The set of values that we need to search is the set of sums of weights of vertices from i to j, for all pairs $i \leq j$. We succinctly maintain this set of values, plus others, in a data structure, the *succinct description.* For $i = 0, 1, \cdots, n$, let A_i be the sum of the weights of vertices 1 to i. Note that for any pair i, j with $i \leq j$, the sum of the weights from vertex i to j is $A_j - A_{i-1}$. Let X_1 be the sequence of sums $A_1, A_2, \cdots, A_n$, and let X_2 be the sequence of sums $A_0, A_1, \cdots, A_{n-1}$. Then sorted matrix $M(P)$ is the $n \times n$ Cartesian matrix $X_1 - X_2$, where the ij-th entry is $A_j - A_{i-1}$. In determining the above, we can use proper subtraction, in which, $a - b$ gives $\max\{a - b, 0\}$. Clearly, the values in any row of $M(P)$ are in nondecreasing order, and the values in any column of $M(P)$ are in nonincreasing order. Representing $M(P)$ explicitly requires $\Theta(n^2)$ time and space. Thus, the algorithm succinctly represents $M(P)$ (and hence, P) in $O(n)$ space by the two vectors X_1 and X_2.

The algorithm may also need to inspect specific subpaths of P. However, repeatedly copying subvectors of X_1 and X_2 can take more than linear time in total. On the other hand, for a subpath Q of P, the corresponding matrix $M(Q)$ is a submatrix of $M(P)$, which can be recovered from the succinct representation of P. Thus, the algorithm succinctly represents $M(Q)$ by the start and end indices of $M(Q)$ within $M(P)$. In this way, the values of $M(Q)$ can be generated from the vectors X_1 and X_2 of $M(P)$, as well as the location of the submatrix as given by $X(Q)$. Therefore, the algorithm avoids needlessly recopying vectors and instead succinctly represents all subpaths in $O(n)$ total time.