Homework 12 sample solution, part 2

Due 10/19/16

October 13, 2016

Stirling Numbers of the Second Kind, represented as $S(n, k)$, count (among other things) the number of ways to divide n objects into exactly k nonempty piles. For example, there are 7 ways to divide four objects into two piles:

$$\begin{array}{cc} \{1,2,3\} & \{4\} \\ \{1,2,4\} & \{3\} \\ \{1,3,4\} & \{2\} \\ \{2,3,4\} & \{1\} \\ \{1,2\} & \{3,4\} \\ \{1,3\} & \{2,4\} \\ \{1,4\} & \{2,3\} \end{array}$$

Stirling Numbers of the Second Kind can be calculated using the following recurrence relation:

$$S(n,k) = \begin{cases} 1, & \text{if } k = 1 \text{ or } k = n \\ kS(n-1,k) + S(n-1,k-1), & \text{otherwise} \end{cases}$$

In Homework 12, you developed a memoized dynamic programming algorithm to compute $S(n, k)$ efficiently. In this assignment, you will transform your memoized algorithm into an iterative algorithm.

1. Draw a representation of the data structure used to compute $S(6, 3)$. You do not need to fill in the values. Decorate this diagram as follows:

   (a) Mark the data structure element(s) representing base cases of the recurrence with a capital 'B'.

   (b) Mark the data structure element representing $S(5, 3)$ with a star ($\star$).

   (c) Mark the data structure element(s) representing the subproblems that $S(5, 3)$ depends on *directly* with a capital 'D'.

   (d) Mark all other data structure elements representing subproblems that $S(5, 3)$ depends on with a capital 'I'. Do not "overwrite" any previous marks.

B		
B	B	
B	I	B
B	D	D
B		$\star$
B		

or its transpose

2. Describe, either in English or using one or more loops, an order for evaluating the elements of this data structure so that

   (a) The first elements encountered are base cases ('B')

   (b) The dependent data elements ('D') always precede the element that depends on them ($\star$), for every element. Note that this implies that every indirect dependency ('I') will precede every direct dependency ('D').

   Multiple possible answers; for example, left-to-right, top-to-bottom.

   Or:

```
for i = 1 to n do
    for j = 1 to k do
        /* ...                 */
    end
end
```

   Another possible answer:

```
for i = 1 to k do
    for j = i to n do
        /* ...                 */
    end
end
```

3. Describe an iterative dynamic programming algorithm that computes $S(n, k)$. You do not need to reduce the space complexity of this algorithm relative to the memoized algorithm, though you may, if you want.

Input: n: number of objects
Input: k: number of piles
Output: $S(n, k)$: number of ways to divide n objects into k nonempty piles

Algorithm: IterStirling2

$S = \text{Array}(n, k)$
for $i = 1$ to n **do**
    for $j = 1$ to k **do**
        /* It's a good idea to do *something* about $i = 1$ (or $i < j$) so that the recursive case doesn't go out of bounds */
        if $j = 1$ or $j = i$ or $i = 1$ **then**
            $S[i, j] = 1$
        else
            $S[i, j] = k \cdot S[i - 1, j] + S[i - 1, j - 1]$
        end
    end
end
return $S[n, k]$

Alternate solution that uses $\Theta(n - k)$ space:

Input: n: number of objects
Input: k: number of piles
Output: $S(n, k)$: number of ways to divide n objects into k nonempty piles

Algorithm: BetterStirling2

$prev = \text{Array}(n - k)$
$curr = \text{Array}(n - k)$
Initialize $prev$ to 1
for $i = 2$ to k **do**
    $curr[1] = 1$
    for $j = 2$ to $n - k$ **do**
        $curr[j] = k \cdot curr[j - 1] + prev[j]$
    end
end
return $curr[n - k]$