Chapter 10
Parallel Algorithms

Algorithm Theory
WS 2017/18

Fabian Kuhn

Computing Prefix Sums

Theorem: Given a sequence $a_1, \dots, a_n$ of n values, all prefix sums $s_i = a_1 \oplus \cdots \oplus a_i$ (for $1 \leq i \leq n$) can be computed in time $O(\log n)$ using $O(n/\log n)$ processors on an EREW PRAM.

Proof:

- Computing the sums of all sub-trees can be done in parallel in time $O(\log n)$ using $O(n)$ total operations.
- The same is true for the top-down step to compute the $r(v)$
- The theorem then follows from Brent's theorem:

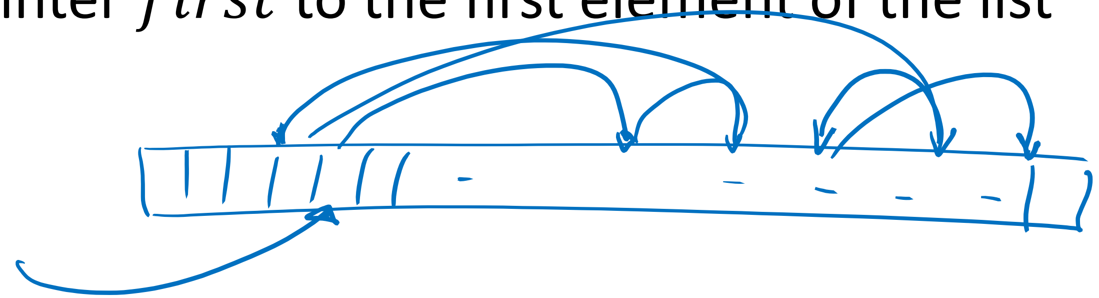
$$T_1 = O(n), \qquad T_\infty = O(\log n) \qquad \Longrightarrow \qquad T_p < T_\infty + \frac{T_1}{p}$$

Remark: This can be adapted to other parallel models and to different ways of storing the value (e.g., array or list)

Prefix Sums in Linked Lists

Given: Linked list L of length n in the following way

- Elements are in an array A of length n in an unordered way
- Each array element $A[i]$ also contains a next pointer
- Pointer $first$ to the first element of the list

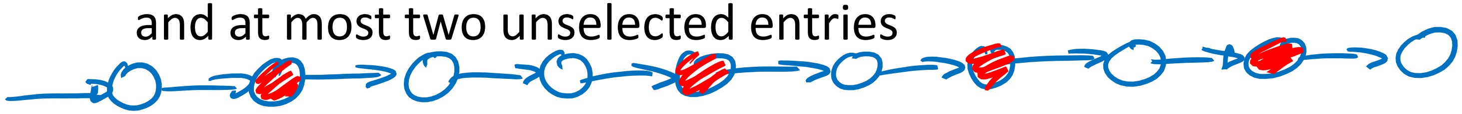


Goal: Compute all prefix sums w.r.t. to the order given by the list

2-Ruling Set of a Linked List

Given a linked list, select a subset of the entries such that

- No two neighboring entries are selected
- For every entry that is not selected, either the predecessor or the successor is selected
  - i.e., between two consecutive selected entries there are at least one and at most two unselected entries

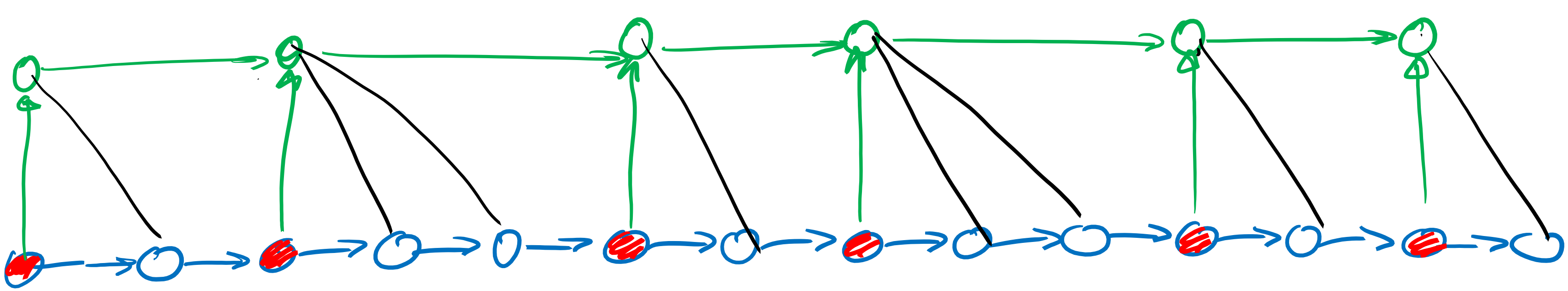


- We will see that a 2-ruling set of a linked list can be computed efficiently in parallel

Using 2-Ruling Sets to Get Prefix Sums

Observations:

- To compute the prefix sums of an array/list of numbers, we need a binary tree such that the numbers are at the leaves and an in-order traversal of the tree gives the right order
- The algorithm can be generalized to non-binary trees



Using 2-Ruling Sets to Get Prefix Sums

Lemma: If a 2-Ruling Set of a list of length N can be computed in parallel with $w(N)$ work and $d(N)$ depth, all prefix sums of a list of length n can be computed in parallel with

- Work $O\big(w(n) + w(n/2) + w(n/4) + \cdots + w(1)\big)$
- Depth $O\big(d(n) + d(n/2) + d(n/4) + \cdots + d(1)\big)$

Proof Sketch:

Prefix Sums in Linked Lists

Log-Star Function:

- For $i \geq 1$: $\log_2^{(i)} x = \log_2\left(\log_2^{(i-1)} x\right)$, and $\log_2^{(0)} x = x$
- For $x > 2$: $\log^* x \coloneqq \min\left\{i : \log^{(i)} x \leq 2\right\}$, for $x \leq 2$: $\log^* x \coloneqq 1$

Lemma: A 2-ruling set of a linked list of length n can be computed in parallel with work $O(n \cdot \log^* n)$ and span $O(\log^* n)$.

- i.e., in time $O(\log^* n)$ using $O(n)$ processors
  - We will first see how to apply this and prove it afterwards…

Prefix Sums in Linked Lists

Lemma: A 2-ruling set of a linked list of length n can be computed in parallel with work $O(n \cdot \log^* n)$ and span $O(\log^* n)$.

Theorem: All prefix sums of a linked list of length n can be computed in parallel with total work $O(n \cdot \log^* n)$ and span $O(\log n \cdot \log^* n)$.

- i.e., in time $O(\log n \cdot \log^* n)$ using $O(n/\log n)$ processors.

Computing 2-Ruling Sets

- Instead of computing a 2-ruling set, we first compute a coloring of the list:
  - each list element gets a color s.t. adjacent elements get different colors
- Each element initially has a unique $\log n$-bit label in $\{1, \dots, N\}$
  - can be interpreted as an initial coloring with N colors

Algorithm runs in phases:

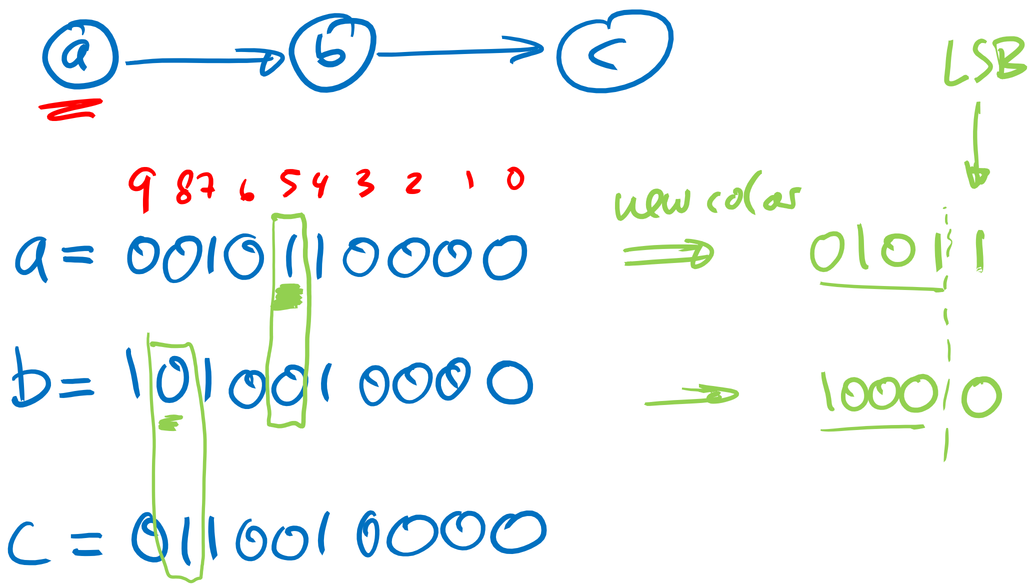
- Each phase: compute new coloring with smaller number of colors

We will show that

- #phases to get to $O(1)$ colors is $O(\log^* n)$
- each phase has $O(n)$ work and $O(1)$ depth

Reducing the number of colors

Assume that we start with a coloring with colors $\{0, \ldots, x-1\}$

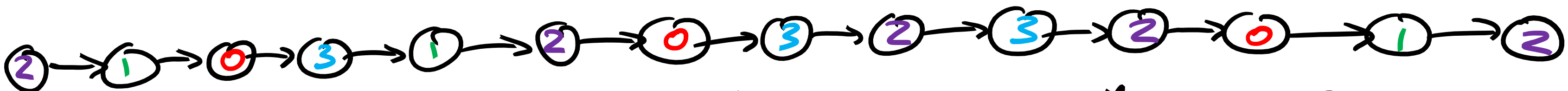


From a Coloring to a 2-Ruling Set S

Assume that we are given a coloring with colors $\{0, \dots, 5\}$

coloring → 2-ruling set: $O(n)$ work, $O(1)$ span

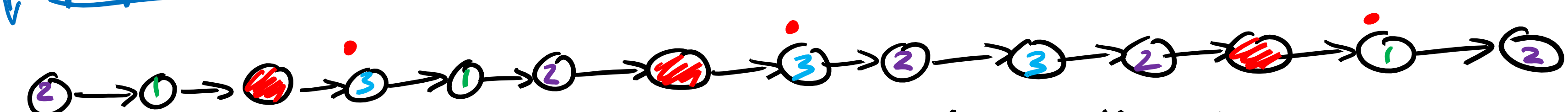
$0, \dots, x-1 \Rightarrow$ new max. color $\le \lfloor \log_2 x \rfloor \cdot 2 + 1$



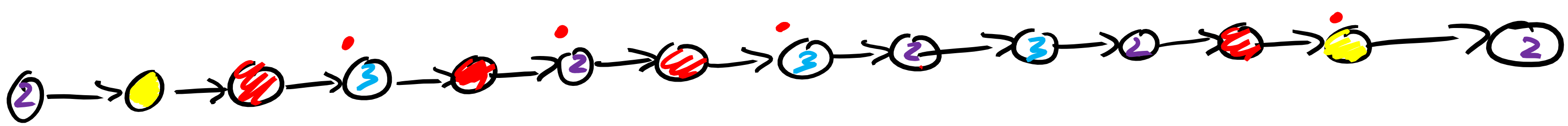
⇓ add entries with color 0

work $O(n)$

depth/span: $O(1)$



⇓ add elm. with color 1 & no neighbor in S



Prefix Sums in Linked Lists

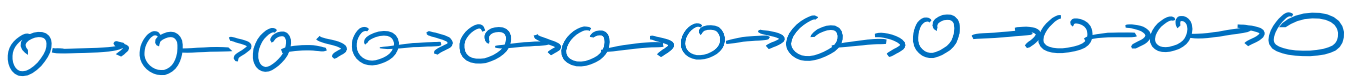
Lemma: A 2-ruling set of a linked list of length n can be computed in parallel with work $O(n \cdot \log^* n)$ and span $O(\log^* n)$.

Theorem: All prefix sums of a linked list of length n can be computed in parallel with total work $O(n \cdot \log^* n)$ and span $O(\log n \cdot \log^* n)$.

- i.e., in time $O(\log n \cdot \log^* n)$ using $O(n/\log n)$ processors.

List Ranking Problem: Compute the rank of each element of a linked list (rank: position in the list)

Distributed Coloring



synchronous rounds

compute 6-coloring in $O(\log^* n)$ rounds

↳ 3 colors

distributed coloring

given arb. graph $G=(V,E)$, typical goal: color with

$\Delta+1$ colors

↑

max. degree

a lot of results depending on

- Δ small or arbitr. large
- randomization?

Distributed Coloring