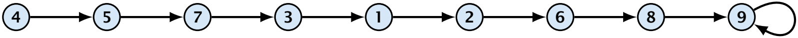
List Ranking

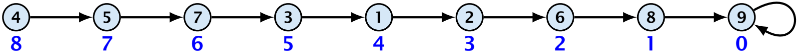
Input:

A list given by successor pointers;



Output:

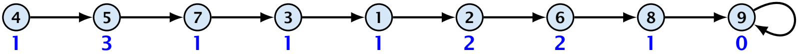
For every node number of hops to end of the list;



Observation:

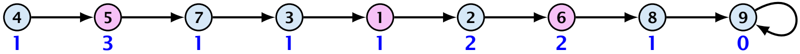
Special case of parallel prefix

List Ranking



1. Given a list with values; perhaps from previous iterations.
   The list is given via predecessor pointers $P(i)$ and successor pointers $S(i)$.
   $S(4) = 5$, $S(2) = 6$, $P(3) = 7$, etc.

List Ranking

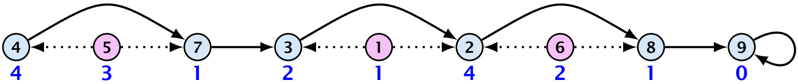


2. Find an independent set; time: $\mathcal{O}(\log n)$; work: $\mathcal{O}(n)$.

   The independent set should contain a constant fraction of the vertices.

   Color vertices; take local minima

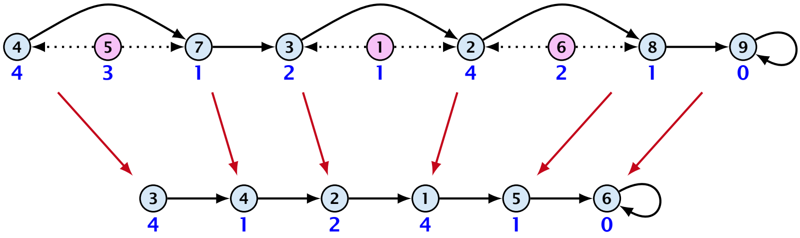
List Ranking



3. Splice the independent set out of the list;

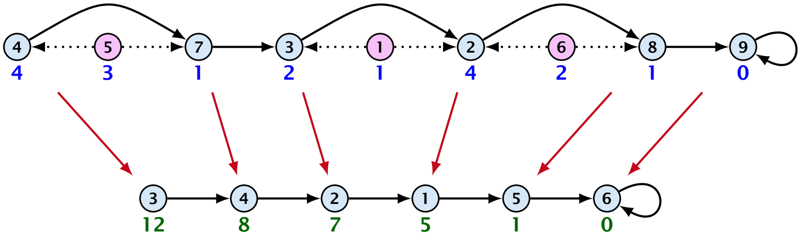
At the independent set vertices the array still contains old values for $P(i)$ and $S(i)$;

List Ranking



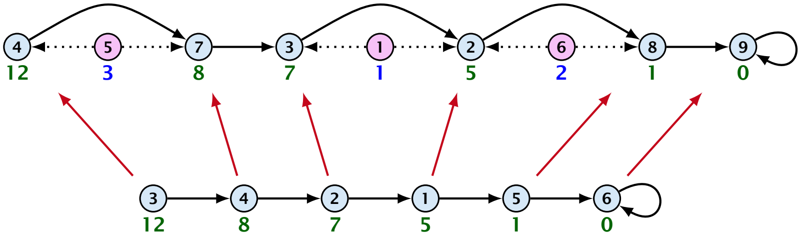
4. Compress remaining n' nodes into a new array of n' entries.
   The index positions can be computed by a prefix sum in time $\mathcal{O}(\log n)$ and work $\mathcal{O}(n)$
   Pointers can then be adjusted in time $\mathcal{O}(1)$.

List Ranking



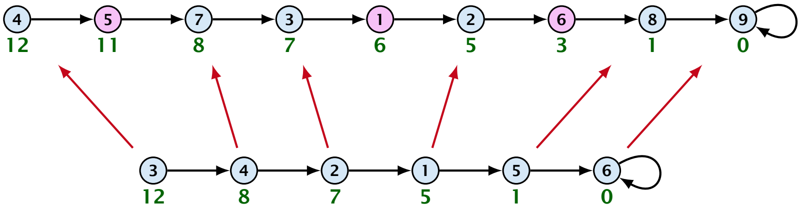
5. Solve the problem on the remaining list.
If current size is less than $n/\log n$ do pointer jumping: time $\mathcal{O}(\log n)$; work $\mathcal{O}(n)$.
Otherwise continue shrinking the list by finding an independent set

List Ranking



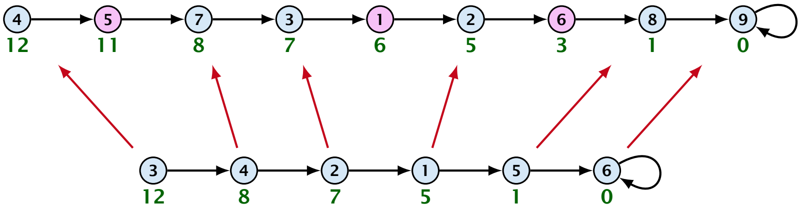
6. Map the values back into the larger list. Time: $\mathcal{O}(1)$; Work: $\mathcal{O}(n)$

List Ranking



7. Compute values for independent set nodes. Time: $\mathcal{O}(1)$; Work: $\mathcal{O}(1)$.
8. Splice nodes back into list. Time: $\mathcal{O}(1)$; Work: $\mathcal{O}(1)$.

List Ranking



We need $\mathcal{O}(\log\log n)$ shrinking iterations until the size of the remaining list reaches $\mathcal{O}(n/\log n)$.

Each shrinking iteration takes time $\mathcal{O}(\log n)$.

The work for all shrinking operations is just $\mathcal{O}(n)$, as the size of the list goes down by a constant factor in each round.

List Ranking can be solved in time $\mathcal{O}(\log n \log\log n)$ and work $\mathcal{O}(n)$ on an EREW-PRAM.

We need $\mathcal{O}(\log\log n)$ shrinking iterations until the size of the remaining list reaches $\mathcal{O}(n/\log n)$.

Each shrinking iteration takes time $\mathcal{O}(\log n)$.

The work for all shrinking operations is just $\mathcal{O}(n)$, as the size of the list goes down by a constant factor in each round.

List Ranking can be solved in time $\mathcal{O}(\log n \log\log n)$ and work $\mathcal{O}(n)$ on an EREW-PRAM.

We need $\mathcal{O}(\log\log n)$ shrinking iterations until the size of the remaining list reaches $\mathcal{O}(n/\log n)$.

Each shrinking iteration takes time $\mathcal{O}(\log n)$.

The work for all shrinking operations is just $\mathcal{O}(n)$, as the size of the list goes down by a constant factor in each round.

List Ranking can be solved in time $\mathcal{O}(\log n \log\log n)$ and work $\mathcal{O}(n)$ on an EREW-PRAM.

We need $\mathcal{O}(\log\log n)$ shrinking iterations until the size of the remaining list reaches $\mathcal{O}(n/\log n)$.

Each shrinking iteration takes time $\mathcal{O}(\log n)$.

The work for all shrinking operations is just $\mathcal{O}(n)$, as the size of the list goes down by a constant factor in each round.

List Ranking can be solved in time $\mathcal{O}(\log n \log\log n)$ and work $\mathcal{O}(n)$ on an EREW-PRAM.

Optimal List Ranking

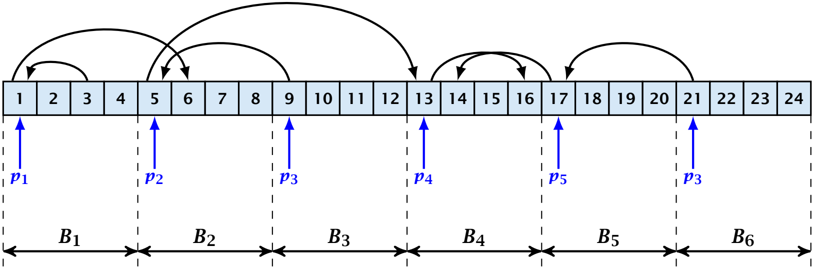
In order to reduce the work we have to improve the shrinking of the list to $\mathcal{O}(n/\log n)$ nodes.

After this we apply pointer jumping

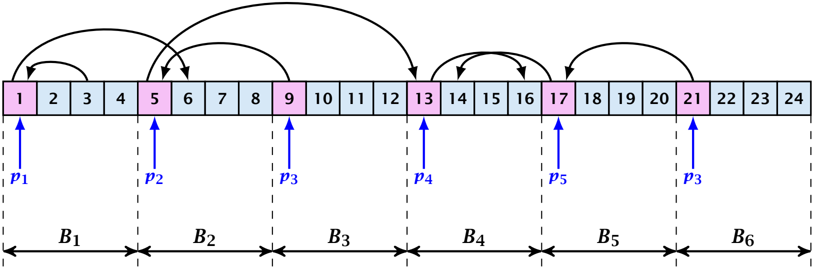
Optimal List Ranking

In order to reduce the work we have to improve the shrinking of the list to $\mathcal{O}(n/\log n)$ nodes.

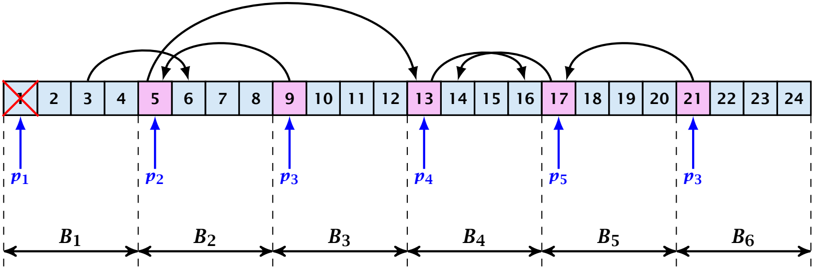
After this we apply pointer jumping



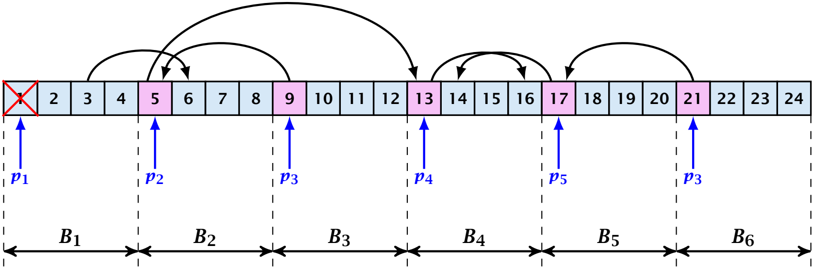
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
p_1 p_2 p_3 p_4 p_5 p_3
B_1 B_2 B_3 B_4 B_5 B_6



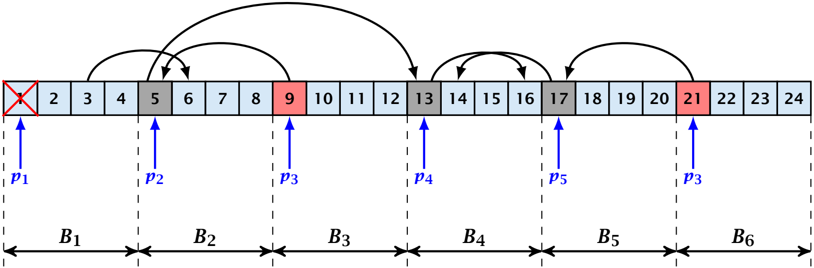
- some nodes are active;
- active nodes without neighbouring active nodes are isolated;
- the others form sublists;



1 delete isolated nodes from the list;



1. delete isolated nodes from the list;
2. color each sublist with $\mathcal{O}(\log\log n)$ colors; time: $\mathcal{O}(1)$; work: $\mathcal{O}(n)$;

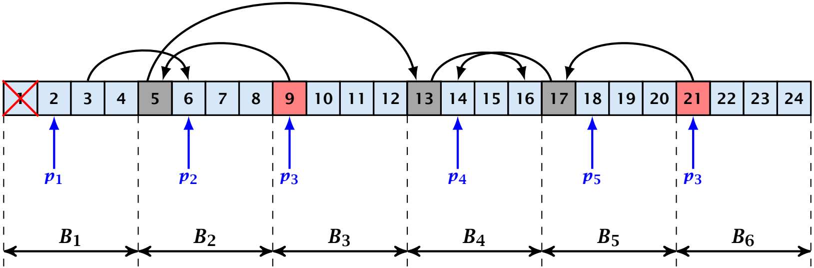


1 delete isolated nodes from the list;

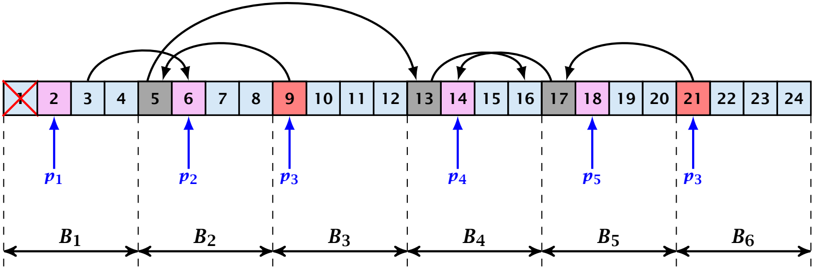
2 color each sublist with $\mathcal{O}(\log\log n)$ colors; time: $\mathcal{O}(1)$; work: $\mathcal{O}(n)$;

label local minima w.r.t. color as ruler; others as subject

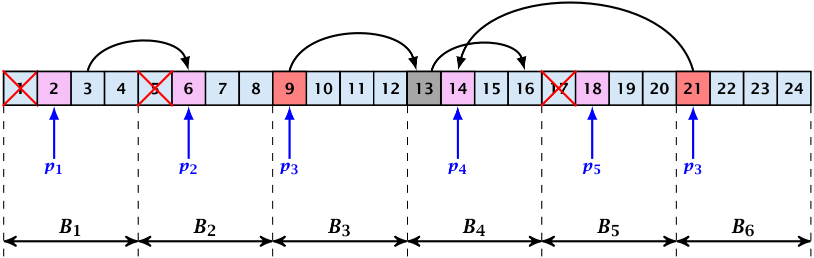
first node of sublist is ruler; needs to be changed!!!



3 advance pointers of removed nodes and of subjects;



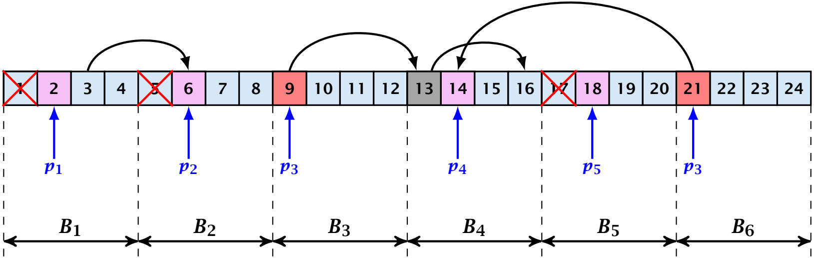
3 advance pointers of removed nodes and of subjects; make new nodes active



New Iteration

0. every ruler deletes its next subject;

   rulers without a subject become active


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
p_1
p_2
p_3
p_4
p_5
p_3
B_1
B_2
B_3
B_4
B_5
B_6

Optimal List Ranking

Each iteration requires constant time and work $\mathcal{O}(n/\log n)$, because we just work on one node in every block.

We need to prove that we just require $\mathcal{O}(\log n)$ iterations to reduce the size of the list to $\mathcal{O}(n/\log n)$.

Optimal List Ranking

Each iteration requires constant time and work $\mathcal{O}(n/\log n)$, because we just work on one node in every block.

We need to prove that we just require $\mathcal{O}(\log n)$ iterations to reduce the size of the list to $\mathcal{O}(n/\log n)$.

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either
  - becomes a ruler (with a subject)
  - becomes a subject
  - is isolated and therefore gets deleted

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either
  - becomes a ruler (with a subject)
  - becomes a subject
  - is isolated and therefore gets deleted

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either
  - becomes a ruler (with a subject)
  - becomes a subject
  - is isolated and therefore gets deleted

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either
  - becomes a ruler (with a subject)
  - becomes a subject
  - is isolated and therefore gets deleted

Observations/Remarks:

- If the p-pointer of a block cannot be advanced without leaving the block, the processor responsible for this block simply stops working; all other blocks continue.
- The p-node of a block (the node p_i is pointing to) at the beginning of a round is either a ruler with a living subject or the node will become active during the round.
- The subject nodes always lie to the left of the p-node of the respective block (if it exists).

Measure of Progress:

- a ruler will delete a subject
- an active node either
  - becomes a ruler (with a subject)
  - becomes a subject
  - is isolated and therefore gets deleted

Analysis

For the analysis we assign a weight to every node in every block as follows.

Definition 1
The weight of the i-th node in a block is

$$(1-q)^i$$

with $q = \frac{1}{\log\log n}$, where the node-numbering starts from 0. Hence, a block has nodes $\{0, \ldots, \log n - 1\}$.

Analysis

For the analysis we assign a weight to every node in every block as follows.

Definition 1
The weight of the i-th node in a block is

$$(1-q)^i$$

with $q = \frac{1}{\log\log n}$, where the node-numbering starts from 0. Hence, a block has nodes $\{0, \dots, \log n - 1\}$.

Definition of Rulers

Properties:

- A ruler should have at most $\log\log n$ subjects.
- The weight of a ruler should be at most the weight of any of its subjects.
- Each ruler must have at least one subject.
- We must be able to remove the next subject in constant time.
- We need to make the ruler/subject decision in constant time.

Definition of Rulers

Properties:

- A ruler should have at most $\log\log n$ subjects.
- The weight of a ruler should be at most the weight of any of its subjects.
- Each ruler must have at least one subject.
- We must be able to remove the next subject in constant time.
- We need to make the ruler/subject decision in constant time.

Definition of Rulers

Properties:

- A ruler should have at most $\log\log n$ subjects.
- The weight of a ruler should be at most the weight of any of its subjects.
- Each ruler must have at least one subject.
- We must be able to remove the next subject in constant time.
- We need to make the ruler/subject decision in constant time.

Definition of Rulers

Properties:

- A ruler should have at most $\log\log n$ subjects.
- The weight of a ruler should be at most the weight of any of its subjects.
- Each ruler must have at least one subject.
- We must be able to remove the next subject in constant time.
- We need to make the ruler/subject decision in constant time.

Definition of Rulers

Properties:

- A ruler should have at most $\log \log n$ subjects.
- The weight of a ruler should be at most the weight of any of its subjects.
- Each ruler must have at least one subject.
- We must be able to remove the next subject in constant time.
- We need to make the ruler/subject decision in constant time.

Given a sublist of active nodes.

Color the sublist with $\mathcal{O}(\log\log n)$ colors. Take the local minima w.r.t. this coloring.

If the first node is not a ruler

- if the second node is a ruler switch ruler status between first and second
- o/w just make the first node into a ruler

This partitions the sub-list into chains of length at most $\log\log n$ each starting with a ruler

Given a sublist of active nodes.

Color the sublist with $\mathcal{O}(\log\log n)$ colors. Take the local minima w.r.t. this coloring.

If the first node is not a ruler

- if the second node is a ruler switch ruler status between first and second
- o/w. just make the first node into a ruler

This partitions the sub-list into chains of length at most $\log\log n$ each starting with a ruler

Given a sublist of active nodes.

Color the sublist with $\mathcal{O}(\log\log n)$ colors. Take the local minima w.r.t. this coloring.

If the first node is not a ruler

- if the second node is a ruler switch ruler status between first and second
- otw. just make the first node into a ruler

This partitions the sub-list into chains of length at most $\log\log n$ each starting with a ruler

Given a sublist of active nodes.

Color the sublist with $\mathcal{O}(\log\log n)$ colors. Take the local minima w.r.t. this coloring.

If the first node is not a ruler

- if the second node is a ruler switch ruler status between first and second
- otw. just make the first node into a ruler

This partitions the sub-list into chains of length at most $\log\log n$ each starting with a ruler

Given a sublist of active nodes.

Color the sublist with $\mathcal{O}(\log\log n)$ colors. Take the local minima w.r.t. this coloring.

If the first node is not a ruler

- if the second node is a ruler switch ruler status between first and second
- otw. just make the first node into a ruler

This partitions the sub-list into chains of length at most $\log\log n$ each starting with a ruler

Now we change the ruler definition.

Consider some chain.

We make all local minima w.r.t. the weight function into a ruler; ties are broken according to block-id (so that comparing weights gives a strict inequality).

A ruler gets as subjects the nodes left of it until the next local maximum (or the start of the chain) (including the local maximum) and the nodes right of it until the next local maximum (or the end of the chain) (excluding the local maximum).

In case the first node is a ruler the above definition could leave it without a subject. We use constant time to fix this in some arbitrary manner

Now we change the ruler definition.

Consider some chain.

We make all local minima w.r.t. the weight function into a ruler; ties are broken according to block-id (so that comparing weights gives a strict inequality).

A ruler gets as subjects the nodes left of it until the next local maximum (or the start of the chain) (including the local maximum) and the nodes right of it until the next local maximum (or the end of the chain) (excluding the local maximum).

In case the first node is a ruler the above definition could leave it without a subject. We use constant time to fix this in some arbitrary manner

Now we change the ruler definition.

Consider some chain.

We make all local minima w.r.t. the weight function into a ruler; ties are broken according to block-id (so that comparing weights gives a strict inequality).

A ruler gets as subjects the nodes left of it until the next local maximum (or the start of the chain) (including the local maximum) and the nodes right of it until the next local maximum (or the end of the chain) (excluding the local maximum).

In case the first node is a ruler the above definition could leave it without a subject. We use constant time to fix this in some arbitrary manner

Now we change the ruler definition.

Consider some chain.

We make all local minima w.r.t. the weight function into a ruler; ties are broken according to block-id (so that comparing weights gives a strict inequality).

A ruler gets as subjects the nodes left of it until the next local maximum (or the start of the chain) (including the local maximum) and the nodes right of it until the next local maximum (or the end of the chain) (excluding the local maximum).

In case the first node is a ruler the above definition could leave it without a subject. We use constant time to fix this in some arbitrary manner

Now we change the ruler definition.

Consider some chain.

We make all local minima w.r.t. the weight function into a ruler; ties are broken according to block-id (so that comparing weights gives a strict inequality).

A ruler gets as subjects the nodes left of it until the next local maximum (or the start of the chain) (including the local maximum) and the nodes right of it until the next local maximum (or the end of the chain) (excluding the local maximum).

In case the first node is a ruler the above definition could leave it without a subject. We use constant time to fix this in some arbitrary manner

Set $q = \frac{1}{\log\log n}$.

The i-th node in a block is assigned a weight of $(1-q)^i$, $0 \le i < \log n$

The total weight of a block is at most $1/q$ and the total weight of all items is at most $\frac{n}{q \log n}$.

to show:
After $\mathcal{O}(\log n)$ iterations the weight is at most $(n/\log n)(1-q)^{\log n}$

This means at most $n/\log n$ nodes remain because the smallest weight a node can have is $(1-q)^{\log n - 1}$.

Set $q = \frac{1}{\log\log n}$.

The i-th node in a block is assigned a weight of $(1-q)^i$, $0 \le i < \log n$

The total weight of a block is at most $1/q$ and the total weight of all items is at most $\frac{n}{q \log n}$.

to show:
After $\mathcal{O}(\log n)$ iterations the weight is at most $(n/\log n)(1-q)^{\log n}$

This means at most $n/\log n$ nodes remain because the smallest weight a node can have is $(1-q)^{\log n - 1}$.

Set $q = \frac{1}{\log\log n}$.

The i-th node in a block is assigned a weight of $(1-q)^i$, $0 \le i < \log n$

The total weight of a block is at most $1/q$ and the total weight of all items is at most $\frac{n}{q\log n}$.

to show:
After $\mathcal{O}(\log n)$ iterations the weight is at most $(n/\log n)(1-q)^{\log n}$

This means at most $n/\log n$ nodes remain because the smallest weight a node can have is $(1-q)^{\log n - 1}$.

Set $q = \frac{1}{\log\log n}$.

The i-th node in a block is assigned a weight of $(1-q)^i$, $0 \le i < \log n$

The total weight of a block is at most $1/q$ and the total weight of all items is at most $\frac{n}{q \log n}$.

to show:

After $\mathcal{O}(\log n)$ iterations the weight is at most $(n/\log n)(1-q)^{\log n}$

This means at most $n/\log n$ nodes remain because the smallest weight a node can have is $(1-q)^{\log n - 1}$.

Set $q = \frac{1}{\log\log n}$.

The i-th node in a block is assigned a weight of $(1-q)^i$, $0 \le i < \log n$

The total weight of a block is at most $1/q$ and the total weight of all items is at most $\frac{n}{q \log n}$.

to show:

After $\mathcal{O}(\log n)$ iterations the weight is at most $(n/\log n)(1-q)^{\log n}$

This means at most $n/\log n$ nodes remain because the smallest weight a node can have is $(1-q)^{\log n - 1}$.

In every iteration the weight drops by a factor of

$$(1 - q/4) \ .$$

We consider subject nodes to just have half their weight.

We can view the step of becoming a subject as a precursor to deletion.

Hence, a node looses half its weight when becoming a subject and the remaining half when deleted.

Note that subject nodes will be deleted after just an additional $\mathcal{O}(\log \log n)$ iterations.

We consider subject nodes to just have half their weight.

We can view the step of becoming a subject as a precursor to deletion.

Hence, a node looses half its weight when becoming a subject and the remaining half when deleted.

Note that subject nodes will be deleted after just an additional $\mathcal{O}(\log \log n)$ iterations.

We consider subject nodes to just have half their weight.

We can view the step of becoming a subject as a precursor to deletion.

Hence, a node looses half its weight when becoming a subject and the remaining half when deleted.

Note that subject nodes will be deleted after just an additional $\mathcal{O}(\log\log n)$ iterations.

We consider subject nodes to just have half their weight.

We can view the step of becoming a subject as a precursor to deletion.

Hence, a node looses half its weight when becoming a subject and the remaining half when deleted.

Note that subject nodes will be deleted after just an additional $\mathcal{O}(\log \log n)$ iterations.

We consider subject nodes to just have half their weight.

We can view the step of becoming a subject as a precursor to deletion.

Hence, a node looses half its weight when becoming a subject and the remaining half when deleted.

Note that subject nodes will be deleted after just an additional $\mathcal{O}(\log\log n)$ iterations.

The weight is reduced because

- An isolated node is removed.
- A node is labelled as ruler, and the corresponding subjects reduce their weight by a factor of 1/2.
- A node is a ruler and deletes one of its subjects.

Hence, the weight reduction comes from p-nodes (ruler/active).

The weight is reduced because

- An isolated node is removed.
- A node is labelled as ruler, and the corresponding subjects reduce their weight by a factor of 1/2.
- A node is a ruler and deletes one of its subjects.

Hence, the weight reduction comes from p-nodes (ruler/active).

The weight is reduced because

- An isolated node is removed.
- A node is labelled as ruler, and the corresponding subjects reduce their weight by a factor of $1/2$.
- A node is a ruler and deletes one of its subjects.

Hence, the weight reduction comes from p-nodes (ruler/active).

The weight is reduced because

- An isolated node is removed.
- A node is labelled as ruler, and the corresponding subjects reduce their weight by a factor of 1/2.
- A node is a ruler and deletes one of its subjects.

Hence, the weight reduction comes from p-nodes (ruler/active).

The weight is reduced because

- An isolated node is removed.
- A node is labelled as ruler, and the corresponding subjects reduce their weight by a factor of 1/2.
- A node is a ruler and deletes one of its subjects.

Hence, the weight reduction comes from p-nodes (ruler/active).

Each p-node is responsible for some other nodes; it has to generate a weight reduction large enough so that the weight of all nodes it is responsible for decreases by the desired factor.

An active node is responsible for all nodes that come after it in its block.

A ruler is responsible for all nodes that come after it in its block **and** for all its subjects.

Note that by this definition every node remaining in the list is covered.

Each p-node is responsible for some other nodes; it has to generate a weight reduction large enough so that the weight of all nodes it is responsible for decreases by the desired factor.

An active node is responsible for all nodes that come after it in its block.

A ruler is responsible for all nodes that come after it in its block **and** for all its subjects.

Note that by this definition every node remaining in the list is covered.

Each p-node is responsible for some other nodes; it has to generate a weight reduction large enough so that the weight of all nodes it is responsible for decreases by the desired factor.

An active node is responsible for all nodes that come after it in its block.

A ruler is responsible for all nodes that come after it in its block **and** for all its subjects.

Note that by this definition every node remaining in the list is covered.

Each p-node is responsible for some other nodes; it has to generate a weight reduction large enough so that the weight of all nodes it is responsible for decreases by the desired factor.

An active node is responsible for all nodes that come after it in its block.

A ruler is responsible for all nodes that come after it in its block **and** for all its subjects.

Note that by this definition every node remaining in the list is covered.

Case 1: Isolated Node

Case 1: Isolated Node

Suppose we delete an isolated node v that is the i-th node in its block.

Case 1: Isolated Node

Suppose we delete an isolated node v that is the i-th node in its block.

The weight of all node that v is responsible for is

$$\sum_{i \le j < \log n} (1-q)^j$$

Case 1: Isolated Node

Suppose we delete an isolated node v that is the i-th node in its block.

The weight of all node that v is responsible for is

$$\sum_{i \le j < \log n} (1-q)^j$$

This weight reduces to

$$\sum_{i < j < \log n} (1-q)^j$$

Case 1: Isolated Node

Suppose we delete an isolated node v that is the i-th node in its block.

The weight of all node that v is responsible for is

$$\sum_{i \le j < \log n} (1-q)^j$$

This weight reduces to

$$\sum_{i < j < \log n} (1-q)^j \le (1-q) \sum_{i \le j < \log n} (1-q)^j$$

Case 1: Isolated Node

Suppose we delete an isolated node v that is the i-th node in its block.

The weight of all node that v is responsible for is

$$\sum_{i \le j < \log n} (1-q)^j$$

This weight reduces to

$$\sum_{i < j < \log n} (1-q)^j \le (1-q) \sum_{i \le j < \log n} (1-q)^j$$

Hence, weight reduces by a factor $(1-q) \le (1-q/4)$.

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j < \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le \left(1 - \frac{q}{4}\right) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell \le \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le \left(1 - \frac{q}{4}\right) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le (1 - \frac{q}{4}) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le \left(1 - \frac{q}{4}\right) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le \left(1 - \frac{q}{4}\right) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le (1 - \frac{q}{4}) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le \left(1 - \frac{q}{4}\right) Q$$

Case 2: Creating Subjects

Suppose we generate a ruler with at least one subject.

Weight of ruler: $(1-q)^{i_1}$.
Weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$.

Initial weight:

$$Q = \sum_{j=1}^{k} \sum_{i_j \le \ell < \log n} (1-q)^{\ell} \le \frac{1}{q} \sum_{j=1}^{k} (1-q)^{i_j} \le \frac{2}{q} \sum_{j=2}^{k} (1-q)^{i_j}$$

New weight:

$$Q' = Q - \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \le (1 - \frac{q}{4}) Q$$

Case 3: Removing Subjects

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \leq j \leq k$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$Q$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$Q = \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j}$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$
\begin{aligned}
Q &= \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \\
&\le \frac{1}{q}(1-q)^{i_1} + \frac{k}{2}(1-q)^{i_2}
\end{aligned}
$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$
\begin{aligned}
Q &= \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2}\sum_{j=2}^{k}(1-q)^{i_j} \\
&\le \frac{1}{q}(1-q)^{i_1} + \frac{k}{2}(1-q)^{i_2} \\
&\le \frac{1}{q}(1-q)^{i_2} + \frac{1}{2q}(1-q)^{i_2}
\end{aligned}
$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$
\begin{aligned}
Q &= \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2}\sum_{j=2}^{k}(1-q)^{i_j} \\
&\le \frac{1}{q}(1-q)^{i_1} + \frac{k}{2}(1-q)^{i_2} \\
&\le \frac{1}{q}(1-q)^{i_2} + \frac{1}{2q}(1-q)^{i_2}
\end{aligned}
$$

New weight:

$$Q'$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$
\begin{aligned}
Q &= \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2} \sum_{j=2}^{k} (1-q)^{i_j} \\
&\le \frac{1}{q}(1-q)^{i_1} + \frac{k}{2}(1-q)^{i_2} \\
&\le \frac{1}{q}(1-q)^{i_2} + \frac{1}{2q}(1-q)^{i_2}
\end{aligned}
$$

New weight:

$$
Q' = Q - \frac{1}{2}(1-q)^{i_2}
$$

Case 3: Removing Subjects

weight of ruler: $(1-q)^{i_1}$; weight of subjects: $(1-q)^{i_j}$, $2 \le j \le k$

Assume ruler removes subject with largest weight say i_2 (why?).

Initial weight:

$$
\begin{aligned}
Q &= \sum_{i_1 \le \ell < \log n} (1-q)^{\ell} + \frac{1}{2}\sum_{j=2}^{k} (1-q)^{i_j} \\
&\le \frac{1}{q}(1-q)^{i_1} + \frac{k}{2}(1-q)^{i_2} \\
&\le \frac{1}{q}(1-q)^{i_2} + \frac{1}{2q}(1-q)^{i_2}
\end{aligned}
$$

New weight:

$$
Q' = Q - \frac{1}{2}(1-q)^{i_2} \le (1 - \frac{q}{3})Q
$$

After s iterations the weight is at most

$$\frac{n}{q \log n}\left(1-\frac{q}{4}\right)^{s} \overset{!}{\leq} \frac{n}{\log n}(1-q)^{\log n}$$

Choosing $i = 5 \log n$ the inequality holds for sufficiently large n.