What are the lesser known but useful data structures?

There are some data structures around that are really useful but are unknown to most programmers. Which ones are they?

Everybody knows about linked lists, binary trees, and hashes, but what about Skip lists and Bloom filters for example. I would like to know more data structures that are not so common, but are worth knowing because they rely on great ideas and enrich a programmer's tool box.

PS: I am also interested in techniques like Dancing links which make clever use of properties of a common data structure.

EDIT: Please try to *include links* to pages describing the data structures in more detail. Also, try to add a couple of words on *why* a data structure is cool (as Jonas Kölker already pointed out). Also, try to provide ***one data-structure per answer***. This will allow the better data structures to float to the top based on their votes alone.

language-agnostic data-structures computer-science

edited Feb 13 '12 at 0:15

community wiki
12 revs, 7 users 31%
f3lix

locked by Robert Harvey ♦ Mar 15 '12 at 18:42

This question exists because it has historical significance, but **it is not considered a good, on-topic question for this site**, so please do not use it as evidence that you can ask similar questions here. This question and its answers are frozen and cannot be changed. More info: help center.

16 en.wikipedia.org/wiki/List_of_data_structures + en.wikipedia.org/wiki/Binary_decision_diagram – Fanatic23 Jul 10 '10 at 18:04

83 Answers

1 2 3 next

Tries, also known as prefix-trees or crit-bit trees, have existed for over 40 years but are still relatively unknown. A very cool use of tries is described in "TRASH - A dynamic LC-trie and hash data structure", which combines a trie with a hash function.

edited Oct 5 '10 at 15:34

community wiki
2 revs, 2 users 91%
David Phillips

12 very commonly used by spell-checkers – Steven A. Lowe Feb 1 '09 at 17:32

18 Since no SO question, regardless of topic, is complete without someone mentioning jQuery.... John Resig, creator of jQuery, has an interesting data structure series of posts where he looks at various trie implementations among others: ejohn.org/blog/revised-javascript-dictionary-search – Oskar Austegard Mar 24 '11 at 20:18

4 "very commonly used by spell-checkers" - it's at least funny to see my spell checker claim not to know tries. – dascandy Jul 6 '11 at 17:38

Bloom filter: Bit array of *m* bits, initially all set to 0.

To add an item you run it through *k* hash functions that will give you *k* indices in the array which you then set to 1.

To check if an item is in the set, compute the *k* indices and check if they are all set to 1.

Of course, this gives some probability of false-positives (according to wikipedia it's about 0.61^(m/n) where n is the number of inserted items). False-negatives are not possible.

Removing an item is impossible, but you can implement *counting bloom filter*, represented by array of ints and increment/decrement.

edited Feb 18 '09 at 20:01

community wiki
2 revs, 2 users 95%
albwq

20 You forget to mention their use with dictionaries :) You can squeeze a full dictionary into a bloom filter with about 512k, like a hashtable without the values – Chris S Mar 24 '09 at 21:36

8 Google cites the use of Bloom filters in there implementation of BigTable. – Brian Gianforcaro May 22 '10 at 23:33

16 @FreshCode It actually lets you cheaply test for the *absence* of an element in the set since you can get false positives but never false negatives – Tom Savage May 24 '10 at 17:19

26 @FreshCode As @Tom Savage said, it's more useful when checking for negatives. For example, you can use it as a fast and small (in terms of memory usage) spell checker. Add all of the words to it and then try to look up words the user enters. If you get a negative it means it's misspelled. Then you can run some more expensive check to find closest matches and offer corrections. – lacop May 25 '10 at 20:06

5 @abhin4v: Bloom filters are often used when most requests are likely to return an answer of "no" (such as the case here), meaning that the small number of "yes" answers can be checked with a slower exact test. This still results in a big reduction in the *average* query response time. Don't know if Chrome's Safe Browsing does that, but that would be my guess. – j_random_hacker Jun 9 '10 at 14:32

Rope: It's a string that allows for cheap prepends, substrings, middle insertions and appends. I've really only had use for it once, but no other structure would have sufficed. Regular strings and arrays prepends were just far too expensive for what we needed to do, and reversing everthing was out of the question.

edited Feb 19 '09 at 19:09

community wiki
2 revs
Patrick

15 There's an implementation in the SGI STL (1998): sgi.com/tech/stl/Rope.html – quark Feb 18 '09 at 21:17

2 Without knowing what is was called I recently wrote something very similar to this for Java - performance has been excellent: code.google.com/p/mikeralib/source/browse/trunk/Mikera/src/… – mikera May 23 '10 at 22:01

6 Mikera's link is stale, here's the current. – aptwebapps Mar 24 '11 at 15:04

Skip lists are pretty neat.

> Wikipedia
> A skip list is a probabilistic data structure, based on multiple parallel, sorted linked lists, with efficiency comparable to a binary search tree (order log n average time for most operations).

They can be used as an alternative to balanced trees (using probalistic balancing rather than strict enforcement of balancing). They are easy to implement and faster than say, a red-black tree. I think they should be in every good programmers toolchest.
If you want to get an in-depth introduction to skip-lists here is a link to a video of MIT's Introduction to Algorithms lecture on them.

Also, here is a Java applet demonstrating Skip Lists visually.

edited Feb 22 '09 at 4:15

community wiki
3 revs
Simucal

2 Redis uses skip lists to implement "Sorted Sets". – antirez Mar 24 '11 at 12:11

Spatial Indices, in particular R-trees and KD-trees, store spatial data efficiently. They are good for geographical map coordinate data and VLSI place and route algorithms, and sometimes for nearest-neighbor search.

Bit Arrays store individual bits compactly and allow fast bit operations.

answered Feb 1 '09 at 12:23 community wiki Yuval F

6 Spatial indices are also useful for N-body simulations involving long-range forces like gravity. – Justin Peel May 25 '10 at 16:13

Zippers - derivatives of data structures that modify the structure to have a natural notion of 'cursor' -- current location. These are really useful as they guarantee indicies cannot be out of bound -- used, e.g. in the xmonad window manager to track which window has focused.

Amazingly, you can derive them by applying techniques from calculus to the type of the original data structure!

edited Jul 23 '10 at 3:01 community wiki 2 revs, 2 users 93% Don Stewart

2 this is only useful in functional programming (in imperative languages you just keep a pointer or an index). Also tbh I still don't get how Zippers really work. – Stefan Monov May 24 '10 at 14:23

4 @Stefan the point is that you don't need to keep a separate index or pointer now. – Don Stewart May 24 '10 at 16:03

Here are a few:

- Suffix tries. Useful for almost all kinds of string searching (http://en.wikipedia.org/wiki/Suffix_trie#Functionality). See also suffix arrays; they're not quite as fast as suffix trees, but a whole lot smaller.
- Splay trees (as mentioned above). The reason they are cool is threefold:
  - They are small: you only need the left and right pointers like you do in any binary tree (no node-color or size information needs to be stored)
  - They are (comparatively) very easy to implement
  - They offer optimal amortized complexity for a whole host of "measurement criteria" (log n lookup time being the one everybody knows). See http://en.wikipedia.org/wiki/Splay_tree#Performance_theorems
- Heap-ordered search trees: you store a bunch of (key, prio) pairs in a tree, such that it's a search tree with respect to the keys, and heap-ordered with respect to the priorities. One can show that such a tree has a unique shape (and it's not always fully packed up-and-to-the-left). With random priorities, it gives you expected O(log n) search time, IIRC.
- A niche one is adjacency lists for undirected planar graphs with O(1) neighbour queries. This is not so much a data structure as a particular way to organize an existing data structure. Here's how you do it: every planar graph has a node with degree at most 6. Pick such a node, put its neighbors in its neighbor list, remove it from the graph, and recurse until the graph is empty. When given a pair (u, v), look for u in v's neighbor list and for v in u's neighbor list. Both have size at most 6, so this is O(1).

By the above algorithm, if u and v are neighbors, you won't have both u in v's list and v in u's list. If you need this, just add each node's missing neighbors to that node's neighbor list, but store how much of the neighbor list you need to look through for fast lookup.

answered Feb 1 '09 at 12:12 community wiki Jonas Kölker

79 You couldn't just list one per answer, could ya? Makes it easier for the single, good data structures to float to the top. – KingNestor Feb 18 '09 at 20:11

6 I could, but I'm still not /quite/ getting the hang of this weird wiki/forum hybrid. I'm not gonna edit right now (eat-sleep-rinse-repeat), and I'm probably gonna forget to do it later ;) – Jonas Kölker Feb 18 '09 at 23:10

1 "The Heap ordered search tree is called a treap." -- In the definition I've heard, IIRC, a treap is a heap-ordered search tree with *random* priorities. You could choose other priorities, depending on the application... – Jonas Kölker Feb 19 '09 at 12:32

2 A suffix *trie* is almost but not quite the same as the much cooler suffix *tree*, which has strings and not individual letters on its edges and can be built in linear time(!). Also despite being asymptotically slower, in practice suffix arrays are often much faster than suffix trees for many tasks because of their smaller size and fewer pointer indirections. Love the O(1) planar graph lookup BTW! – j_random_hacker Jun 9 '10 at 14:40

1 @Edward Kmett: I have in fact read that paper, it was quite a breakthrough in suffix array *construction*. (Although it was already known that linear time construction was possible by going "via" a suffix tree, this was the 1st undeniably practical "direct" algorithm.) But some operations outside of construction are still asymptotically slower on a suffix array unless a LCA table is also built. That can also be done in O(n), but you lose the size and locality benefits of the pure suffix array by doing so. – j_random_hacker Jul 26 '10 at 0:39

I think lock-free alternatives to standard data structures i.e lock-free queue, stack and list are much overlooked.
They are increasingly relevant as concurrency becomes a higher priority and are much more admirable goal than using Mutexes or locks to handle concurrent read/writes.

Here's some links
http://www.cl.cam.ac.uk/research/srg/netos/lock-free/
http://www.research.ibm.com/people/m/michael/podc-1996.pdf [Links to PDF]
http://www.boyet.com/Articles/LockfreeStack.html

Mike Acton's (often provocative) blog has some excellent articles on lock-free design and approaches

edited May 26 '10 at 17:39

community wiki
2 revs, 2 users 92%
zebrabox

I think Disjoint Set is pretty nifty for cases when you need to divide a bunch of items into distinct sets and query membership. Good implementation of the Union and Find operations result in amortized costs that are effectively constant (inverse of Ackermnan's Function, if I recall my data structures class correctly).

answered Feb 18 '09 at 20:17

community wiki
Dana

8 This is also called the *"union-find data structure."* I was in awe when I first learned about this clever data structure in algorithms class... – BlueRaja - Danny Pflughoeft Jan 28 '10 at 19:54

4 I used a Disjoint Set for my Dungeon generator, to ensure all rooms are reachable by passages :) – goldenratio Mar 24 '11 at 21:56

Fibonacci heaps

They're used in some of the fastest known algorithms (asymptotically) for a lot of graph-related problems, such as the Shortest Path problem. Dijkstra's algorithm runs in O(E log V) time with standard binary heaps; using Fibonacci heaps improves that to O(E + V log V), which is a huge speedup for dense graphs. Unfortunately, though, they have a high constant factor, often making them impractical in practice.

edited Jun 19 '11 at 17:22

community wiki
3 revs, 2 users 80%
Adam Rosenfield

Anyone with experience in 3D rendering should be familiar with BSP trees. Generally, it's the method by structuring a 3D scene to be manageable for rendering knowing the camera coordinates and bearing.

> Binary space partitioning (BSP) is a method for recursively subdividing a space into convex sets by hyperplanes. This subdivision gives rise to a representation of the scene by means of a tree data structure known as a BSP tree.
>
> In other words, it is a method of breaking up intricately shaped polygons into convex sets, or smaller polygons consisting entirely of non-reflex angles (angles smaller than 180°). For a more general description of space partitioning, see space partitioning.
>
> Originally, this approach was proposed in 3D computer graphics to increase the rendering efficiency. Some other applications include performing geometrical operations with shapes (constructive solid geometry) in CAD, collision detection in robotics and 3D computer games, and other computer applications that involve handling of complex spatial scenes.

answered Feb 18 '09 at 20:26

community wiki
spoulson

2 John Carmac FTW – BlueRaja - Danny Pflughoeft Jan 28 '10 at 19:57

Huffman trees - used for compression.

answered Feb 22 '09 at 4:47

community wiki
Lurker Indeed

14 This qualifies as lesser known nowadays? Really? – dietr Mar 24 '11 at 13:07

Have a look at Finger Trees, especially if you're a fan of the previously mentioned purely functional data structures. They're a functional representation of persistent sequences supporting access to the ends in amortized constant time, and concatenation and splitting in time logarithmic in the size of the smaller piece.

As per the original article:

> Our functional 2-3 finger trees are an instance of a general design technique in- troduced by Okasaki (1998), called *implicit recursive slowdown*. We have already noted that these trees are an extension of his implicit deque structure, replacing pairs with 2-3 nodes to provide the flexibility required for efficient concatenation and splitting.

A Finger Tree can be parameterized with a monoid, and using different monoids will result in different behaviors for the tree. This lets Finger Trees simulate other data structures.

edited Jan 15 '11 at 11:24 community wiki
4 revs, 2 users 80%
huitseeker

Circular or ring buffer - used for streaming, among other things.

answered Mar 17 '09 at 18:30 community wiki
cdonner

6 This is most common data structure for buffering data. I think that this one is first one introduced in our class. – Luka Rahne Sep 18 '10 at 15:02

4 Also, disgustingly, somehow managed to be patented (at least when used for video). ip.com/patent/USRE36801 – David Eison Mar 24 '11 at 18:32

I'm surprised no one has mentioned Merkle trees (ie. Hash Trees).

Used in many cases (P2P programs, digital signatures) where you want to verify the hash of a whole file when you only have part of the file available to you.

answered Jan 28 '10 at 20:03 community wiki
BlueRaja - Danny Pflughoeft

> <zvrba> Van Emde-Boas trees

I think it'd be useful to know *why* they're cool. In general, the question "why" is the most important to ask ;)

My answer is that they give you O(log log n) dictionaries with {1..n} keys, independent of how many of the keys are in use. Just like repeated halving gives you O(log n), repeated sqrting gives you O(log log n), which is what happens in the vEB tree.

answered Feb 1 '09 at 13:27 community wiki
Jonas Kölker

3 I fully agree that "why" is important, but that was not included in the question ;) – zvrba Feb 2 '09 at 15:30

How about splay trees?

Also, Chris Okasaki's purely functional data structures come to mind.

edited Jan 17 '10 at 10:24 community wiki
2 revs
starblue

An interesting variant of the hash table is called Cuckoo Hashing. It uses multiple hash functions instead of just 1 in order to deal with hash collisions. Collisions are resolved by removing the old object from the location specified by the primary hash, and moving it to a location specified by an alternate hash function. Cuckoo Hashing allows for more efficient use of memory space because you can increase your load factor up to 91% with only 3 hash functions and still have good access time.

answered May 10 '09 at 21:56 community wiki A. Levy

5 Check hopscotch hashing claimed to be faster. – chmike May 23 '10 at 6:43

A min-max heap is a variation of a heap that implements a double-ended priority queue. It achieves this by by a simple change to the heap property: A tree is said to be min-max ordered if every element on even (odd) levels are less (greater) than all childrens and grand children. The levels are numbered starting from 1.

answered Jan 15 '11 at 12:18 community wiki marcog

2 min-max heap is beautiful! Used in AI as well :) – Ricko M May 12 '11 at 13:59

I like Cache Oblivious datastructures. The basic idea is to lay out a tree in recursively smaller blocks so that caches of many different sizes will take advantage of blocks that convenient fit in them. This leads to efficient use of caching at everything from L1 cache in RAM to big chunks of data read off of the disk without needing to know the specifics of the sizes of any of those caching layers.

answered Mar 24 '11 at 22:20 community wiki btilly

1 +1 Of course!!! – Jon Harrop Mar 24 '11 at 22:55

Left Leaning Red-Black Trees. A significantly simplified implementation of red-black trees by Robert Sedgewick published in 2008 (~half the lines of code to implement). If you've ever had trouble wrapping your head around the implementation of a Red-Black tree, read about this variant.

Very similar (if not identical) to Andersson Trees.

answered May 23 '10 at 17:21 community wiki Lucas

Work Stealing Queue

Lock-free data structure for dividing the work equaly among multiple threads
http://stackoverflow.com/questions/2101789/implementation-of-a-work-stealing-queue-in-c-c

answered Sep 19 '10 at 17:54 community wiki Marko Tintor

Bootstrapped skew-binomial heaps by Gerth Stølting Brodal and Chris Okasaki:

Despite their long name, they provide asymptotically optimal heap operations, even in a function setting.

- `O(1)` size, **union**, insert, minimum
- `O(log n)` deleteMin

Note that union takes `O(1)` rather than `O(log n)` time unlike the more well-known heaps that are commonly covered in data structure textbooks, such as leftist heaps. And unlike Fibonacci heaps, those asymptotics are worst-case, rather than amortized, even if used persistently!

There are multiple implementations in Haskell.

They were jointly derived by Brodal and Okasaki, after Brodal came up with an imperative heap with the same asymptotics.

heap with the same asymptotics.

answered Jul 23 '10 at 6:15

community wiki
Edward KMETT

- Kd-Trees, spatial data structure used (amongst others) in Real-Time Raytracing, has the downside that triangles that cross intersect the different spaces need to be clipped. Generally BVH's are faster because they are more lightweight.
- MX-CIF Quadtrees, store bounding boxes instead of arbitrary point sets by combining a regular quadtree with a binary tree on the edges of the quads.
- HAMT, hierarchical hash map with access times that generally exceed O(1) hash-maps due to the constants involved.
- Inverted Index, quite well known in the search-engine circles, because it's used for fast retrieval of documents associated with different search-terms.

Most, if not all, of these are documented on the NIST Dictionary of Algorithms and Data Structures

edited Feb 18 '09 at 20:16

community wiki
3 revs, 2 users 55%
Jasper Bekkers

Ball Trees. Just because they make people giggle.

A ball tree is a data structure that indexes points in a metric space. Here's an article on building them. They are often used for finding nearest neighbors to a point or accelerating k-means.

edited Jul 23 '10 at 2:28

community wiki
2 revs
anon

1 There you go. They are a little obscure, I grant you. – Anonymous Jul 23 '10 at 2:29

Not really a data structure; more of a way to optimize dynamically allocated arrays, but the gap buffers used in Emacs are kind of cool.

answered May 23 '10 at 21:52

community wiki
kerkeslager

1 I would definitely consider that to be a data structure. – Christopher Barber Mar 28 '11 at 18:53

Fenwick Tree. It's a data structure to keep count of the sum of all elements in a vector, between two given subindexes i and j. The trivial solution, precalculating the sum since the begining doesn't allow to update a item (you have to do O(n) work to keep up).

Fenwick Trees allow you to update and query in O(log n), and how it works is really cool and simple. It's really well explained in Fenwick's original paper, freely available here:

http://www.cs.ubc.ca/local/reading/proceedings/spe91-95/spe/vol24/issue3/spe884.pdf

Its father, the RQM tree is also very cool: It allows you to keep info about the minimum element between two indexes of the vector, and it also works in O(log n) update and query. I like to teach first the RQM and then the Fenwick Tree.

answered Jul 23 '10 at 3:45

community wiki
eordano

Van Emde-Boas trees. I have even a C++ implementation of it, for up to 2^20 integers.

answered Feb 1 '09 at 13:06

community wiki
zvrba

1 duplicate answer – huitseeker Jan 14 '11 at 1:39

Nested sets are nice for representing trees in the relational databases and running queries on

Nested sets are nice for representing trees in the relational databases and running queries on them. For instance, ActiveRecord (Ruby on Rails' default ORM) comes with a very simple nested set plugin, which makes working with trees trivial.

edited May 23 '10 at 0:02

community wiki
2 revs
esad

It's pretty domain-specific, but half-edge data structure is pretty neat. It provides a way to iterate over polygon meshes (faces *and* edges) which is very useful in computer graphics and computational geometry.

answered May 23 '10 at 7:31

community wiki
mpen

1 2 3 next