The Price Isn't Always Right

2016-10-20 Thu

Aconcagua.com sells storage on both an auction and fixed-price basis. They want to use historical auction data to investigate their fixed price choices.

For n seconds, they have the price point reached in each second in their auctions. They want to find the largest price-over-time stretch in their data. That is, given an array A of n price points, they want to find the largest possible value of $f(i, d) = d * \min(A[i], A[i+1], \ldots, A[i+d-1], A[i+d])$ where i is the index of the left end of a stretch of seconds, d is the duration in seconds of that stretch, and the function f computes the duration times the minimum price over that period. (Prices are positive, $d \geq 0$, and for all values of i, $f(i, 0) = 0$ and $f(i, 1) = A[i]$.)

For example, the best stretch is underlined in the following price array: $[8, 2, \underline{9, 5, 6, 5}, 3, 1]$. Using 1-based indexing, the value for this optimal stretch starting at index 3 and running for 4 seconds is $f(3, 4) = 4 * \min(9, 5, 6, 5) = 4 * 5 = 20$.

1. Give a **brute force** algorithm to solve this problem. Your algorithm must run in polynomial time.

2. Give and briefly justify a good asymptotic bound on the runtime of your algorithm.

1 Divide-and-Conqaguar

In this part, you will give a divide-and-conquer approach that is more efficient than the brute force approach.

1. Consider again the example array from the previous part: $[8, 2, 9, 5, 6, 5, 3, 1]$. Imagine that you are told that the solution **must** use the elements at indexes 4 and 5 (i.e., elements 5 and 6). You want to expand the stretch either to the left or right one step at a time while maintaining the invariant that the stretch you have chosen is the best that includes indexes 4 and 5 until the stretch includes the entire array.

   Finish the following table describing this expansion, thinking carefully about why you make the choice you do at each point:

i	d	minimum	$f(i,d)$
4	2	5	10
3	3	5	15
3	4	5	20
...	...	...	...
1	8	1	8

2. Give an efficient algorithm for finding the optimal price stretch.

3. Give and briefly justify a good asymptotic bound on the runtime of your algorithm.