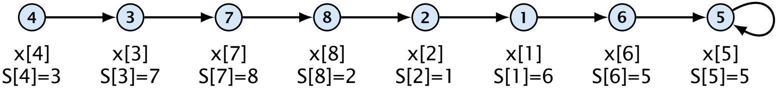
Parallel Prefix

Input: a linked list given by successor pointers; a value $x[i]$ for every list element; an operator $*$;

Output: for every list position ℓ the sum (w.r.t. $*$) of elements after ℓ in the list (including ℓ)



Parallel Prefix

Algorithm 7 ParallelPrefix

> **for** $1 \leq i \leq n$ **pardo**
> $P[i] \leftarrow S[i]$
> **while** $S[i] \neq S[S[i]]$ **do**
> $x[i] \leftarrow x[i] * x[S[i]]$
> $S[i] \leftarrow S[S[i]]$
> **if** $P[i] \neq i$ **then** $S[i] \leftarrow x[S(i)]$

The algorithm runs in time $\mathcal{O}(\log n)$.

It has work requirement $\mathcal{O}(n \log n)$. non-optimal

This technique is also known as pointer jumping

Parallel Prefix

Algorithm 7 ParallelPrefix

```
for 1 ≤ i ≤ n pardo
    P[i] ← S[i]
    while S[i] ≠ S[S[i]] do
        x[i] ← x[i] ∗ x[S[i]]
        S[i] ← S[S[i]]
    if P[i] ≠ i then S[i] ← x[S(i)]
```

The algorithm runs in time $\mathcal{O}(\log n)$.

It has work requirement $\mathcal{O}(n \log n)$. non-optimal

This technique is also known as pointer jumping

Parallel Prefix

Algorithm 7 ParallelPrefix

for $1 \leq i \leq n$ **pardo**
    $P[i] \leftarrow S[i]$
    while $S[i] \neq S[S[i]]$ **do**
        $x[i] \leftarrow x[i] * x[S[i]]$
        $S[i] \leftarrow S[S[i]]$
    if $P[i] \neq i$ **then** $S[i] \leftarrow x[S(i)]$

The algorithm runs in time $\mathcal{O}(\log n)$.

It has work requirement $\mathcal{O}(n \log n)$. non-optimal

This technique is also known as pointer jumping

Parallel Prefix

Algorithm 7 ParallelPrefix

for $1 \le i \le n$ **pardo**
  $P[i] \leftarrow S[i]$
  while $S[i] \neq S[S[i]]$ **do**
    $x[i] \leftarrow x[i] * x[S[i]]$
    $S[i] \leftarrow S[S[i]]$
  if $P[i] \neq i$ **then** $S[i] \leftarrow x[S(i)]$

The algorithm runs in time $\mathcal{O}(\log n)$.

It has work requirement $\mathcal{O}(n \log n)$. non-optimal

This technique is also known as pointer jumping