




WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages 
فارسی
Српски / srpski
ไทย

 Edit links

Article **Talk**

Read **Edit** View history

Search 

Fibonacci search technique

From Wikipedia, the free encyclopedia



This article **may be too technical for most readers to understand**. Please help [improve](#) this article to [make it understandable to non-experts](#), without removing the technical details. The [talk page](#) may contain suggestions. *(July 2013)*

This article is about the programming algorithm. For the technique for finding extremum of a mathematical function, see [Golden section search](#).

In [computer science](#), the **Fibonacci search technique** is a method of searching a [sorted array](#) using a [divide and conquer algorithm](#) that narrows down possible locations with the aid of [Fibonacci numbers](#). Compared to [binary search](#), Fibonacci search examines locations whose addresses have lower dispersion. Therefore, when the elements being searched have non-uniform access memory storage (i.e., the time needed to access a storage location varies depending on the location previously accessed), the Fibonacci search has an advantage over binary search in slightly reducing the average time needed to access a storage location. The typical example of non-uniform access storage is that of a [magnetic tape](#), where the time to access a particular element is proportional to its distance from the element currently under the tape's head. Note, however, that large arrays not fitting in [CPU cache](#) or even in [RAM](#) can also be considered as non-uniform access examples. Fibonacci search has a complexity of $O(\log(n))$ (see [Big O notation](#)).

Fibonacci search was first devised by [Jack Kiefer](#) (1953) as a [minimax](#) search for the maximum (minimum) of a [unimodal function](#) in an interval.

Algorithm [\[edit\]](#)

Let k be defined as an element in F , the array of Fibonacci numbers. $n = F_m$ is the array size. If the array size is not a Fibonacci number, let F_m be the smallest number in F that is greater than n .

The array of Fibonacci numbers is defined where $F_{k+2} = F_{k+1} + F_k$, when $k \geq 0$, $F_1 = 1$, and $F_0 = 0$.

To test whether an item is in the list of ordered numbers, follow these steps:

- Set $k = m$.
- If $k = 0$, stop. There is no match; the item is not in the array.
- Compare the item against element in F_{k-1} .
- If the item matches, stop.
- If the item is less than entry F_{k-1} , discard the elements from positions $F_{k-1} + 1$ to n . Set $k = k - 1$ and return to step 2.
- If the item is greater than entry F_{k-1} , discard the elements from positions 1 to F_{k-1} . Renumber the remaining elements from 1 to F_{k-2} , set $k = k - 2$, and return to step 2.

Alternative implementation (from "Sorting and Searching" by Knuth):

Given a table of records R_1, R_2, \dots, R_N whose keys are in increasing order $K_1 < K_2 < \dots < K_N$, the algorithm searches for a given argument K . Assume $N+1 = F_{k+1}$

Step 1. [Initialize] $i \leftarrow F_k$, $p \leftarrow F_{k-1}$, $q \leftarrow F_{k-2}$ (throughout the algorithm, p and q will be consecutive Fibonacci numbers)

Step 2. [Compare] If $K < K_i$, go to *Step 3*; if $K > K_i$ go to *Step 4*; and if $K = K_i$, the algorithm terminates successfully.

Step 3. [Decrease i] If $q=0$, the algorithm terminates unsuccessfully. Otherwise set $(i, p, q) \leftarrow (i - q, q, p - q)$ (which moves p and q one position back in the Fibonacci sequence); then return to *Step 2*

Step 4. [Increase i] If $p=1$, the algorithm terminates unsuccessfully. Otherwise set $(i, p, q) \leftarrow (i + q, p - q, 2q - p)$ (which moves p and q two positions back in the Fibonacci sequence); and return to *Step 2*

See also [\[edit\]](#)

- [Golden section search](#)

- [Search algorithms](#)

References [[edit](#)]

- J. Kiefer, *Sequential minimax search for a maximum*, *Proc. American Mathematical Society* **4** (1953), 502–506. [↗](#)
- David E. Ferguson, "Fibonacci searching", *Communications of the ACM*, vol. 3, is. 12, p. 648, Dec. 1960.
- Manolis Lourakis, "Fibonacci search in C". [\[1\]](#) [↗](#). Retrieved January 18, 2007. Implements Ferguson's algorithm.
- Donald E. Knuth, "[The Art of Computer Programming](#) (second edition)", vol. 3, p. 418, Nov. 2003.

Categories: [Search algorithms](#)

This page was last modified on 13 August 2015, at 14:38.

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.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

