/Sachin-Gupta-6), Programming for some years now :)

19.2k Views • Upvoted by

[Stan Hanks](https://www.quora.com/profile/Stan-Hanks), building "the Internet" since 1981, now building web-scale companies

Don't have a list of 10 algorithms, but 7 algorithms that every programmer should know. The first 2 in the list are Sorting and Searching algorithms, which in themselves have many algorithms that should be known, so the list actually goes beyond just 10.

1. Sorting Algorithms
2. Searching Algorithms
3. Hashing
4. Dynamic Programming
5. Binary Exponentiation
6. String matching and parsing
7. Primality Testing

These topics have been taken from this blog post ([Top 7 algorithms and DS every programmer should know](http://blog.hackerearth.com/2015/05/top-7-algorithms-and-data-structures-every-programmer-should-know-about.html)) that covers all of these topics in detail.

[Written Sep 27](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Sachin-Gupta-6) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=bvavO6QQ5w4)

[Jan Christian Meyer](https://www.quora.com/profile/Jan-Christian-Meyer)

[Jan Christian Meyer](https://www.quora.com/profile/Jan-Christian-Meyer), Ph.D. in Computer Science

14.5k Views •

[Jan](https://www.quora.com/profile/Jan-Christian-Meyer) has [400+ answers](https://www.quora.com/profile/Jan-Christian-Meyer/answers/Computer-Programming) in

[Computer Programming](https://www.quora.com/topic/Computer-Programming).

I don't have a list of 10, and certainly not one better than some already suggested, but I have an addendum which I think is very important. I think it is important enough that if the number must be 10, something else will have to go, but I won't try to pick what.  
  
By any given name (REP-Loop, universal Turing machine, code generator...), every computer science student must at least once implement an algorithm which takes other algorithms as input. Not necessarily a complete programming language, but at least a description of behavior for a more general engine. The factoid that "programs are data" is covered in a myriad of ways through any education track worth its salt, but I believe you have to write a program which treats programs as data to fathom the consequences.  
  
A number of times I've seen that experience radically transform how computer science students approach the entire craft, once they reach the enlightening moment. In practical terms, developing the skill of writing programs which write programs for you enables you to explore large design spaces faster than anyone can by hand.  
  
This reflexive capability of programs is at the heart of the deepest results of computer science, and I think it the most profound mode of thinking we have to offer.

[Written Dec 23, 2012](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Jan-Christian-Meyer) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=S33TNDROax)

[Kumar Ankit](https://www.quora.com/profile/Kumar-Ankit-1)

[Kumar Ankit](https://www.quora.com/profile/Kumar-Ankit-1), Programmer, Learner

15.2k Views

Do it for strengthening your concepts or for fun, the following 10 algorithms are the ones I will go with when it comes to implementation.  
  
**1. *Discrete Fourier Transform:*** Implement it to multiply two polynomials with degree bound *n* in O(n log n), which when performed by any other method takes time of order O(n^2). This is a classic amalgamation of mathematics with divide-and-conquer.  
  
**2. *Sieve of Eratosthenes:*** It's the simulation of an age-old efficient method proposed by Eratosthenes to enumerate primes in a given range. The enumeration of primes is a backbone of many programs especially in cryptography.  
  
**3. *Iterative Merge Sort:*** Although Merge-sort has a simple-to-implement and easy-to-comprehend recursive version, it's iterative implementation reveals a great deal about its internals. Also it is somewhat similar to the concept used in parallel prefix computation. So you kill two birds with a stone.  
  
**4. *The Knuth-Morris-Pratt Algorithm:*** This algorithm primarily aimed at checking whether a sub-string occurs in a given text in linear time, can be extended to a variety of string problems with simple modifications. Also, it's a must-know for the programming interviews.   
  
**5. *Kruskal's Algorithm for MST:***  Implementation of Kruskal's using Disjoint Set    Data structure to find the Minimum spanning forest of a given graph. Do remember to implement union by rank and path compression.  
  
**6.** ***Genetic Algorithms:*** Genetic algorithms are mostly used to find near optimal solution to an NP-complete problem. There is plethora of problems that can be solved using GA's. Try implementing the the most famous one - The Travelling Salesman Problem.  
  
**7.** ***Nth Fibonacci number in O(log n):*** Implementation of this will not only empower you with a tool to calculate n-th fibonacci number in logarithmic time, but will also provide you with the basics how the linear-recurrences can be solved using Matrix exponentiation.  
  
**8.** ***Red Black Trees:*** Red Black Trees are balances version of Binary Search Trees. They are widely used in the Standard Template Library of C++ for implementation of map and set as they support insertion and deletion in O(log n).  
  
**9. *Ford-Fulkerson Algorithm:*** Ford-Fulkerson is a classic algorithm for find Maximal-flow in a network with integer edge capacities. It's proof of correctness is sheer beauty and its implementation is an appreciation to what you call intuition. **PS:** It isone of my personal favorites.  
  
**10. *Hopcroft Karp Algorithm:*** This algorithm finds a maximal cardinality matching in a Bipartite graph. It uses a deries of BFS-followed-by-DFS steps to continueously increase the cardinality of the matching till the maximum is reached.  
  
**PS:** I am yet to implement #1 in my list i.e the DFT :P

[Written Jan 26, 2013](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Kumar-Ankit-1) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=RcqXXNV1Iki)

[Jitender Singh](https://www.quora.com/profile/Jitender-Singh-102)

[Jitender Singh](https://www.quora.com/profile/Jitender-Singh-102)

3.7k Views

I found this article and i believe it is a must read for every programmers by Kaushik MV - [How do I Learn to Code ?](http://theshayna.com/how-do-i-learn-to-code/)  
  
The top 10 are -

1. Dynamic Programming
2. Divide and Conquer
3. Stack and Queue
4. Floyd or Warshall's algorithm
5. Kruskal's or Prim's algorithm
6. Dijkstra's algorithm
7. Min, Max, BFS-DFS
8. Binary Search
9. Sorting Techniques
10. Greedy algorithm

You can read and understand them completely and thoroughly from

[theshayna.com](http://theshayna.com/data-structures-and-algorithms-tutorials/" \t "_blank)

[Data Structures and Algorithms Tutorials](http://theshayna.com/data-structures-and-algorithms-tutorials/" \t "_blank).  
  
I found this page which proved to be very beneficial to me.  
Wish you luck.

[Written May 19, 2015](https://www.quora.com/Which-are-the-10-algorithms-every-computer-science-student-must-implement-at-least-once-in-life/answer/Jitender-Singh-102) • [View Upvotes](https://www.quora.com/api/mobile_expanded_voter_list?type=answer&key=1VRGdh8uAtK)