Cycle and Component Detection in Signed Permutations

1. Problem

Let π be a signed permutation of length n. We are interested in analyzing the cycle and component structure of π using a cycle graph representation.

1. Show that the number of cycles in π can be found in $O(n)$ time.
2. Show that all components in π can be found in $O(n)$ time.

2. Solution

2.1. (a) Computing the Number of Cycles in $O(n)$ Time

To compute the number of cycles efficiently, we transform the signed permutation π into a cycle graph. This involves the following steps:

Step 1: Extend and Transform the Permutation

1. Extend π with sentinel elements:

$$\pi(0) = 0, \quad \pi(2n+1) = 2n+1$$

2. Convert each signed element π_i into an unsigned pair:
   (a) If $\pi_i > 0$: form the pair $(2\pi_i - 1, 2\pi_i)$.
   (b) If $\pi_i < 0$: form the pair $(2|\pi_i|, 2|\pi_i| - 1)$.
3. The transformation produces a permutation over $\{0, 1, 2, \ldots, 2n+1\}$, with $2n+2$ vertices.

Step 2: Construct the Cycle Graph

1. Add gray edges between $\pi(2i)$ and $\pi(2i+1)$ to represent the permutation.
2. Add black edges between $2i$ and $2i+1$ to represent the identity permutation.
3. Each cycle consists of alternating gray and black edges.
4. The resulting structure is a disjoint union of cycles.

Step 3: Count Cycles via Overlap Graph in $O(n)$ Time

1. Construct the overlap graph:
   (a) Each vertex corresponds to a pair (i.e., an interval) from the unsigned transformation.
   (b) An edge exists between two vertices if their intervals overlap.
2. Build a forest where each tree corresponds to a cycle (i.e., connected component).
3. Traverse each vertex and edge once to detect connected components.

Note

The number of cycles in π equals the number of connected components in the overlap graph. Since each element and edge is visited once, the total time complexity is $O(n)$.

2.2. (b) Computing All Components in $O(n)$ Time

We now describe how to find all connected components in linear time using a stack-based method.

Algorithm Description

1. For each i, let $C[i] = (A_i, B_i)$ be the interval from the unsigned pair representation.
2. Initialize an empty stack.
3. Iterate i from 0 to $2n + 1$:
   (a) If $i = C[i].A$, push $C[i]$ onto the stack.
   (b) Set extent $\leftarrow C[i]$.
   (c) While the top of the stack satisfies top.A $> C[i].A$:
      i. Update extent.A $\leftarrow$ $\min(\text{extent.A}, \text{top.A})$.
      ii. Update extent.B $\leftarrow$ $\min(\text{extent.B}, \text{top.B})$.
      iii. Pop the top of the stack and set parent[top.A] $\leftarrow C[i].A$.
   (d) Update the top of the stack with the new extent values.
   (e) If $i =$ top.B, pop the top.
4. After the loop, use the parent[] array to extract trees (i.e., connected components).

Note

Since each vertex and edge is processed only once, all components of π can be found in $O(n)$ time.