Optimal Binary Search Tree in $O(n^2)$ time

The following is taken from the book *Algorithms Illuminated* by Tim Roughgarden.

> First, compute in advance all sums of the form $\sum_{k=i}^{j} p_k$; this can be done in O(n2) time (do you see how?). Then, along with each subproblem solution $c[i][j]$, store the choice of the root $r(i,j)$ that minimizes $c[i][r-1] + c[r+1][j]$ or, equivalently, the root of an optimal search tree for the subproblem. (If there are multiple such roots, use the smallest one.)
>
> The key lemma is an easy-to-believe (but tricky-to-prove) monotonicity property: Adding a new maximum (respectively, minimum) element to a subproblem can only make the root of an optimal search tree larger (respectively, smaller). Intuitively, any change in the root should be in service of rebalancing the total frequency of keys between its left and right subtrees.
>
> Assuming this lemma, for every subproblem with $i < j$, the optimal root $r(i,j)$ is at least $r(i,j-1)$ and at most $r(i+1,j)$. (If $i = j$,then $r(i,j)$ must be i.) Thus, there's no point in exhaustively searching all the roots between $i$and j – the roots between $r(i,j-1)$ and $r(i+1,j)$ suffice. In the worst case, there could be as many as n such roots. In aggregate over all $\Theta(n^2)$ subproblems, however, the number of roots examined is $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (r(i+1,j) - r(i,j-1) + 1)$, which after cancellations is only $O(n^2)$ (as you should check). For further details, see the paper "Optimum Binary Search Trees," by Donald E. Knuth (Acta Informatica, 1971).