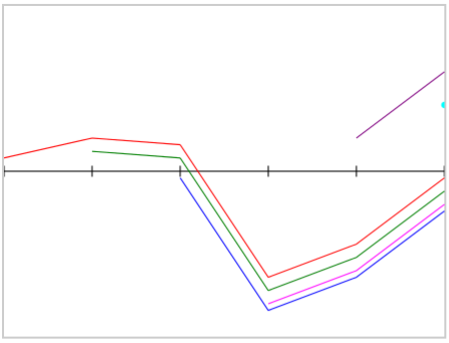
Maximum subarray problem

From Wikipedia, the free encyclopedia

In computer science, the **maximum subarray problem** is the task of finding the contiguous subarray within a one-dimensional array of numbers which has the largest sum. For example, for the sequence of values −2, 1, −3, 4, −1, 2, 1, −5, 4; the contiguous subarray with the largest sum is 4, −1, 2, 1, with sum 6.


Visualization of how sub-arrays change based on start and end positions of a sample. Each possible contiguous sub-array is represented by a point on a colored line. That point's y-coordinate represents the sum of the sample. Its x-coordinate represents the end of the sample, and the leftmost point on that colored line represents the start of the sample. In this case, the array from which samples are taken is [2, 3, -1, -20, 5, 10].

The problem was first posed by Ulf Grenander of Brown University in 1977, as a simplified model for maximum likelihood estimation of patterns in digitized images. A linear time algorithm was found soon afterwards by Jay Kadane of Carnegie Mellon University (Bentley 1984).

Contents

Kadane's algorithm

Kadane's algorithm begins with a simple inductive question: if we know the maximum subarray sum ending at position i, what is the maximum subarray sum ending at position $i+1$? The answer turns out to be relatively straightforward: either the maximum subarray sum ending at position $i+1$ includes the maximum subarray sum ending at position i as a prefix, or it doesn't. Thus, we can compute the maximum subarray sum ending at position i for all positions i by iterating once over the array. As we go, we simply keep track of the maximum sum we've ever seen. Thus, the problem can be solved with the following code, expressed here in Python:

```python
def max_subarray(A):
    max_ending_here = max_so_far = A[0]
    for x in A[1:]:
        max_ending_here = max(x, max_ending_here + x)
        max_so_far = max(max_so_far, max_ending_here)
    return max_so_far
```

The algorithm can also be easily modified to keep track of the starting and ending indices of the maximum subarray (when `max_so_far` changes) as well as the case where we want to allow zero-length subarrays (with implicit sum 0) if all elements are negative.

Because of the way this algorithm uses optimal substructures (the maximum subarray ending at each position is calculated in a simple way from a related but smaller and overlapping subproblem: the maximum subarray ending at the previous position) this algorithm can be viewed as a simple/trivial example of dynamic programming.

The runtime complexity of Kadane's algorithm is $O(n)$.

Generalizations

Similar problems may be posed for higher-dimensional arrays, but their solutions are more complicated; see, e.g., Takaoka (2002). Brodal & Jørgensen (2007) showed how to find the *k* largest subarray sums in a one-dimensional array, in the optimal time bound $O(n+k)$.

The Maximum sum *k*-disjoint subarrays can also be computed in the optimal time bound $O(n+k)$.[1]

See also

- Subset sum problem

References

1. Bengtsson, Fredrik; Chen, Jingsen (2007). "Computing maximum-scoring segments optimally" (https://pure.ltu.se/portal/files/1805041/Research_Report_2007_03). *Luleå University of Technology* (3).

- Bentley, Jon (1984), "Programming pearls: algorithm design techniques", *Communications of the ACM*, **27** (9): 865 – 873, doi:10.1145/358234.381162 (https://doi.org/10.1145%2F358234.381162).
- Brodal, Gerth Stølting; Jørgensen, Allan Grønlund (2007), "A linear time algorithm for the *k* maximal sums problem", *Mathematical Foundations of Computer Science 2007*, Lecture Notes in Computer Science, **4708**, Springer-Verlag, pp. 442 – 453, doi:10.1007/978-3-540-74456-6_40 (https://doi.org/10.1007%2F978-3-540-74456-6_40).
- Takaoka, T. (2002), "Efficient algorithms for the maximum subarray problem by distance matrix multiplication" (http://www.cosc.canterbury.ac.nz/tad.takaoka/cats02.pdf) (PDF), *Electronic Notes in Theoretical Computer Science*, **61**.

External links

- www.algorithmist.com (http://www.algorithmist.com/index.php/Kadane's_Algorithm)
- alexeigor.wikidot.com (http://alexeigor.wikidot.com/kadane)
- greatest subsequential sum problem on Rosetta Code (http://rosettacode.org/wiki/Greatest_subsequential_sum)

Categories: Optimization algorithms and methods | Dynamic programming