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
العربية
فارسی
Français
한국어
Русский
Српски / srpski
Edit links

Longest increasing subsequence

From Wikipedia, the free encyclopedia
(Redirected from Longest increasing subsequence problem)

In computer science, the **longest increasing subsequence** problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. This subsequence is not necessarily contiguous, or unique. Longest increasing subsequences are studied in the context of various disciplines related to mathematics, including algorithmics, random matrix theory, representation theory, and physics.[1] The longest increasing subsequence problem is solvable in time $O(n \log n)$, where n denotes the length of the input sequence.[2]

Contents [hide]

Example [edit]

In the first 16 terms of the binary Van der Corput sequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

a longest increasing subsequence is

0, 2, 6, 9, 11, 15.

This subsequence has length six; the input sequence has no seven-member increasing subsequences. The longest increasing subsequence in this example is not unique: for instance,

0, 4, 6, 9, 11, 15 or 0, 4, 6, 9, 13, 15

are other increasing subsequences of equal length in the same input sequence.

Relations to other algorithmic problems [edit]

The longest increasing subsequence problem is closely related to the longest common subsequence problem, which has a quadratic time dynamic programming solution: the longest increasing subsequence of a sequence S is the longest common subsequence of S and T, where T is the result of sorting S. However, for the special case in which the input is a permutation of the integers 1, 2, ..., n, this approach can be made much more efficient, leading to time bounds of the form $O(n \log \log n)$.[3]

The largest clique in a permutation graph is defined by the longest decreasing subsequence of the permutation that defines the graph; the longest decreasing subsequence is equivalent in computational complexity, by negation of all numbers, to the longest increasing subsequence. Therefore, longest increasing subsequence algorithms can be used to solve the clique problem efficiently in permutation graphs.[4]

In the Robinson–Schensted correspondence between permutations and Young tableaux, the length of the first row of the tableau corresponding to a permutation equals the length of the longest increasing subsequence of the permutation, and the length of the first column equals the length of the longest decreasing subsequence.[2]

Efficient algorithms [edit]

The algorithm outlined below solves the longest increasing subsequence problem efficiently with arrays and binary searching. It processes the sequence elements in order, maintaining the longest increasing subsequence found so far. Denote the sequence values as X[0], X[1], etc. Then, after processing X[i], the algorithm will have stored values in two arrays:

M[*j*] — stores the index *k* of the smallest value X[*k*] such that there is an increasing subsequence of length *j* ending at X[*k*] on the range $k \le i$. Note that $j \le k \le i$, because *j* represents the length of the increasing subsequence, and *k* represents the index of its termination. There can never be an increasing subsequence of length 13 ending at index 11. $k \le i$ by definition.

P[*k*] — stores the index of the predecessor of X[*k*] in the longest increasing subsequence ending at X[*k*].

In addition the algorithm stores a variable L representing the length of the longest increasing subsequence found so far. Because the algorithm below uses zero-based numbering, for clarity M is padded with M[0], which goes unused so that M[*j*] corresponds to a subsequence of length *j*. A real implementation can skip M[0] and adjust the indices accordingly.

Note that, at any point in the algorithm, the sequence

X[M[1]], X[M[2]], ..., X[M[L]]

is nondecreasing. For, if there is an increasing subsequence of length *i* ending at X[M[*i*]], then there is also a subsequence of length *i*-1 ending at a smaller value: namely the one ending at X[P[M[*i*]]]. Thus, we may do binary searches in this sequence in logarithmic time.

The algorithm, then, proceeds as follows:

```
P = array of length N
M = array of length N + 1

L = 0
for i in range 0 to N-1:
  // Binary search for the largest positive j ≤ L
  // such that X[M[j]] < X[i]
  lo = 1
  hi = L
  while lo ≤ hi:
    mid = ceil((lo+hi)/2)
    if X[M[mid]] < X[i]:
      lo = mid+1
    else:
      hi = mid-1

  // After searching, lo is 1 greater than the
  // length of the longest prefix of X[i]
  newL = lo

  // The predecessor of X[i] is the last index of
  // the subsequence of length newL-1
  P[i] = M[newL-1]
  M[newL] = i

  if newL > L:
    // If we found a subsequence longer than any we've
    // found yet, update L
    L = newL

// Reconstruct the longest increasing subsequence
S = array of length L
k = M[L]
for i in range L-1 to 0:
  S[i] = X[k]
  k = P[k]

return S
```

Because the algorithm performs a single binary search per sequence element, its total time can be expressed using Big O notation as $O(n \log n)$. Fredman (1975) discusses a variant of this algorithm, which he credits to Donald Knuth; in the variant that he studies, the algorithm tests whether each value X[*i*] can be used to extend the current longest increasing sequence, in constant time, prior to doing the binary search. With this modification, the algorithm uses at most $n \log_2 n - n \log_2 \log_2 n + O(n)$ comparisons in the worst case, which is optimal for a comparison-based algorithm up to the constant factor in the $O(n)$ term.[5]

Length bounds [edit]

According to the Erdős–Szekeres theorem, any sequence of n^2+1 distinct integers has an increasing or a decreasing subsequence of length $n + 1$.[6][7] For inputs in which each permutation of the input is equally likely, the expected length of the longest increasing subsequence is approximately $2\sqrt{n}$. [8] In the limit as n approaches infinity, the length of the longest increasing subsequence of a randomly permuted sequence of n items has a distribution approaching the Tracy–Widom distribution, the distribution of the largest eigenvalue of a random matrix in the Gaussian unitary ensemble.[9]

Online algorithms [edit]

The longest increasing subsequence has also been studied in the setting of online algorithms, in which the elements of a sequence of independent random variables with continuous distribution F – or alternatively the elements of a random permutation – are presented one at the time to an algorithm that must decide whether to include or exclude each element, without knowledge of the later elements. In this variant of the problem, which allows for interesting applications in several contexts, it is possible to devise a optimal selection procedure that, given a random sample of size n as input, will generate an increasing sequence with maximal expected length of size approximately $\sqrt{2n}$. [10] The length of the increasing subsequence selected by this optimal procedure has variance approximately equal to $\sqrt{2n/3}$, and its limiting distribution is asymptotically normal after the usual centering and scaling.[11] The same asymptotic results hold with more precise bounds for the corresponding problem in the setting of a Poisson arrival process.[12] A further refinement in the Poisson process setting is given through the proof of a central limit theorem for the optimal selection process which holds, with a suitable normalization, in a more complete sense than one would expect. The proof yields not only the "correct" functional limit theorem but also the (singular) covariance matrix of the three-dimensional process summarizing all interacting processes. [13]

See also [edit]

- Patience sorting, an efficient technique for finding the length of the longest increasing subsequence
- Plactic monoid, an algebraic system defined by transformations that preserve the length of the longest increasing subsequence
- Anatoly Vershik, a Russian mathematician who studied applications of group theory to longest increasing subsequences
- Longest common subsequence
- Longest alternating subsequence

References [edit]

1. ^ Aldous, David; Diaconis, Persi (1999), "Longest increasing subsequences: from patience sorting to the Baik–Deift–Johansson theorem", *Bulletin of the American Mathematical Society* **36** (04): 413–432, doi:10.1090/S0273-0979-99-00796-X.
2. ^ [a] [b] Schensted, C. (1961), "Longest increasing and decreasing subsequences", *Canadian Journal of Mathematics* **13**: 179–191, doi:10.4153/CJM-1961-015-3, MR 0121305.
3. ^ Hunt, J.; Szymanski, T. (1977), "A fast algorithm for computing longest common subsequences", *Communications of the ACM* **20** (5): 350–353, doi:10.1145/359581.359603.
4. ^ Golumbic, M. C. (1980), *Algorithmic Graph Theory and Perfect Graphs*, Computer Science and Applied Mathematics, Academic Press, p. 159.
5. ^ Fredman, Michael L. (1975), "On computing the length of longest increasing subsequences", *Discrete Mathematics* **11** (1): 29–35, doi:10.1016/0012-365X(75)90103-X.
6. ^ Erdős, Paul; Szekeres, George (1935), "A combinatorial problem in geometry", *Compositio Mathematica* **2**: 463–470.
7. ^ Steele, J. Michael (1995), "Variations on the monotone subsequence theme of Erdős and Szekeres", in Aldous, David; Diaconis, Persi; Spencer, Joel et al., *Discrete Probability and Algorithms* (PDF), IMA Volumes in Mathematics and its Applications **72**, Springer-Verlag, pp. 111–131 .
8. ^ Vershik, A. M.; Kerov, C. V. (1977), "Asymptotics of the Plancheral measure of the symmetric group and a limiting form for Young tableaux", *Dokl. Akad. Nauk USSR* **233**: 1024–1027.
9. ^ Baik, Jinho; Deift, Percy; Johansson, Kurt (1999), "On the distribution of the length of the longest increasing subsequence of random permutations", *Journal of the American Mathematical Society* **12** (4): 1119–1178, arXiv:math/9810105, doi:10.1090/S0894-0347-99-00307-0.
10. ^ Samuels, Stephen. M.; Steele, J. Michael (1981), "Optimal Sequential Selection of a Monotone Sequence From a Random Sample", *Annals of Probability* **9** (6): 937–947, doi:10.1214/aop/1176994265
11. ^ Arlotto, Alessandro; Nguyen, Vinh V.; Steele, J. Michael (2015), "Optimal online selection of a monotone subsequence: a central limit theorem", *Stochastic Processes and their Applications*,

doi:10.1016/j.spa.2015.03.009

12. ^ Bruss, F. Thomas; Delbaen, Freddy (2001), "Optimal rules for the sequential selection of monotone subsequences of maximum expected length", *Stochastic Processes and their Applications* **96** (2): 313–342.
13. ^ Bruss, F. Thomas; Delbaen, Freddy (2004), "A central limit theorem for the optimal selection process for monotone subsequences of maximum expected length", *Stochastic Processes and their Applications* **114** (2): 287–311, doi:10.1016/j.spa.2004.09.002.

External links [edit]

- Algorithmist's Longest Increasing Subsequence
- Explanation on Geeksforgeeks
- Finding count of longest increased subsequences

Categories: Problems on strings | Combinatorics | Formal languages | Dynamic programming