Date
Page

Lecture 19: 19th March

List Ranking

- given LL find distance of elements from one end of list

→ slow in parallel

Storing LL in parallel program

- using successor pointers
- using an array of successors S.

$|S| = |LL|$

S[i] = next of i in LL

S[last element] = -1

eg.

LL : 1 → 5 → 2 → 6 → 4 → 3

	1	2	3	4	5	6
S:	5	6	-1	3	2	4

head = value missing in S

→ Start loop : S[head] * fill S.

Ranking

Rank [i] = j if i is jth element of LL

R :

1	3	6	5	2	4
1	2	3	4	5	6

i.e. Rank is # elements incl. me to my left.

Pointer Jumping (Wyllie 1980)

head → 2 → 5 → 6 → 8 → 3 → 1 → 7 → 4 ↻

Find who is end node

for every u ← in parallel
  while S[u] ≠ S[S[u]]:
    S[u] = S[S[u]]

2 → 6 → 3 → 7 }
5 → 8 → 1 → 4 ↙ } after one iteration

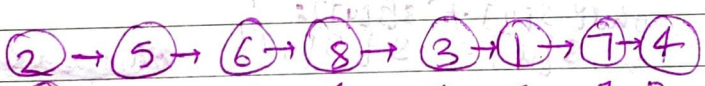
2 → 3 → 4 ←
6 → 7 ↗
5 → 1 ↗ 8 } second

5 →
1 → 4 ← 8 } third
2 ↗ ↑ ↑ ↖
3 6

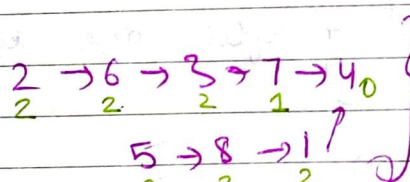
Extend this algo to find Ranks from end.

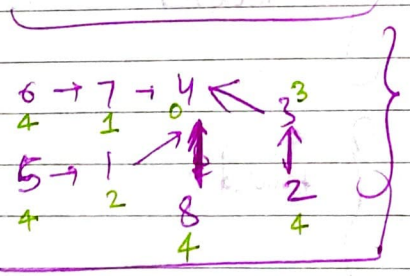
Algo FindRoot
for $1 \le i \le n$ do in parallel
  $R(i) = 1$
  $R(i) = 0$ if node i is last node
  while $S[i] \neq S[S[i]]$ do
    $R[i] = R[i] + R[S[i]]$
    $S[i] = S[S[i]]$
  end
end

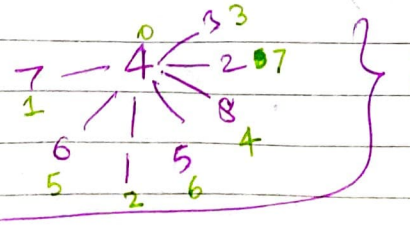
my rank + my parents rank



Rank







→ In PRAM → syn execution – all n processors execute each step in while loop at same time

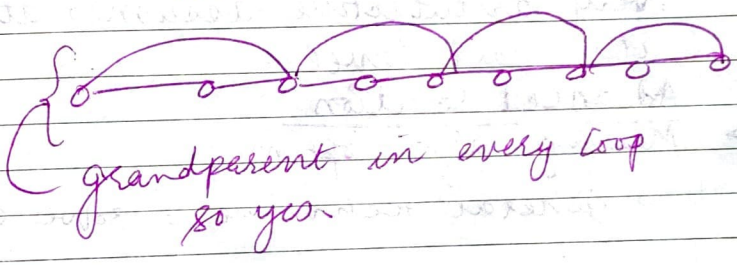
Trouble 1

→ use synchronisation primitives to avoid sync issues

Claim: Algo Find root finishes in $O(\log n)$ time

Proof: Show that the distance b/w a node and root reduces by a factor of 2 in every iteration of while loop.

↳ maximum distance is n



Claim = Work = $O(n \log n)$

In sequential time comp = $O(n)$

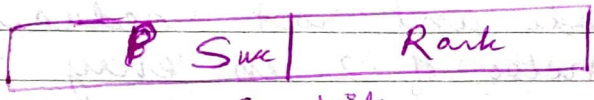
so, sub-optimal algo

Trouble 2 = Inconsistent Update

↳ so, in while loop either both statements should get executed for a given i or none.

So, consider packing R and P values of a node into a single word

If list has no more than 2^{32} elements, we use 64 bit architectures with each word packing two 32 bit numbers.



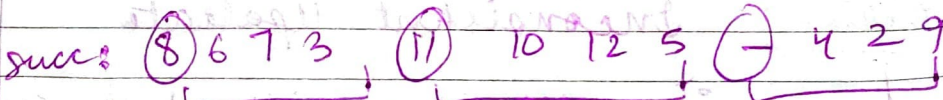
64 bits

every architecture assumes atomicity at word level.

Advanced Solution

→ Making algo optimal

- General technique = solve smaller ↓ extend soln to larger problem
- Input - array of Successor pointers
  so reducing array size to get smaller problem — won't give valid LL.



So to identify a sublist of size $\frac{n}{\log n}$ where we cannot take equi-distant parts of array

→ However, we can pick independent nodes (nodes = mutual non-neighbours)

LL : 1 → (8) → 5 → 11 → (2) → 6 → (10) → 4 → 3 → (7) → 12 → 9

these nodes can be removed in parallel & not any problem and we will get a smaller list

→ So - now we need to look at
① finding independent nodes
② small enough ?
③ extend soln to larger problem

→ INDEPENDENT SET
↳ vertices mutually non-neighbours

So, when we remove some elements from LL, and apply our algorithm (suboptimal)

We assign ranks as follows :

Step 1:
Initially

1 → 8 → 5 → 11 → 2 → 6 → 10 → 4 → 3 → 7 → 12 → 9

when elements are removed:

1 → 5 → 11 → 6 → 4 → 3 → 12 → 9
R: 1 2 1 2 2 1 2 1

we transfer their rank along to successor during removal, to represent the deleted elements.

Step 2:
Now, Rank the new LL

1 → 5 → 11 → 6 → 4 → 3 → 12 → 9
R: 1 3 4 6 8 9 11 12

↑ In this case, ranks are from Head not tail

Step 3: Finally, we reintroduce the removed elements

1 → 5 → 11 → 6 → 4 → 3 → 12 → 9
R: (1) (3) (4) (6) (8) (9) (11) (12)
8 2 10 7
(3-1=2) (6-1=5) (8-1=7) (11-1=10)