The Combinatorics of Nearest and Furthest Smaller Values

Trevor Clokie, Jeffrey Shallit, Lily Wang

School of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

trevor.clokie@uwaterloo.ca
shallit@uwaterloo.ca
x654wang@uwaterloo.ca

Nearest Smaller Value

- Given A, an array of integers, we consider two corresponding arrays:
- C, the **nearest smaller value** array, where $C[i]$ is the position of the nearest smaller (or equal) element of $A[i]$ on its left, or 0 if all elements to its left are larger. Formally, we set $A[0] = -\infty$, and

$$C[i] := \max \{0 \leq j < i \ : \ A[j] \leq A[i]\},$$

- E.g. if $A = [3, 4, 1, 5, 2]$, then $C = [0, 1, 0, 3, 3]$.

Furthest Smaller Value

- B, the **furthest smaller value** array, where $B[i]$ is the position of the furthest smaller (or equal) element of $A[i]$ on its left, or i if all elements to its left are larger. Formally,

$$B[i] = \min \{1 \leq j \leq i : A[j] \leq A[i]\}$$

- e.g., if $A = [3, 4, 1, 5, 2]$, then $B = [1, 1, 3, 1, 3]$.
- Let S_n denote the set of all permutations of $\{1, 2, \ldots, n\}$. For each A in S_n, we consider generating the corresponding arrays B and C.

Distinct Nearest Smaller Value Arrays

Result of generating C for all A in S_3:

$$\begin{aligned} [1,2,3] &\to [0,1,2] & [1,3,2] &\to [0,1,1] \\ [2,1,3] &\to [0,0,2] & [2,3,1] &\to [0,1,0] \\ [3,1,2] &\to [0,0,2] & [3,2,1] &\to [0,0,0]. \end{aligned}$$

How many distinct arrays can we generate over S_n?

Let us denote the set of C over all A in a set S by $nsv(S)$. If we compute the values of $|nsv(S_n)|$ for small n, we get 1, 2, 5, 14, 42, ... the first few terms of the **Catalan numbers**.

$C_{n+1} = \sum_{0 \leq i \leq n} C_i C_{n-i}$

- We'll prove that $|nsv(S_n)| = C_n$ using the recurrence relation.
- Consider the possible positions of 1 in any $A \in S_n$. If 1 is at position i in A, then $C[1 \ldots i-1]$ corresponds to the nearest smaller values of $A[1 \ldots i-1]$ and $C[i+1 \ldots n]$ corresponds to the nearest smaller values of $A[i+1 \ldots n]$ increased by i.
- Conversely, given any C, we can construct an A generating it by placing 1 at the index of the rightmost 0, and recursively constructing the two subarrays.
- This gives us a bijection from $nsv(S_{n+1})$ to $\bigcup_{i=0}^{n} nsv(S_i) \times nsv(S_{n-i})$

Sum of all Values

- If we sum over all values of all arrays generated by each array from S_n, we get
- **Theorem**: Let T_n denote the sum of all elements of all entries corresponding to all arrays generated by $A \in (S_n)$. Then $T_n = (n+1)!(n+2-2H_{n+1})/2$, where $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$, the n'th Harmonic Number
- For any value we fix at the last index of A, the first $n-1$ values can be arranged to be any ordering of S_n; the sum of the first $n-1$ values of C is nT_{n-1}.
- The sum of the last values of C over S_n is $n!(n-H_n)$.
- We can verify $(n+1)!(n+2-2H_{n+1})/2 = n!(n-H_n) + n!n(n+1-2H_n)/2$.

Standard Algorithm for Nearest Smaller Value

The stack consists of (value, index) pairs.

---

1. C := array[1..n] of integer;
2. Initialize stack S with pair $(-\infty, 0)$.
3. For $i := 1$ to n do
4.   While $(\text{top}(S))[1] \geq A[i]$ do pop(S);
5. $C[i] := (\text{top}(S))[2]$;
6.   push$(S, (A[i], i))$;
7. Return(C)

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = []$
- Stack $= (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0]$
- Stack $= (4, 1), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0]$
- Stack $= (4, 1), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0]$
- Stack $= (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0]$
- Stack $= (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0]$
- Stack $= (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2]$
- Stack $= (3, 3), (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2]$
- Stack $= (3, 3), (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2]$
- Stack $= (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2, 2]$
- Stack $= (2, 4), (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2, 2]$
- Stack $= (2, 4), (1, 2), (-\infty, 0)$

Example of the Algorithm

- $A = [4, 1, 3, 2, 5]$
- $C = [0, 0, 2, 2, 4]$
- Stack $= (5, 5), (2, 4), (1, 2), (-\infty, 0)$

The Stack at the End of the Algorithm

- Define $i_0 = 0$ and iteratively define i_{j+1} to be the position of the smallest element in $A[i_j + 1..n]$, to get an increasing sequence $i_0 = 0, i_1, i_2, \ldots, i_t = n$.
- **Lemma:** On input $A = A[1..n]$ the stack contents at the end of the algorithm is

$$(A[i_t], i_t), (A[i_{t-1}], i_{t-1}), \ldots, (A[i_1], i_1), (-\infty, 0)$$

Cycles from the Stack

- **Theorem:** The number of permutations for which the algorithm has stack height k at the end of the computation is $\left[{n \atop k-1}\right]$ for $2 \leq k \leq n+1$, a stirling number of the first kind.
- A value is permanently added to the stack from A when it is less than all values to its right; this motivates a natural bijection for A into cycles, where we end each cycle with the least remaining value in the list.
- e.g. $[41325] \to (41)(32)(5)$.
- **Corollary:** The number of size-n permutations for which the algorithm performs k stack pops is $\left[{n \atop n-k}\right]$, $0 \leq k < n$.

Expected Height

- What is the expected height of the stack at the end of the algorithm?

$$\frac{1}{n!}\sum_{2\leq k\leq n+1} k \begin{bmatrix} n \\ k-1 \end{bmatrix} = \frac{1}{n!}\sum_{1\leq j\leq n} (j+1) \begin{bmatrix} n \\ j \end{bmatrix}$$

$$= \left(\frac{1}{n!}\sum_{1\leq j\leq n} j \begin{bmatrix} n \\ j \end{bmatrix} \right) + 1$$

$$= H_n + 1$$

$$= \Theta(log(n))$$

Limited Space

- If we limit the maximum height of the stack during the entire computation, how many permutations achieve the limit?
- let $M(n,i)$ denote the number of permutations in S_n which have max stack height at most i during the algorithm.

$n \backslash i$	1	2	3	4	5	6	7	8	9
1	0	1	1	1	1	1	1	1	1
2	0	1	2	2	2	2	2	2	2
3	0	1	5	6	6	6	6	6	6
4	0	1	15	23	24	24	24	24	24
5	0	1	52	106	119	120	120	120	120
6	0	1	203	568	700	719	720	720	720
7	0	1	877	3459	4748	5013	5039	5040	5040
8	0	1	4140	23544	36403	39812	40285	40319	40320

Furthest Smaller Value

- We'll now look at B, the array of furthest smaller values generated from A.
- How many distinct arrays B can we generate from A over S_n?
- Let $fsv(S_n)$ denote the set of furthest smaller value arrays generated from S_n.

Furthest Smaller Value

- Successive minima of A are a set of values where each is least among all values to its left. E.g. in [4, 3, 1, 5, 2], the successive minima are [4, 3, 1].
- We'll define $T(n, k)$ to be the number of distinct furthest smaller arrays over $A \in S_n$ where A has k successive minima.
- **Lemma:** $T(n+1, k) = kT(n, k) + T(n, k-1)$ for all n and $2 \leq k \leq n$.

$T(n+1,k) = kT(n,k) + T(n,k-1)$

- Let A be any permutation of S_{n+1} with k successive minima, which generates B as its furthest smaller value array.
- If $B[n+1] = n+1$, then the last value of A must be 1 and it is a successive minimum. If we subtract 1 from all other values of A and remove the last value, we get a permutation of S_n with $k-1$ successive minima which generates $B[1..n]$. This gives us a bijection to count $T(n,k-1)$ values of B.
- Otherwise, then the last value of A is not 1 and not a successive minimum. If we subtract 1 from values of A greater than $A[n+1]$ and remove the last value, we preserve the relative orders and generate the same $B[1..n]$ except at the last value. Since each furthest smallest value is also a successive minimum, there are k possible $B[n+1]$ and we get $kT(n,k)$ values of B.

Counting Distinct Arrays

- The result of $T(n+1,k) = kT(n,k) + T(n,k-1)$ resembles the recurrence relations for the **Stirling number of the second kind**, which counts the number of ways to parition n values into k sets:

$$\left\{ {n+1 \atop k} \right\} = k \left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

- We can verify that $T(1,1) = \left\{ {1 \atop 1} \right\}$ and use induction to show that $T(n,k) = \left\{ {n \atop k} \right\}$.
- Then $|fsv(S_n)| = \sum_{k=0}^{n} T(n,k) = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} = B_n$, the **Bell numbers**, which also count the ways to partition a set.

Further Work - Asymptotic Estimate

- We end with an unsolved question: Returning to our stack height question from the nearest smaller value algorithm, what is our expected maximum stack height?
- **Theorem:** We have $M(1, i) = 1$ for $i \geq 2$; $M(n, 2) = 1$ for $n \geq 1$; and $M(n+1, i) = \sum_{0 \leq k \leq n} \binom{n}{k} M(k, i) M(n-k, i-1)$ for $n \geq 1$ and $i \geq 1$.

$$
\begin{aligned}
\mathbb{E}(\mathrm{StackHeight}(A_n)) &= \frac{1}{n!} \sum_{i=1}^{n+1} i(M(n, i) - M(n, i-1)) \\
&= \frac{1}{n!}[(n+1)n! - \sum_{i=1}^{n+1} M(n, i-1)] \\
&= n + 1 - \frac{1}{n!} \sum_{i=0}^{n} M(n, i)
\end{aligned}
$$