codility

WE TEST CODERS

Chapter 3

Prefix sums

Prefix sums provide a simple yet powerful technique that allows for the fast calculation of sums of elements in given contiguous segments of arrays. Prefix sums are defined as the consecutive totals of the first $0, 1, 2, \ldots, n$ elements of an array.

	a_0	a_1	a_2	...	a_{n-1}
$p_0 = 0$	$p_1 = a_0$	$p_2 = a_0 + a_1$	$p_3 = a_0 + a_1 + a_2$	...	$p_n = a_0 + a_1 + \ldots + a_{n-1}$

We can easily calculate the prefix sums in $O(n)$ time complexity. Notice that the total p_k equals $p_{k-1} + a_{k-1}$, so each consecutive value can be calculated in a constant time.

3.1: Counting prefix sums.

```
1  def prefix_sums(A):
2      n = len(A)
3      p = [0] * (n + 1)
4      for k in xrange(1, n + 1):
5          p[k] = p[k - 1] + A[k - 1]
6      return p
```

Similarly, we can calculate suffix sums, which are the totals of the k last values. Using prefix (or suffix) sums allows us to calculate the total of any slice (contiguous segments) of the array very quickly. For example, you are asked about the totals of m slices (x, y) such that $0 \leqslant x \leqslant y < n$, where the total is the sum $a_x + a_{x+1} + \ldots + a_{y-1} + a_y$.

The simplest approach is to iterate through the whole array for each result separately; however, that requires of $O(n \cdot m)$ time. The better approach is to use prefix sums. If we calculate the prefix sums then we can answer each question directly in constant time. Let's subtract p_x from the value p_{y+1}.

p_{y+1}	a_0	a_1	...	a_{x-1}	a_x	a_{x+1}	...	a_{y-1}	a_y
p_x	a_0	a_1	...	a_{x-1}					
$p_{y+1} - p_x$					a_x	a_{x+1}	...	a_{y-1}	a_y

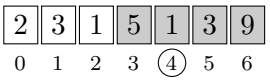
We have calculated the total of $a_x + a_{x+1} + \ldots + a_{y-1} + a_y$ in $O(1)$ time. Using this approach, the total time complexity is $O(n + m)$.

3.1. Exercises

Problem: You are given a non-empty, zero-indexed array A of n integers: $a_0, a_1, \ldots, a_{n-1}$ such that $(0 \leqslant a_i \leqslant 1\,000)$, and integers k and m such that $0 \leqslant k, m < n \leqslant 100\,000$.

A robot is at position k in array A and should perform m moves. One move moves a robot to an adjacent cell of the array. In every cell, the robot collects the value in the cell and replaces it with 0. The sum is the total of all collected values, and the goal is to calculate the maximum sum that the robot can collect in m moves.

For example, consider array A such that:



The robot starts at position $k = 4$ and should perform $m = 4$ moves. The robot might move to cells $(3, 4, 5, 6)$ and thereby collect the values $1 + 5 + 0 + 3 + 9 = 18$. This is the maximal sum that the robot can collect.

Solution: Note that the best option is to move in one direction optionally followed by some moves in the opposite direction. The robot should not change direction more than once. With this observation you can find the simplest solution. Make the first $p = 0, 1, 2, \ldots, m$ moves in one direction, then the next $m - p$ moves in the opposite direction. This is just a simple simulation of the moves of the robot, and requires $O(n \cdot m)$ time.

A better approach is to use prefix sums. If we make p moves in one direction, we can calculate the maximal opposite location of the robot. The robot collects all values between these extremes. We can calculate the total collected values in constant time by using prefix sums, and the total time complexity, at such a solution is $O(n + m)$.

Every lesson will provide you with programming tasks at `http://codility.com/train`.