Scalable Hierarchical Clustering with Tree Grafting

Nicholas Monath*

Ari Kobren*

Akshay Krishnamurthy Microsoft Research

Michael Glass IBM

Andrew McCallum

IESL

UMass Amherst

College *of* Information and Computer Sciences

Microsoft Research

IBM

*The first two authors contributed equally.

Hierarchical Agglomerative Clustering

HAC: Widely-used, highly effective algorithm

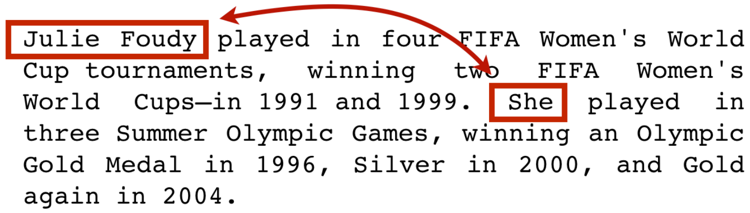
Record Linkage

Type Classes in Haskell. Hall, C. V. and Hammond, K. and Jones, S. and Wadler, P. *Programming Languages and Systems*. 1996.
Imperative Function Programming. Peyton Jones, S. and Wadler, P. *Principles of Programming Languages*. 1993.

The Implementer's Dilemma: A Mathematical Model of Compile Time Garbage Collection. Jones, S. and Tyas, A. *Functional Programming*. 1993.

[Bilenko et al, 2006], [Torvik et al, 2009], [Levin et al, 2012], [Fleming et al, 2015], [Ventura et al, 2014, 2015], [Mamun et al, 2016], [Vashishth et al, 2018]

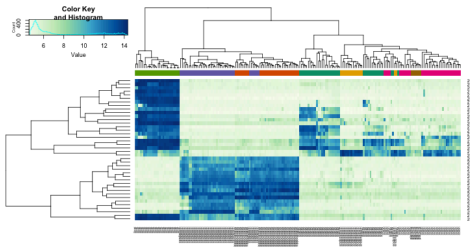
Coreference



```
Julie Foudy played in four FIFA Women's World
Cup tournaments,  winning  two  FIFA  Women's
World  Cups—in 1991 and 1999.  She  played  in
three Summer Olympic Games, winning an Olympic
Gold Medal in 1996, Silver in 2000, and Gold
again in 2004.
```

[Bagga and Baldwin, 1998], [Mann and Yarowsky, 2003] Gooi and Allan, 2004; Chen and Martin, 2007], [Green et al, 2012],[Clark & Manning, 2016], [Kenyon-Dean et al 2018]

Biomedicine



[Irizarry & Love]

[Eisen et al, 1998], [Perou et al, 2000], [Alizadeh et al, 2000],[[Blaveri et al, 2005], [Freyhult, et al, 2010] [Linehan et al 2016], [Subramanian et al, 2017]

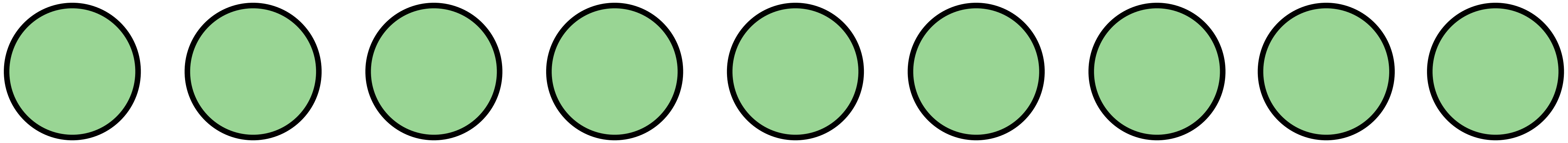
Inference for statistical models

[Heller & Ghahramani, 2006], [Teh et al, 2009], [Blundell et al, 2010], [Telgarsky & Dasgupta, 2012], [Blundell et al, 2013], [Hu et al, 2013], [Lee et al, 2015]

Theoretical Results

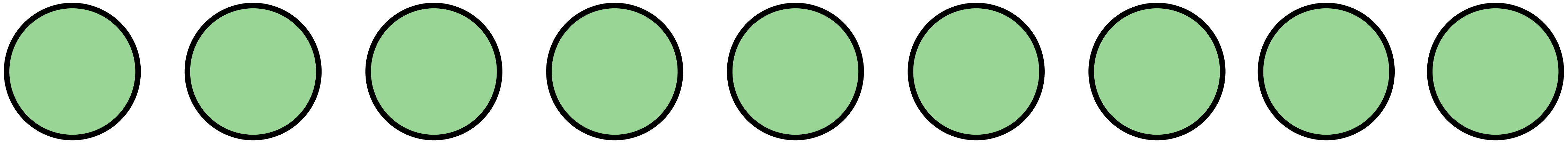
[Dasgupta, 2016], [Moseley & Wang, 2016], [Charikar & Chatziafratis, 2017], [Cohen-Addad, 2017, 2018], [Wang & Wang, 2018], [Emamjomeh-Zadeh & Kempe, 2018], [Charikar et al, 2019],

Hierarchical Agglomerative Clustering



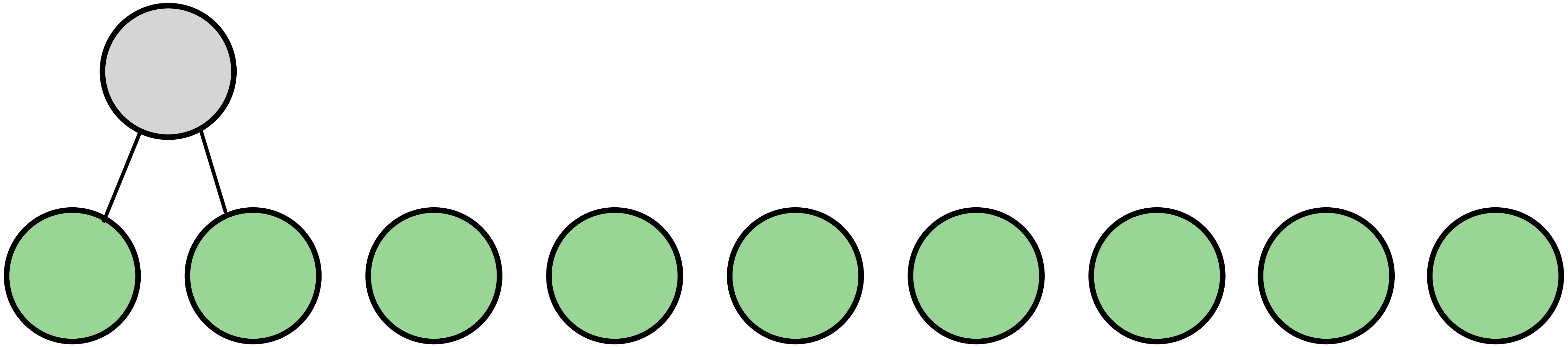
Hierarchical Agglomerative Clustering

```
while not complete_tree
  agglomerate(argmax $g(c_i, c_j)$)
```



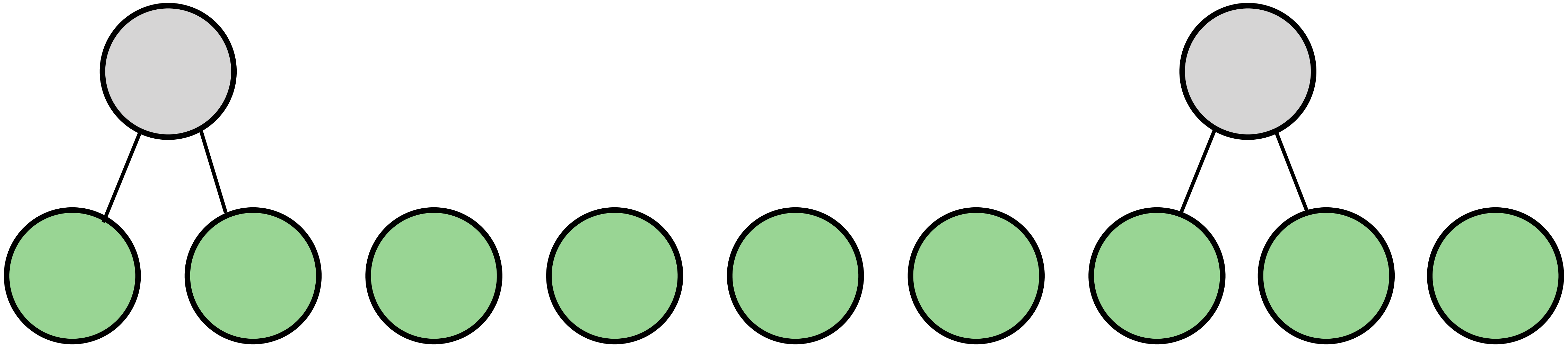
Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax g(c_i, c_j))
```



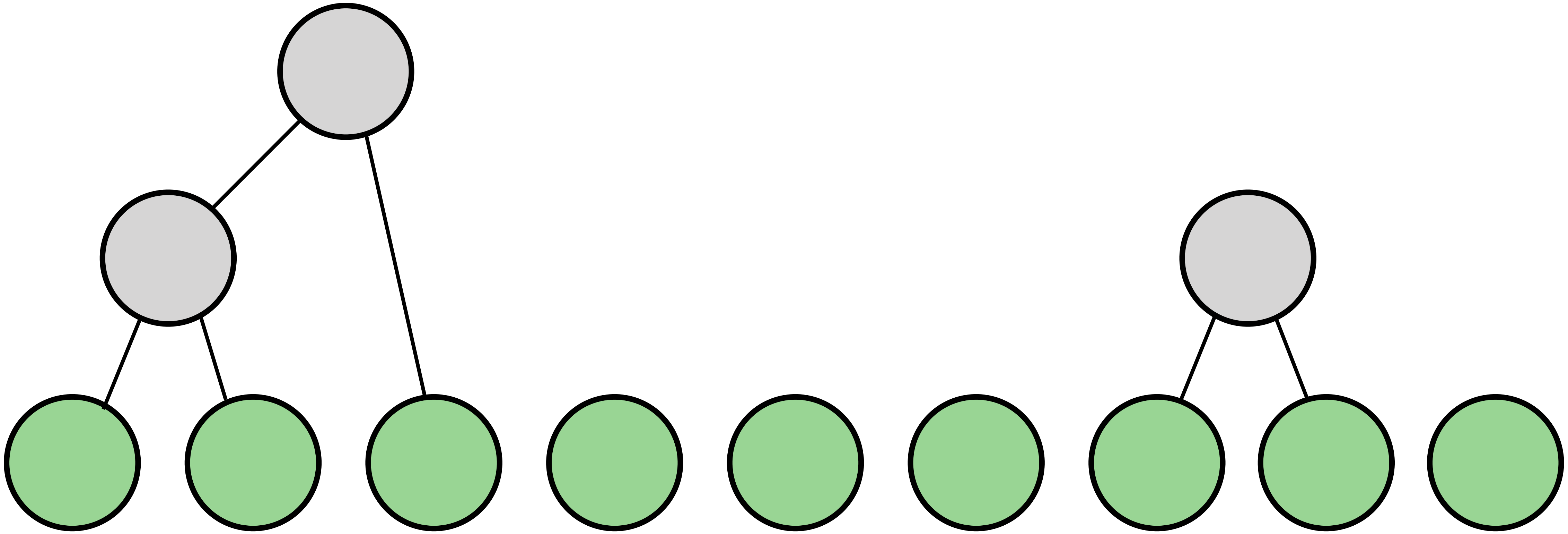
Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax g(c_i, c_j))
```



Hierarchical Agglomerative Clustering

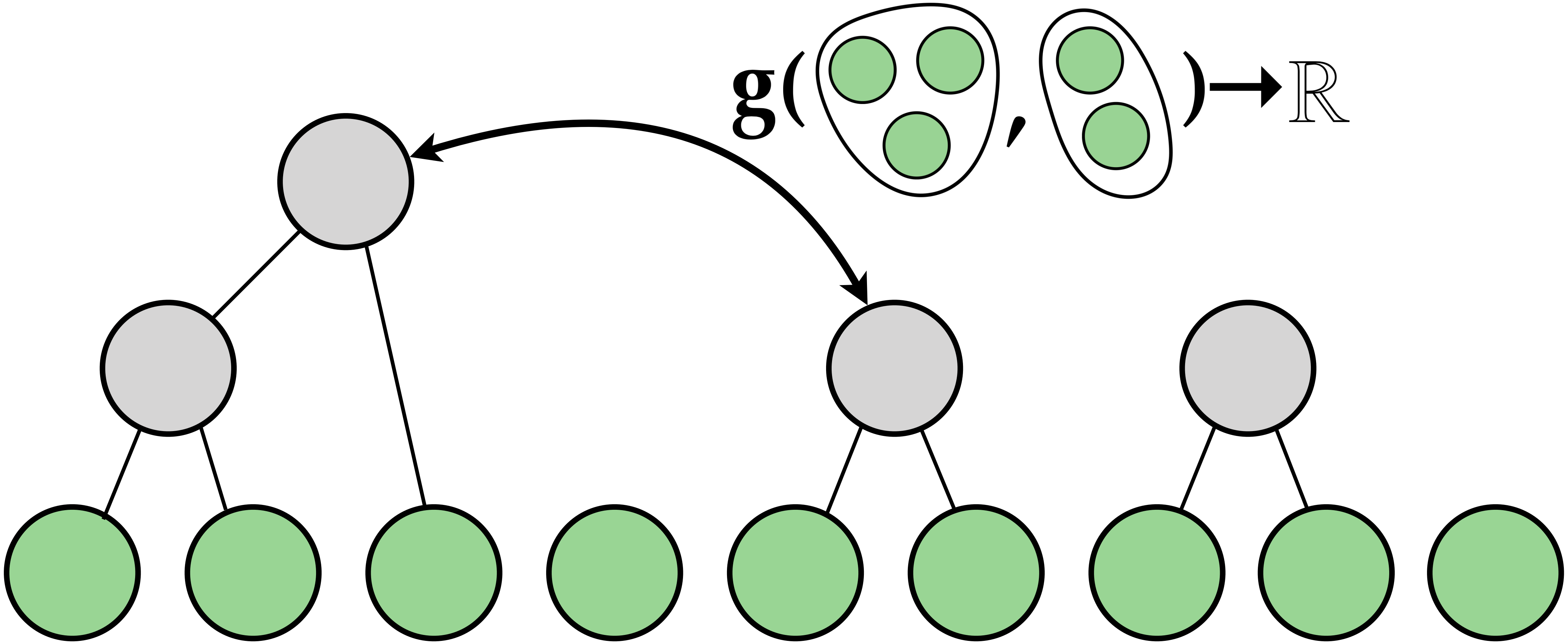
```
while not complete_tree
  agglomerate(argmax g(c_i, c_j))
```



Hierarchical Agglomerative Clustering

```
while not complete_tree
   agglomerate(argmax g($c_i$, $c_j$))
```

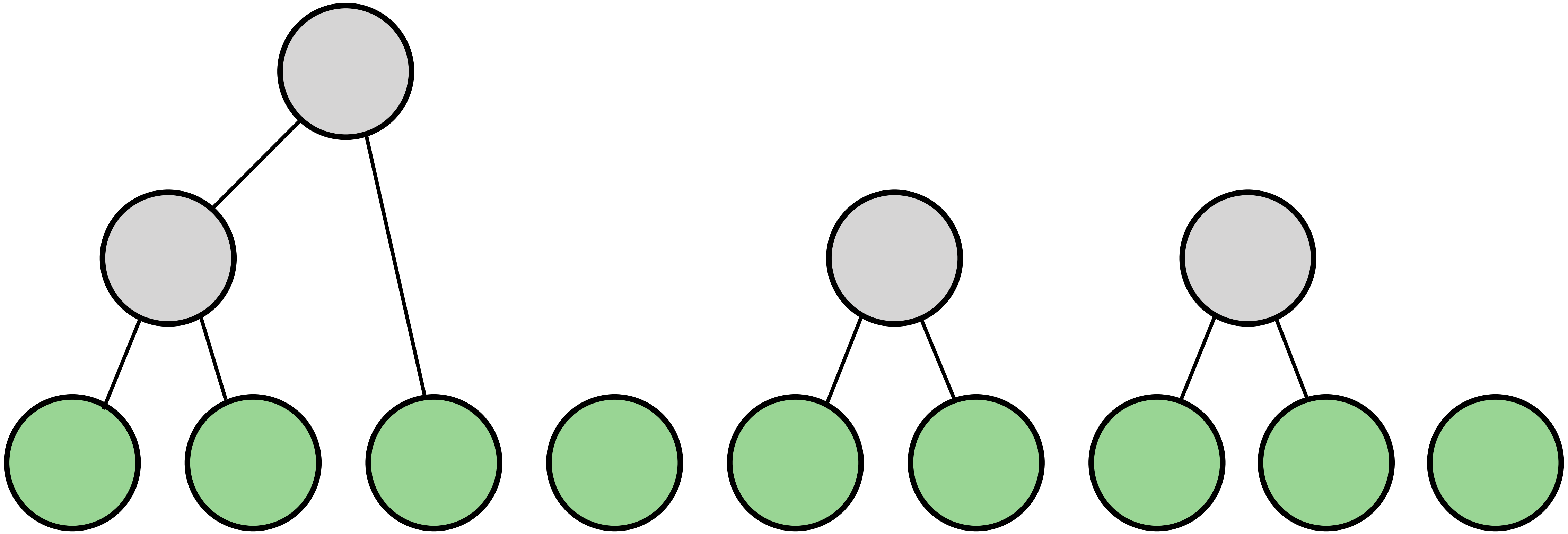
Use *any* linkage function **g**



Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax g(c_i, c_j))
```

However, there are **2 significant drawbacks**

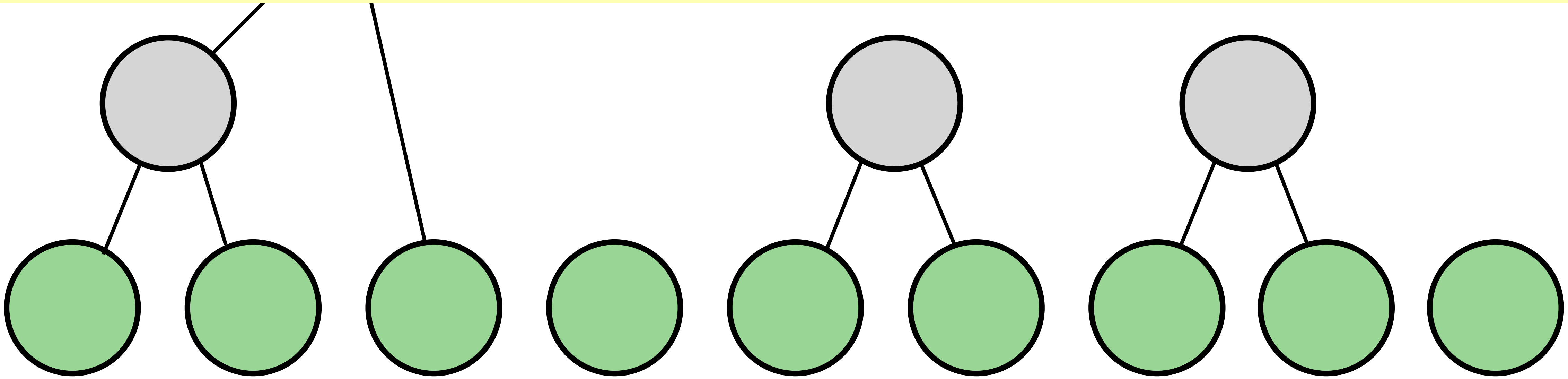


Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax $g(c_i, c_j)$)
```

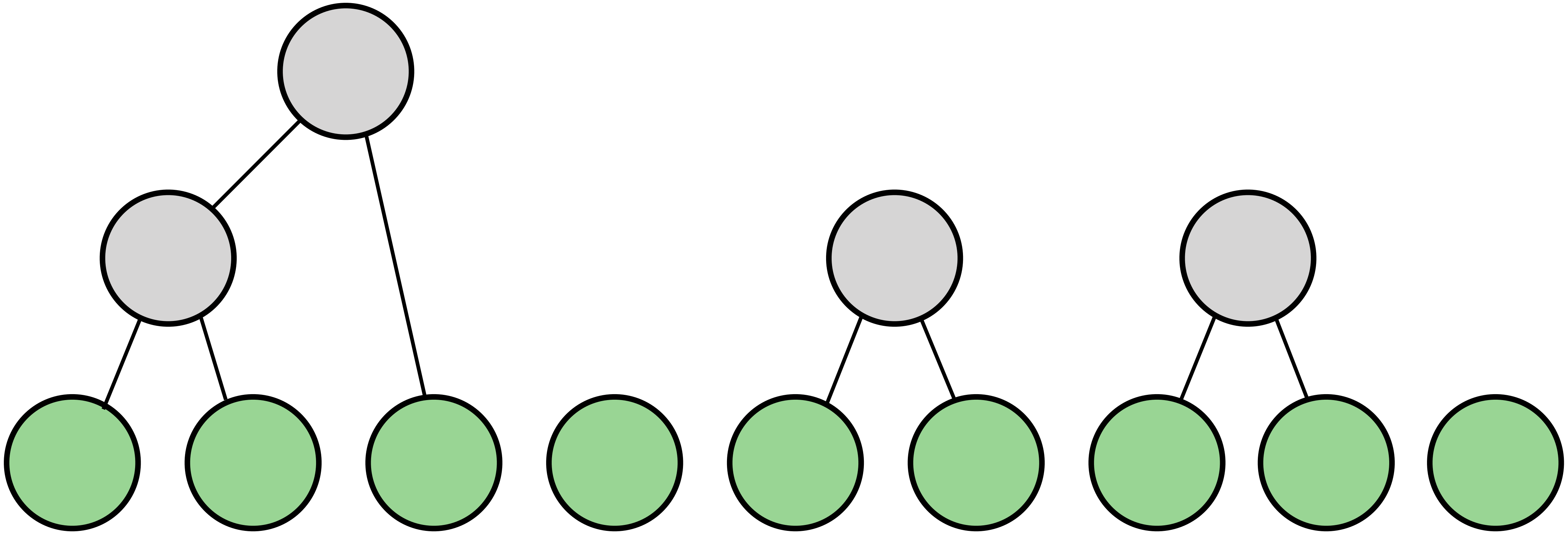
`argmax` can scale quadratically in num. points

Not scalable to large datasets, scales $O(N^2 \mathrm{log} N)$



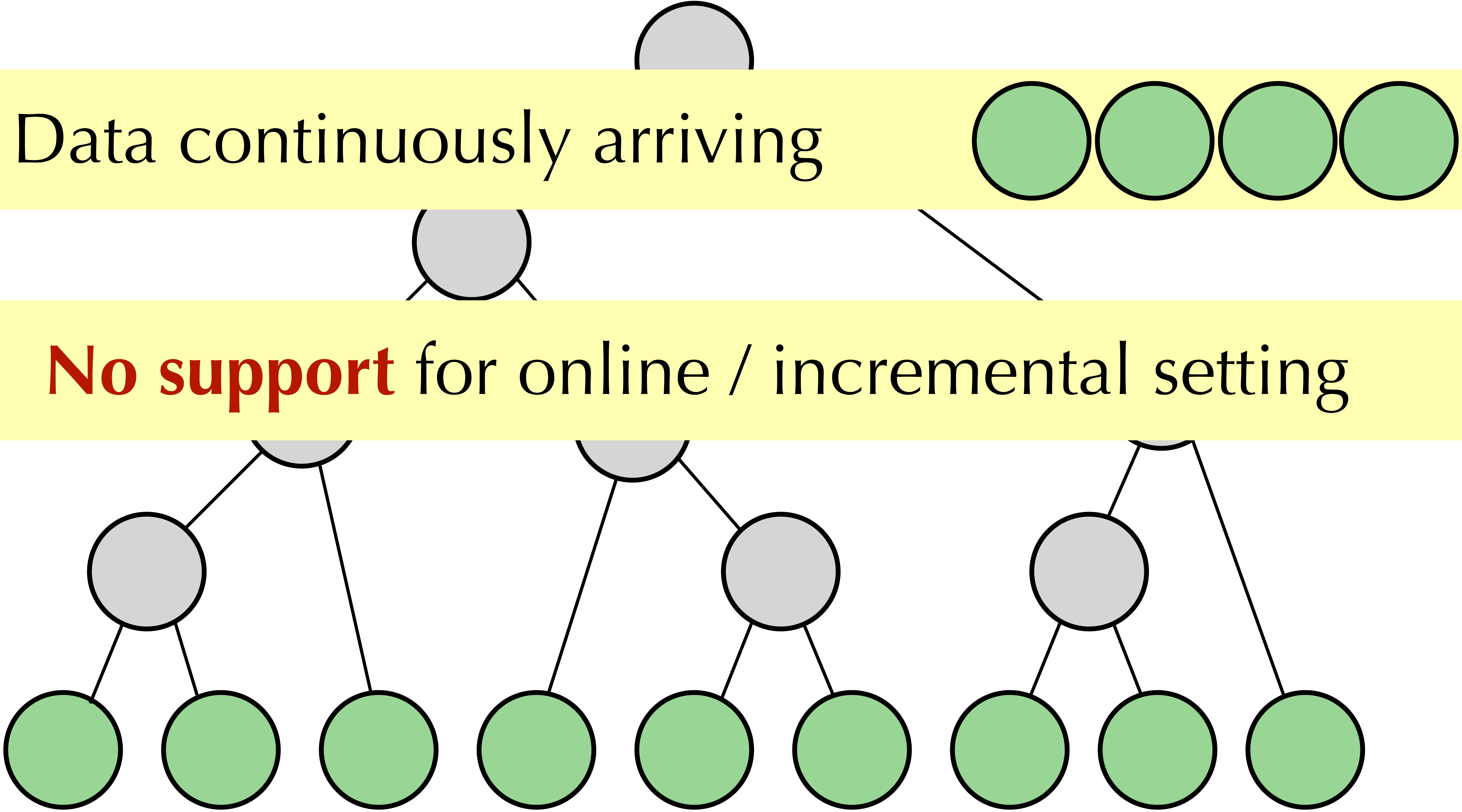
Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax $g(c_i, c_j)$)
```



Hierarchical Agglomerative Clustering

```
while not complete_tree
    agglomerate(argmax g($c_i$, $c_j$))
```



Data continuously arriving

No support for online / incremental setting

This Work

Scalable, incremental alterative to HAC

Support **any linkage**, discover **meaningful** clusterings

Theoretically motivated & **Empirically** effective.

Outline

1. Introduction

2. Proposed methodology

3. Experimental Results

4. Experimental Analysis

5. Theoretical Results

Outline

1. Introduction

2. Proposed methodology

3. Experimental Results

4. Experimental Analysis

5. Theoretical Results

GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

GRINCH

Grafting and Rotation-based INCremental Hierarchical clustering

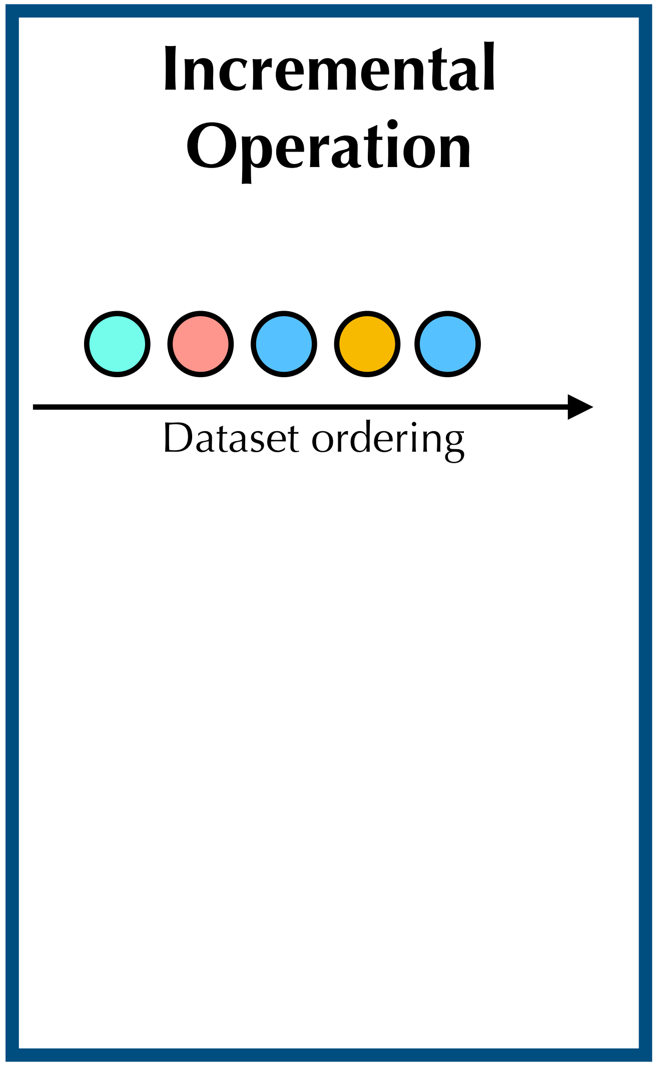
At a high level:

Incremental Operation

GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

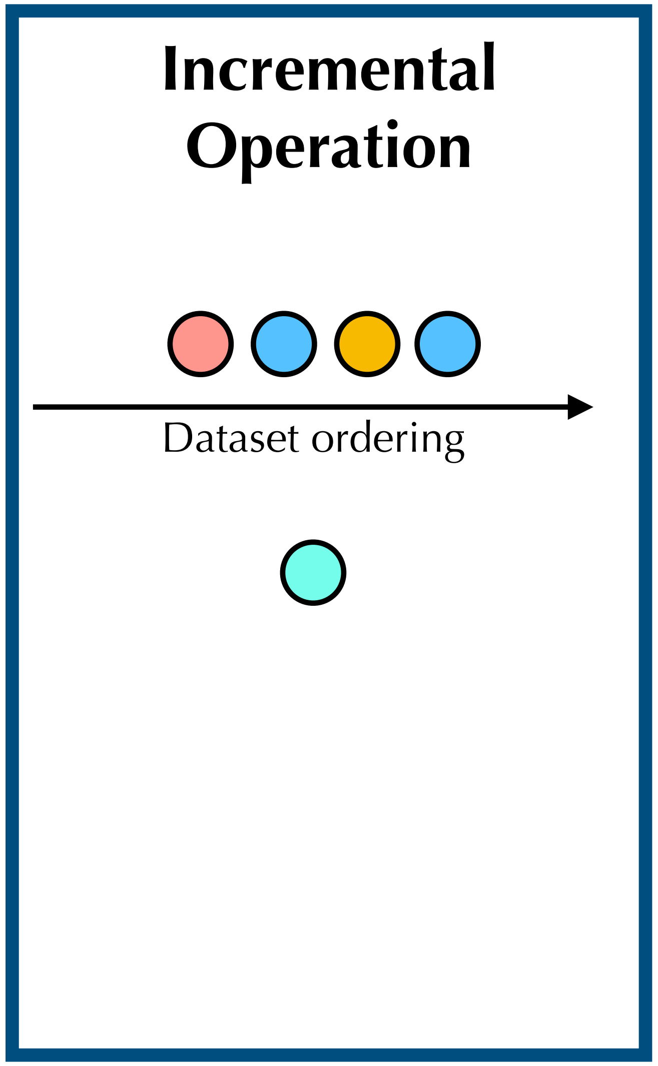
At a high level:



GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

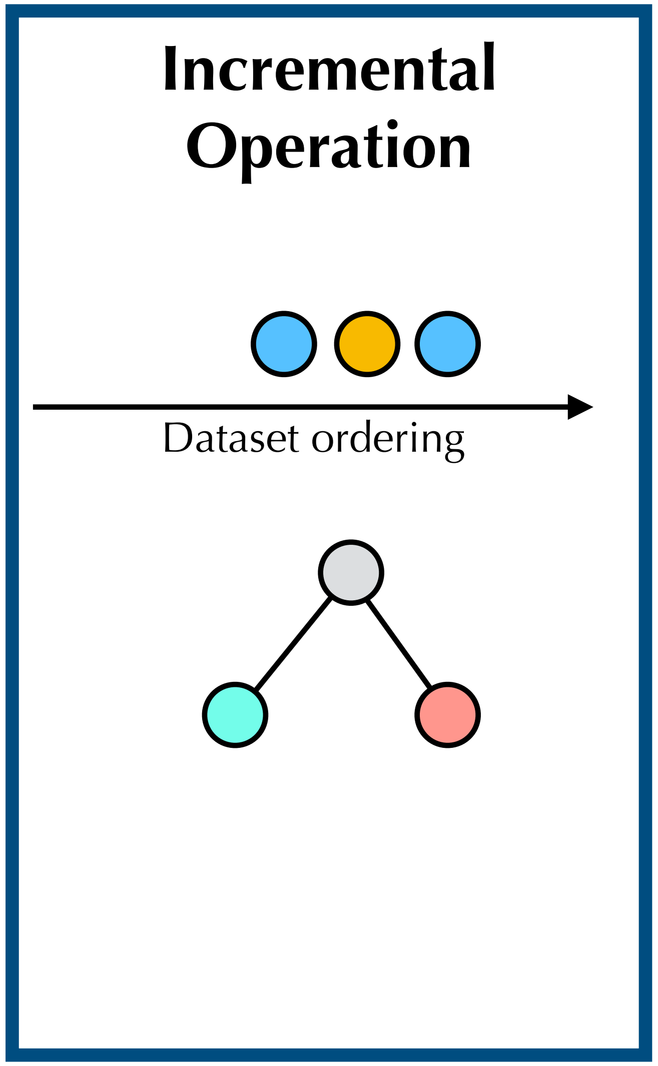
At a high level:



GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

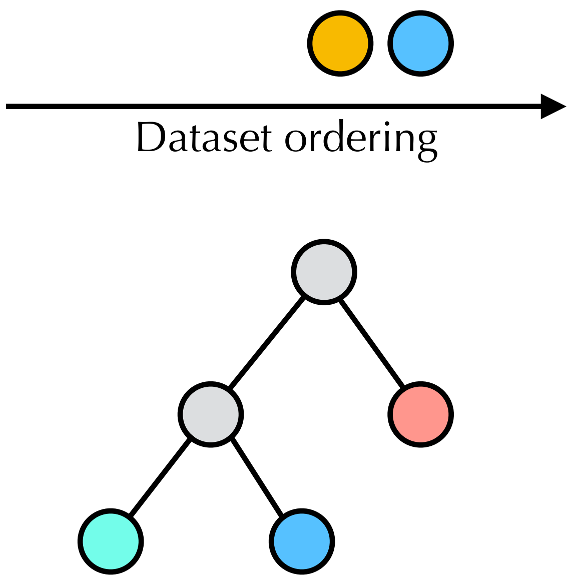


GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

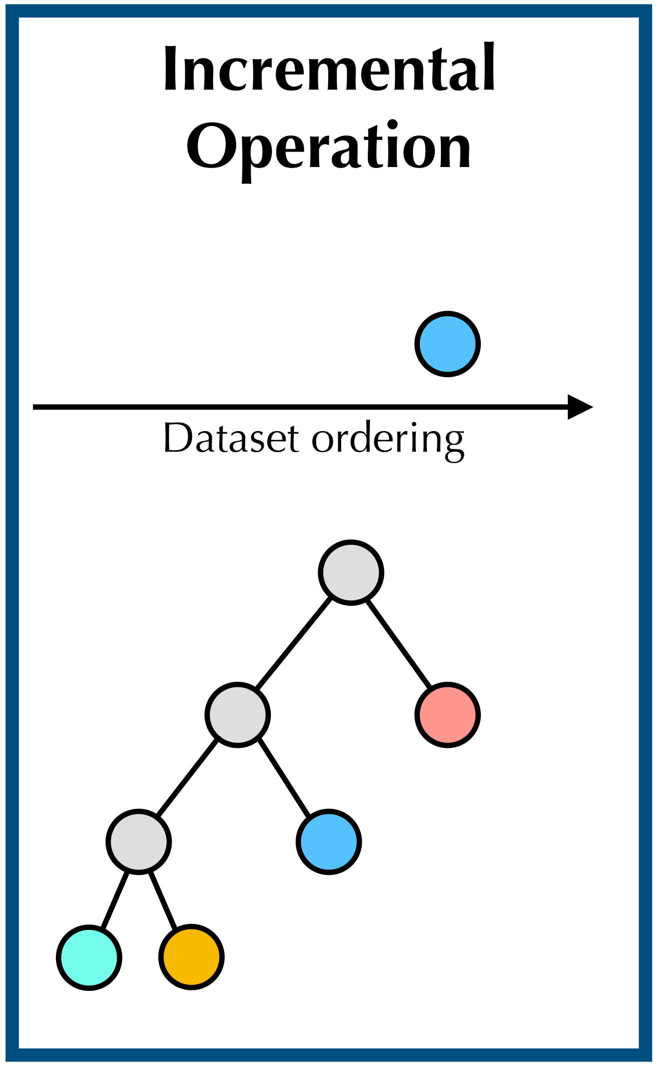
Incremental Operation



GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

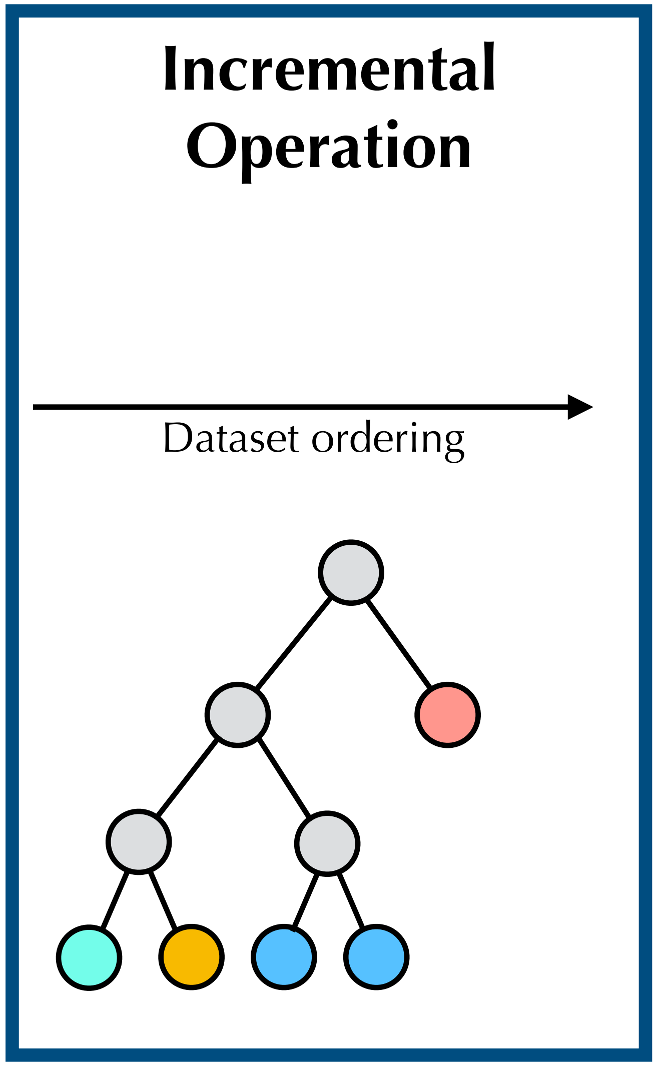
At a high level:



GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

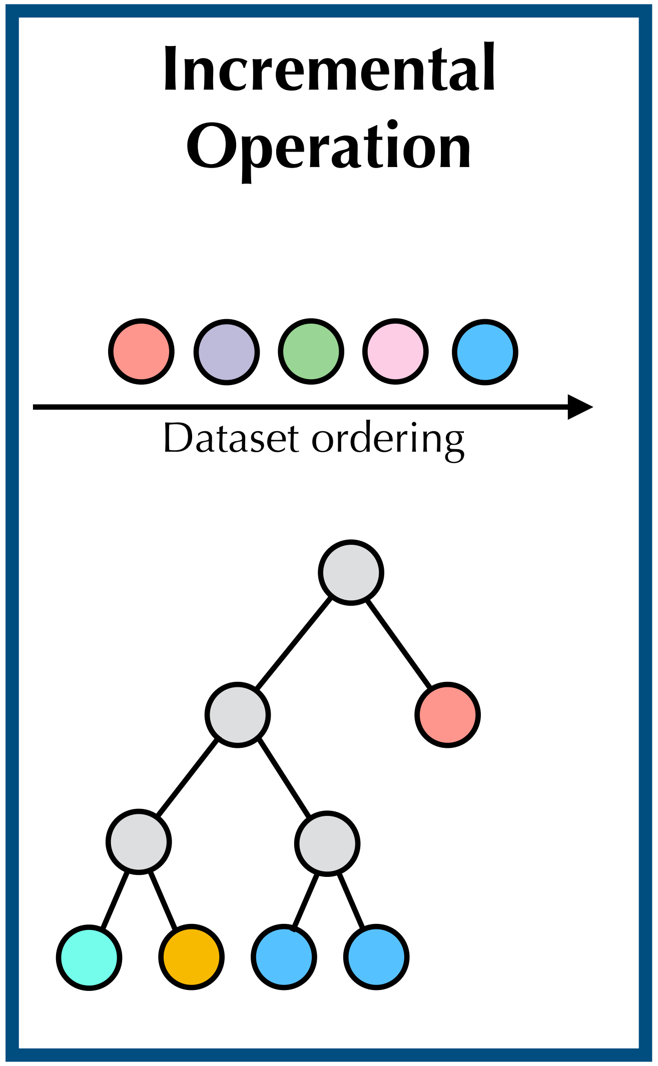
At a high level:



GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

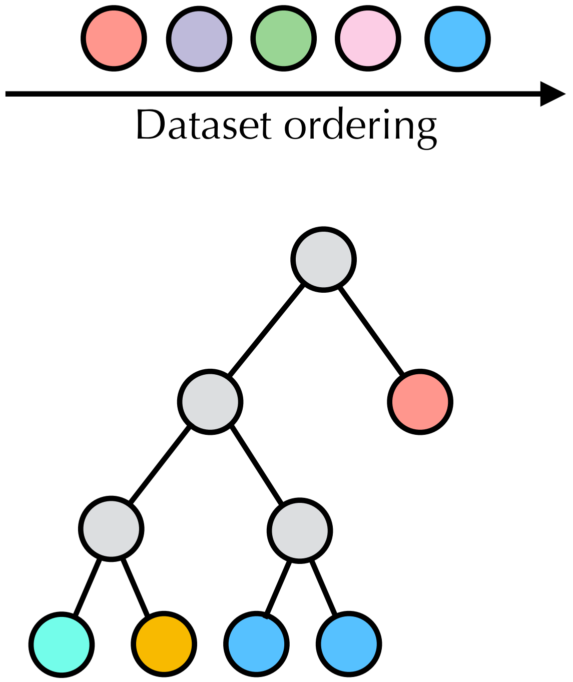


GRINCH

Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

Incremental Operation



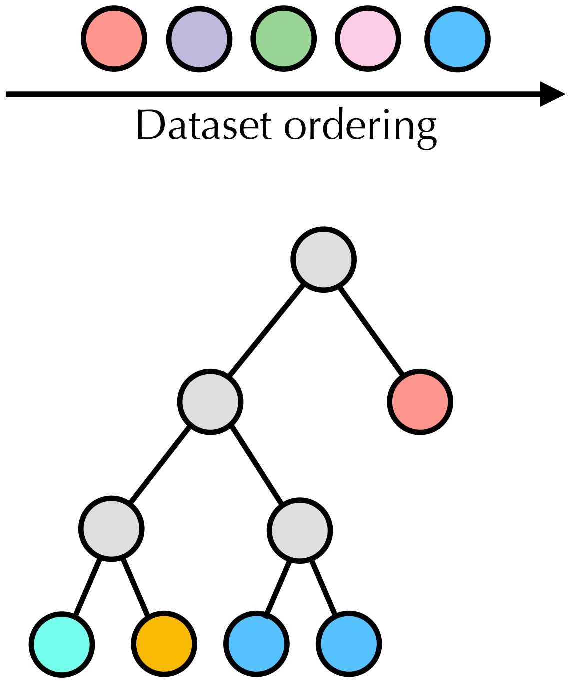
Rotations for Local Re-arrangements

GRINCH

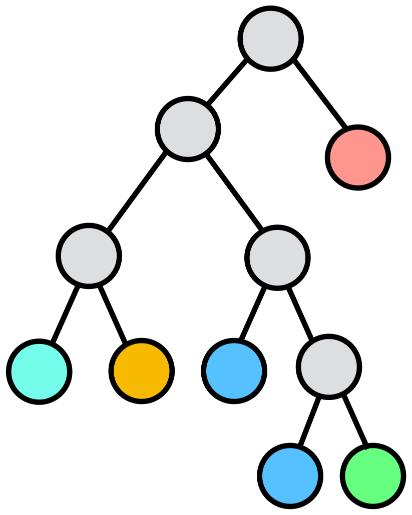
Grafting and Rotation-based INCremental Hierarchical clustering

At a high level:

Incremental Operation



Rotations for Local Re-arrangements

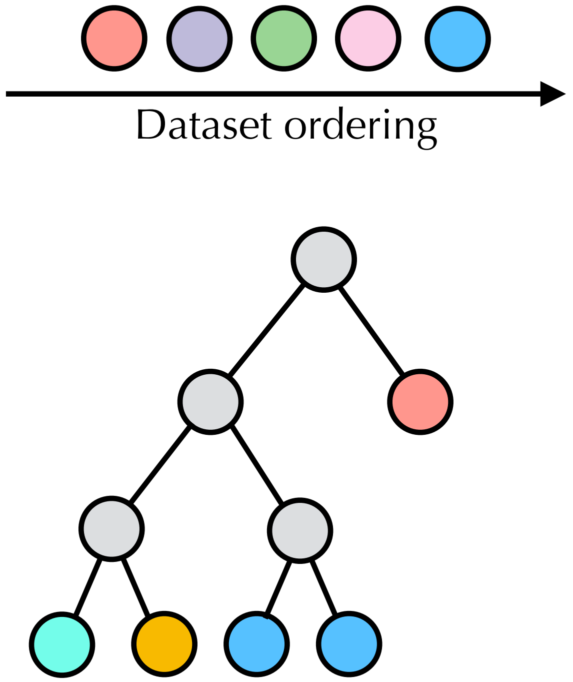


GRINCH

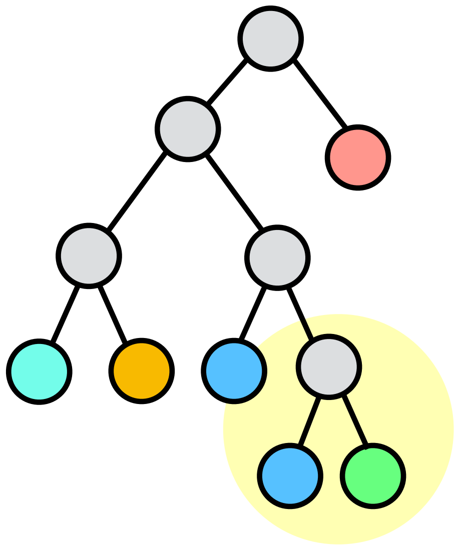
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

Incremental Operation



Rotations for Local Re-arrangements

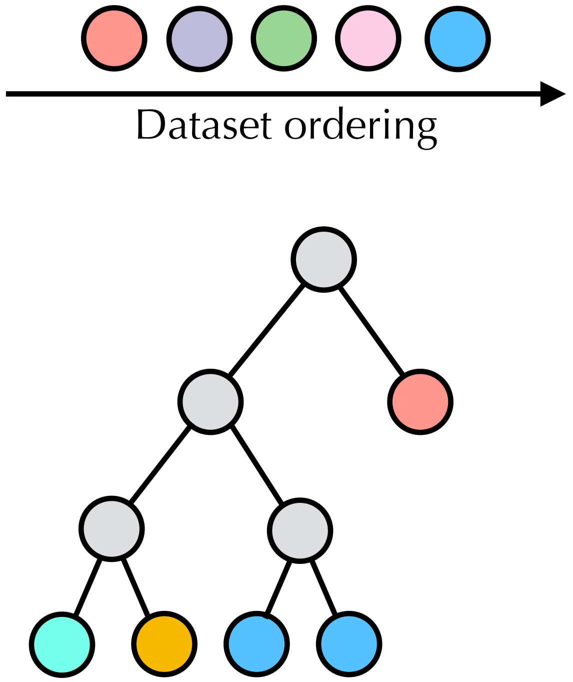


GRINCH

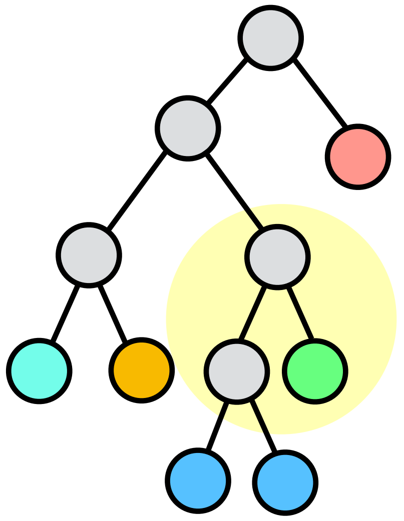
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

Incremental Operation



Rotations for Local Re-arrangements

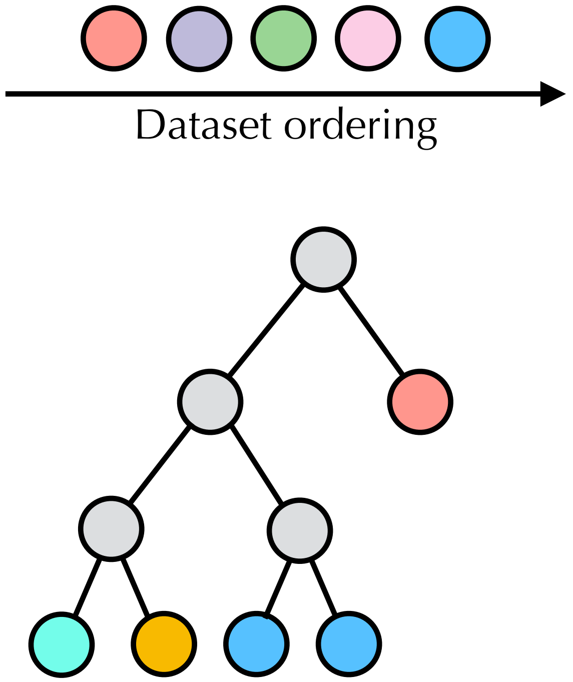


GRINCH

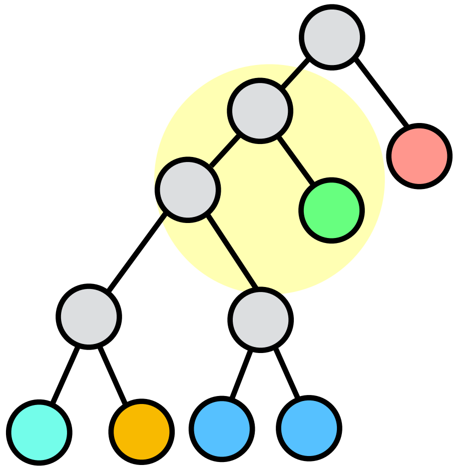
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

Incremental Operation



Rotations for Local Re-arrangements

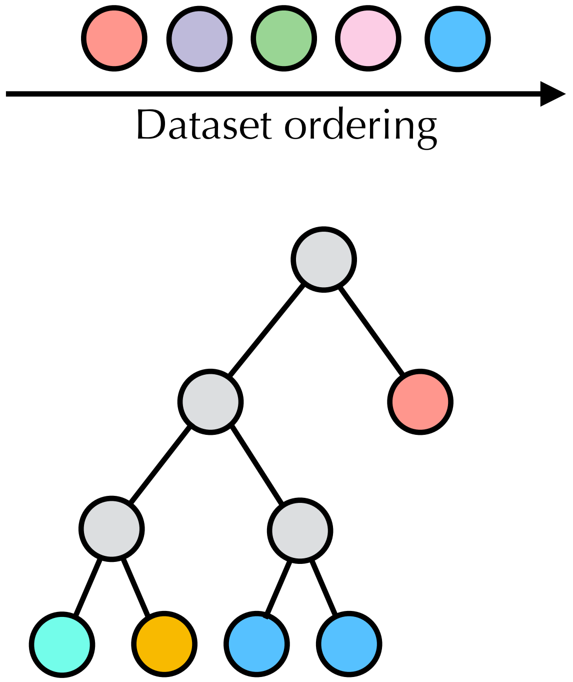


GRINCH

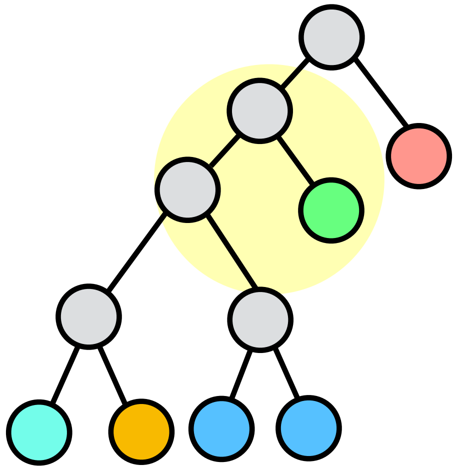
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

Incremental Operation



Rotations for Local Re-arrangements



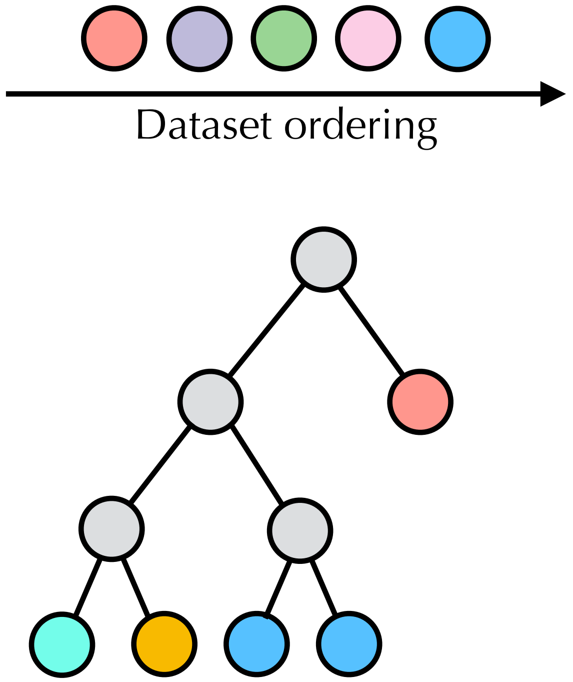
Grafts for Global Re-arrangements

GRINCH

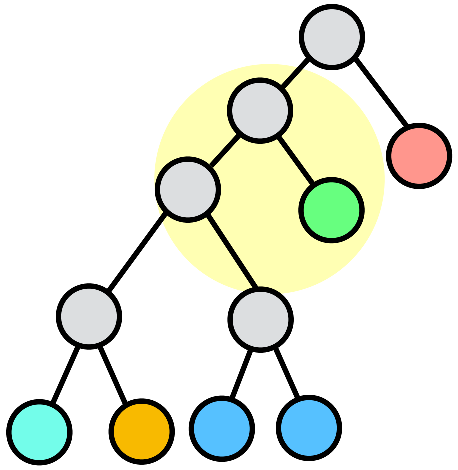
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

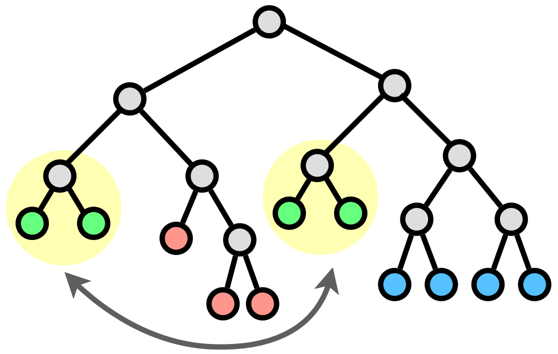
Incremental Operation



Rotations for Local Re-arrangements



Grafts for Global Re-arrangements



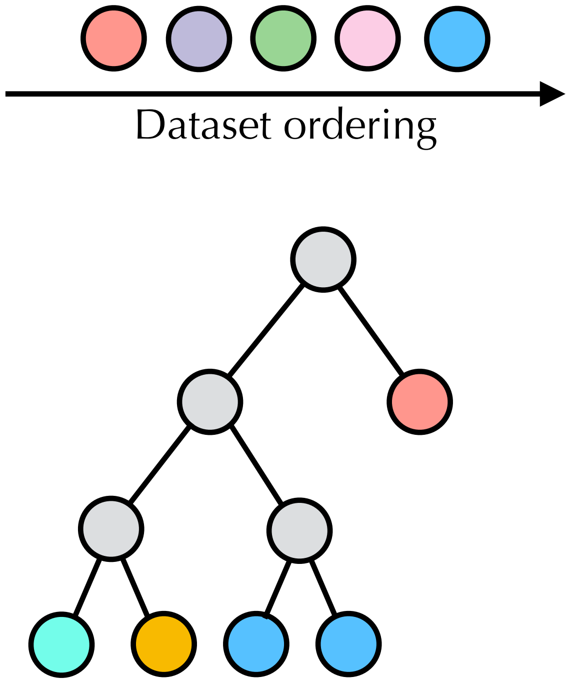
"…GRINCH may steal a subtree…"

GRINCH

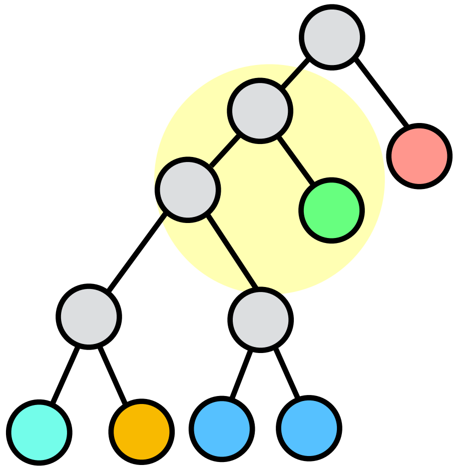
Grafting and **R**otation-based **INC**remental **H**ierarchical clustering

At a high level:

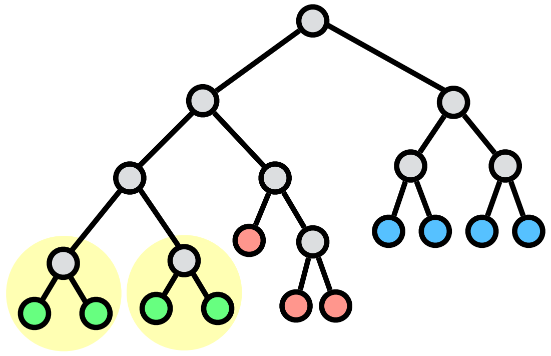
Incremental Operation



Rotations for Local Re-arrangements

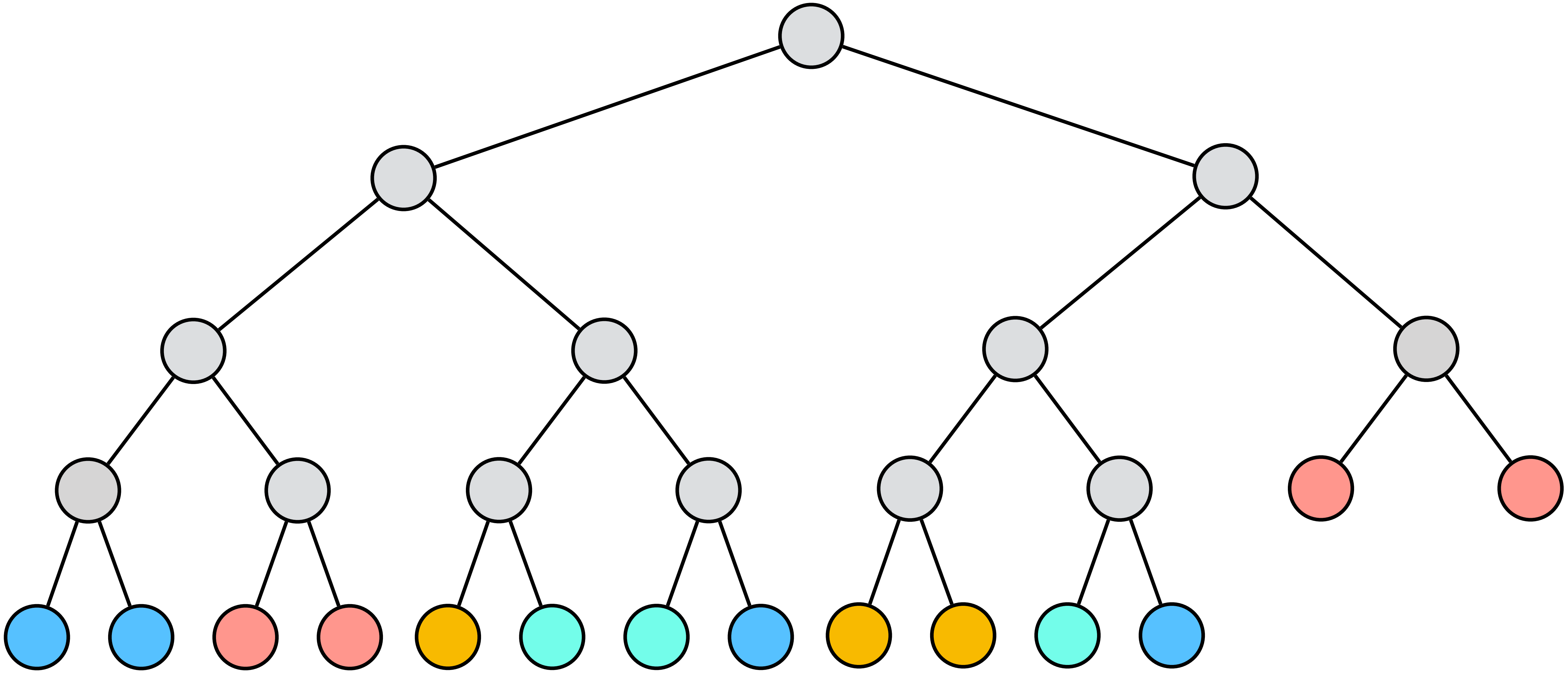


Grafts for Global Re-arrangements

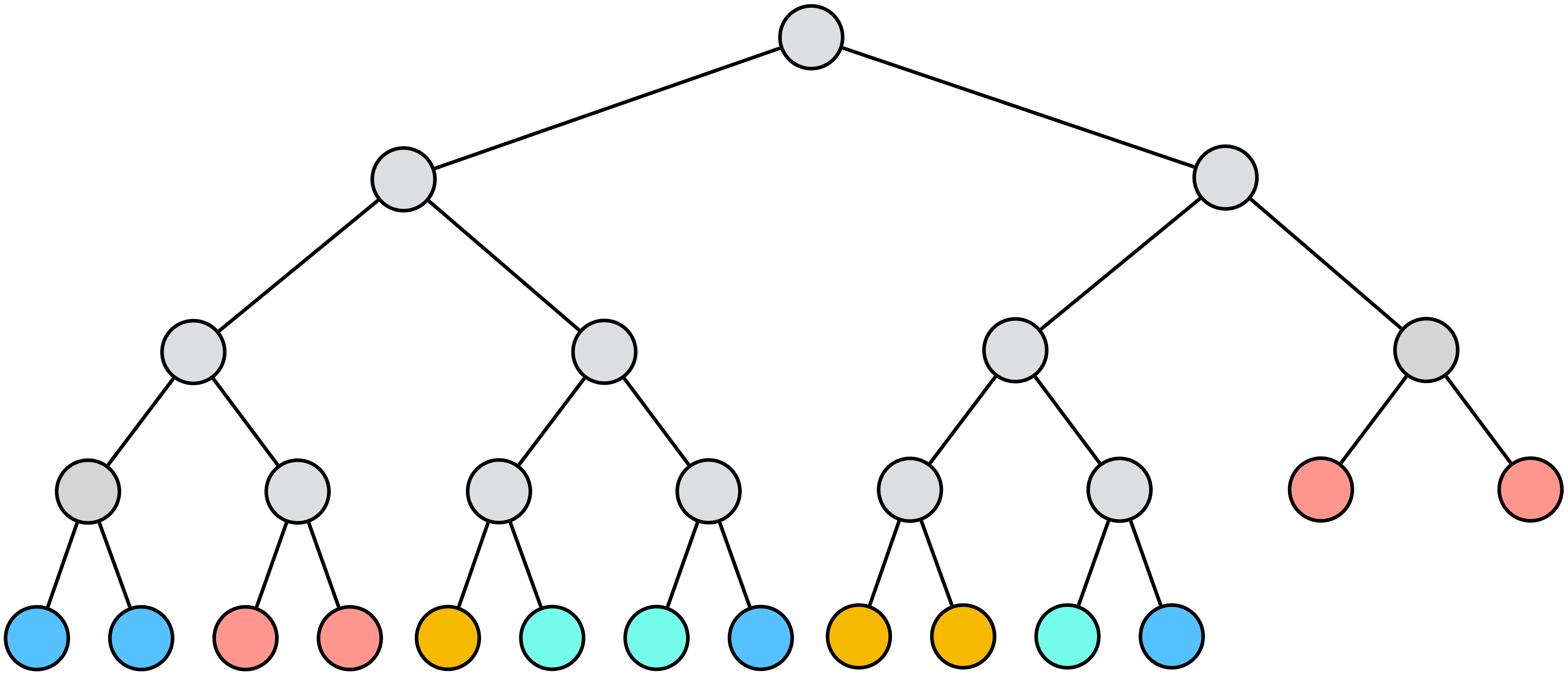


Use with **any linkage function**

GRINCH

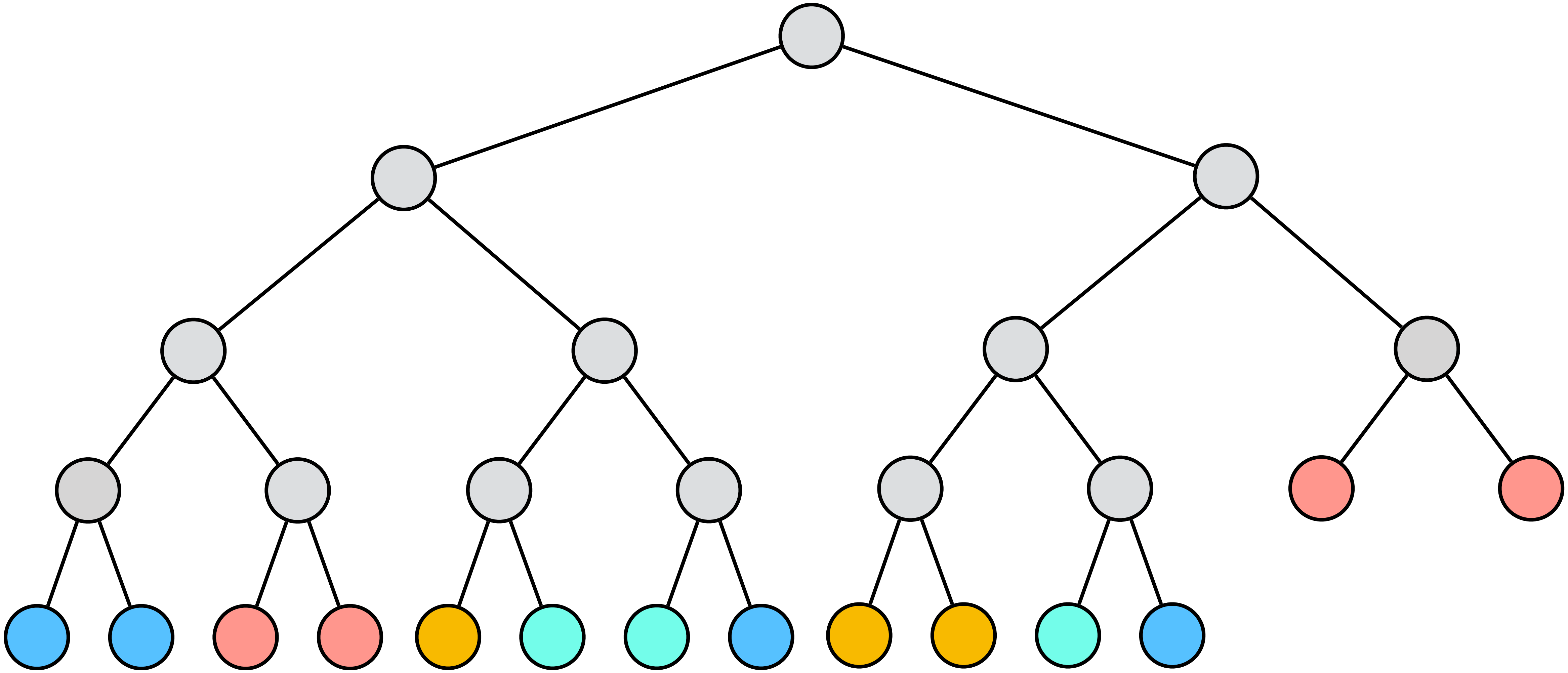


GRINCH



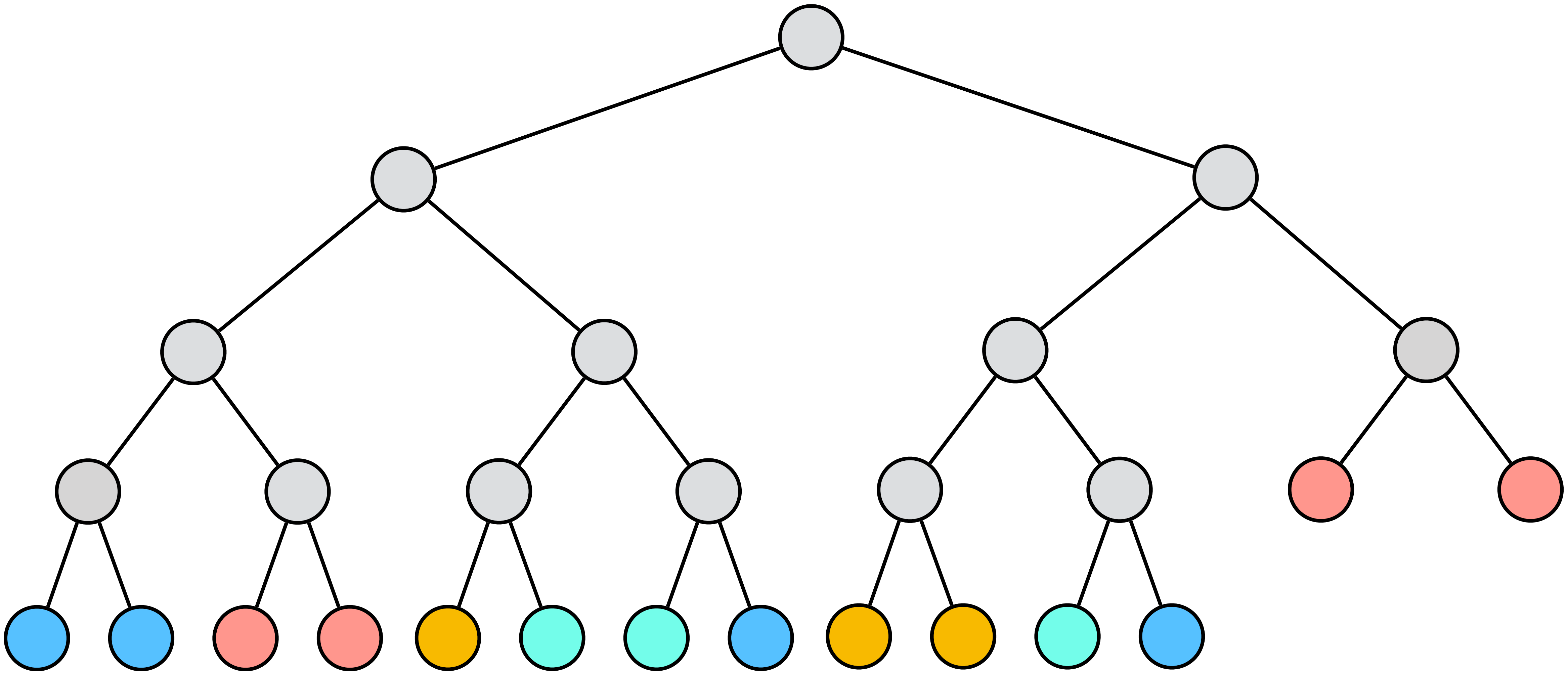
GRINCH

data points stored at the leaves of the tree,



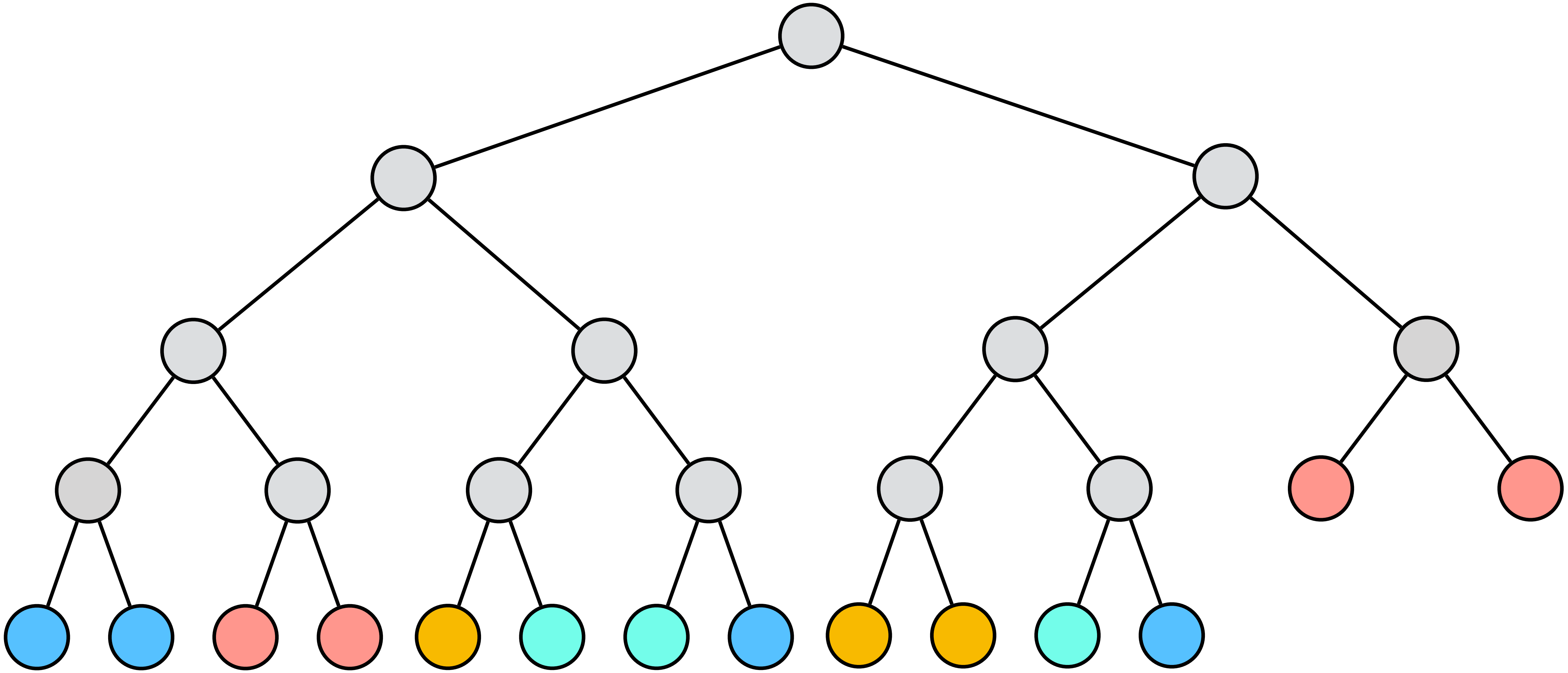
GRINCH

data points stored at the leaves of the tree, color indicates ground-truth cluster



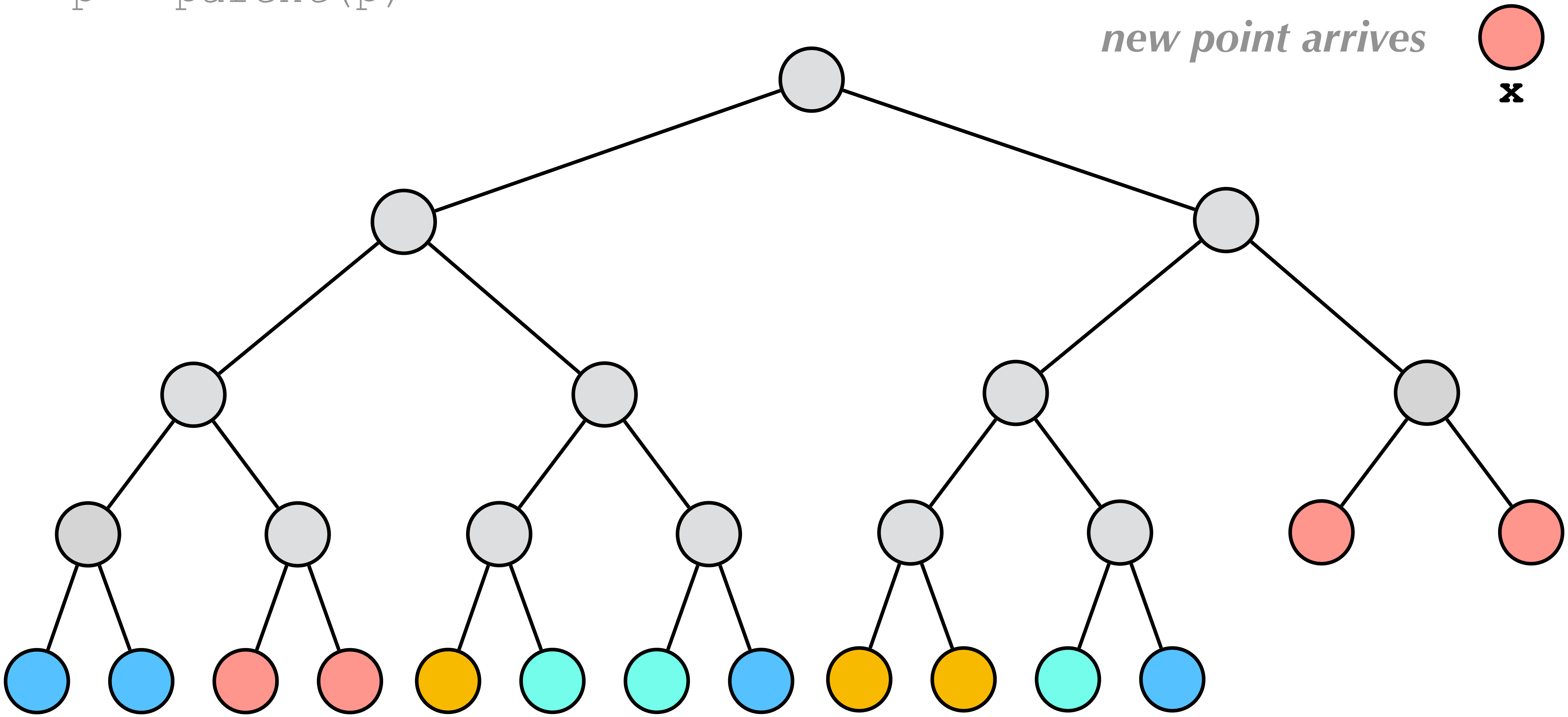
GRINCH

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



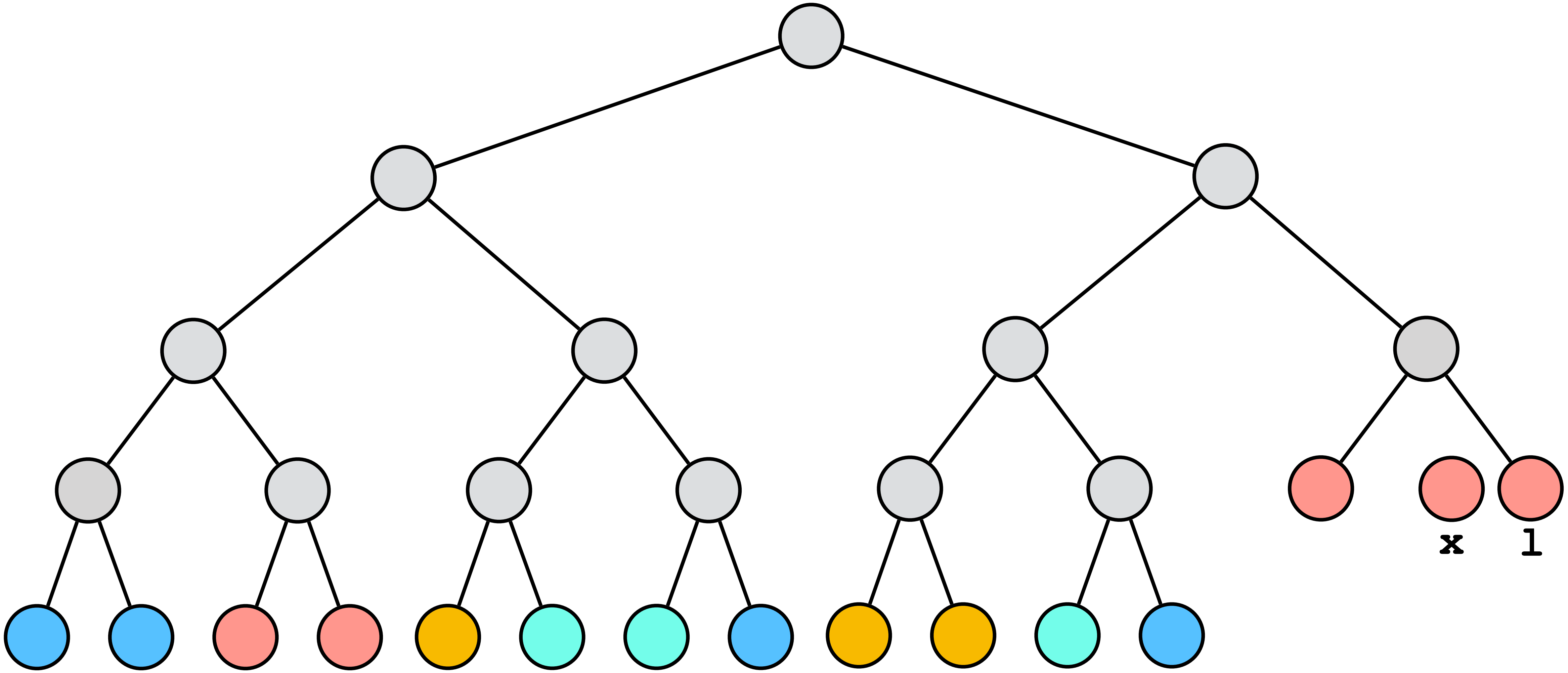
GRINCH

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



GRINCH

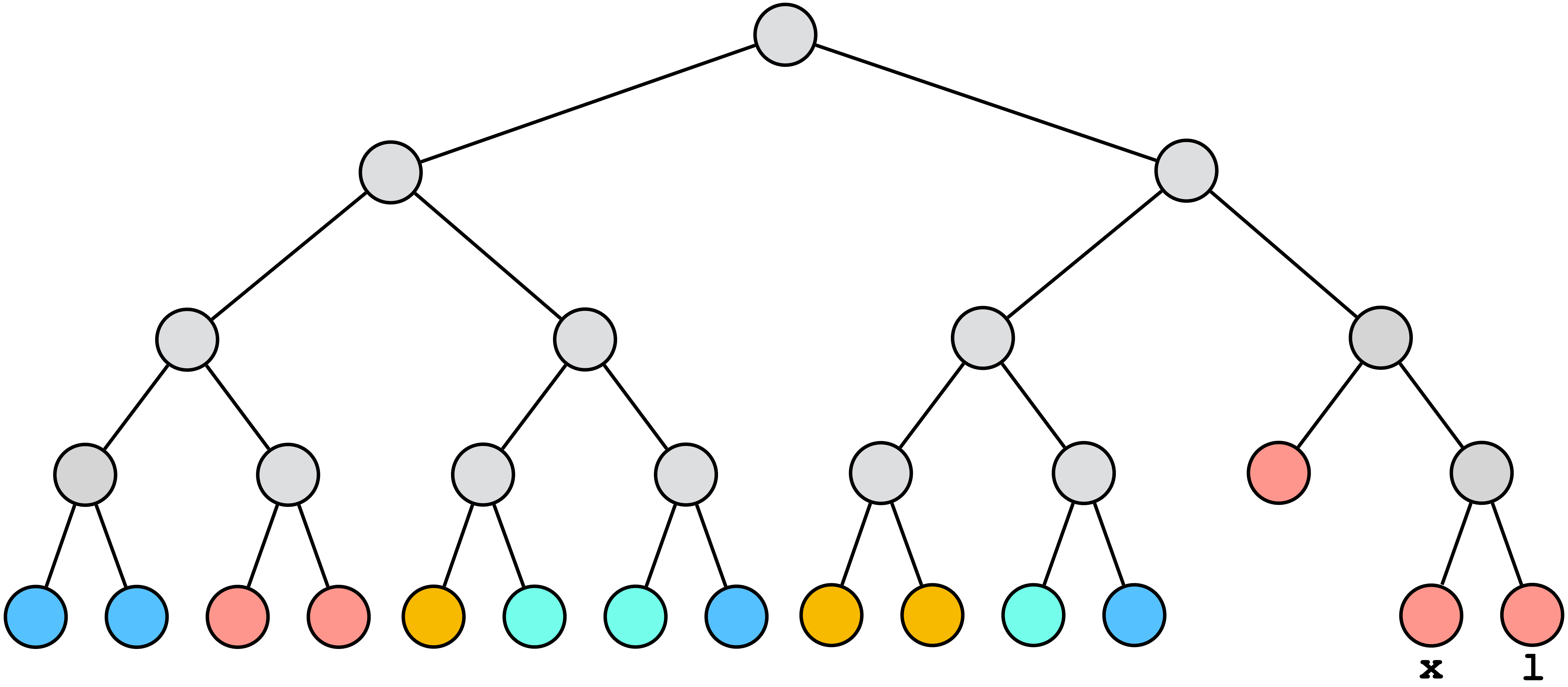
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



find nearest neighbor

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

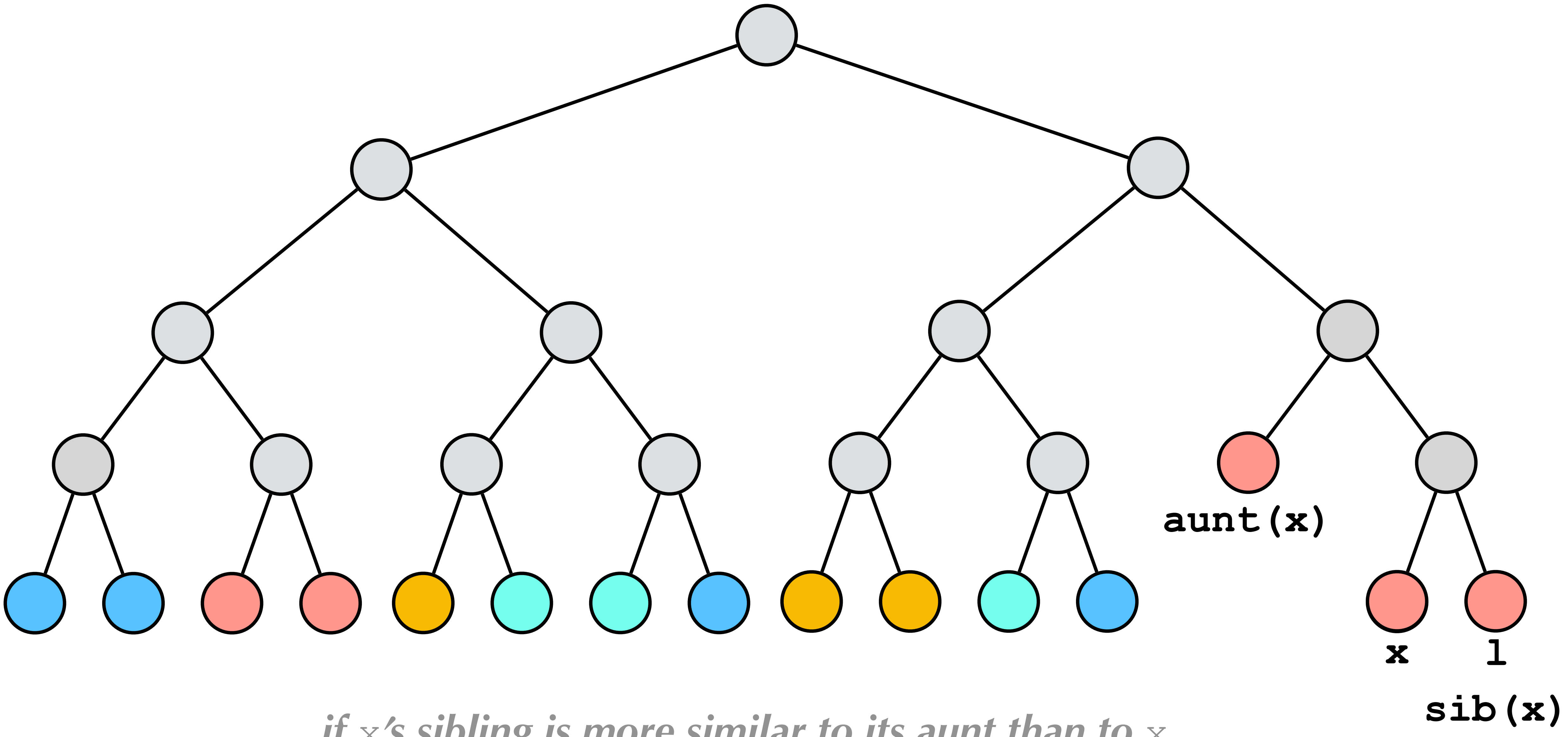
GRINCH



make them siblings

GRINCH

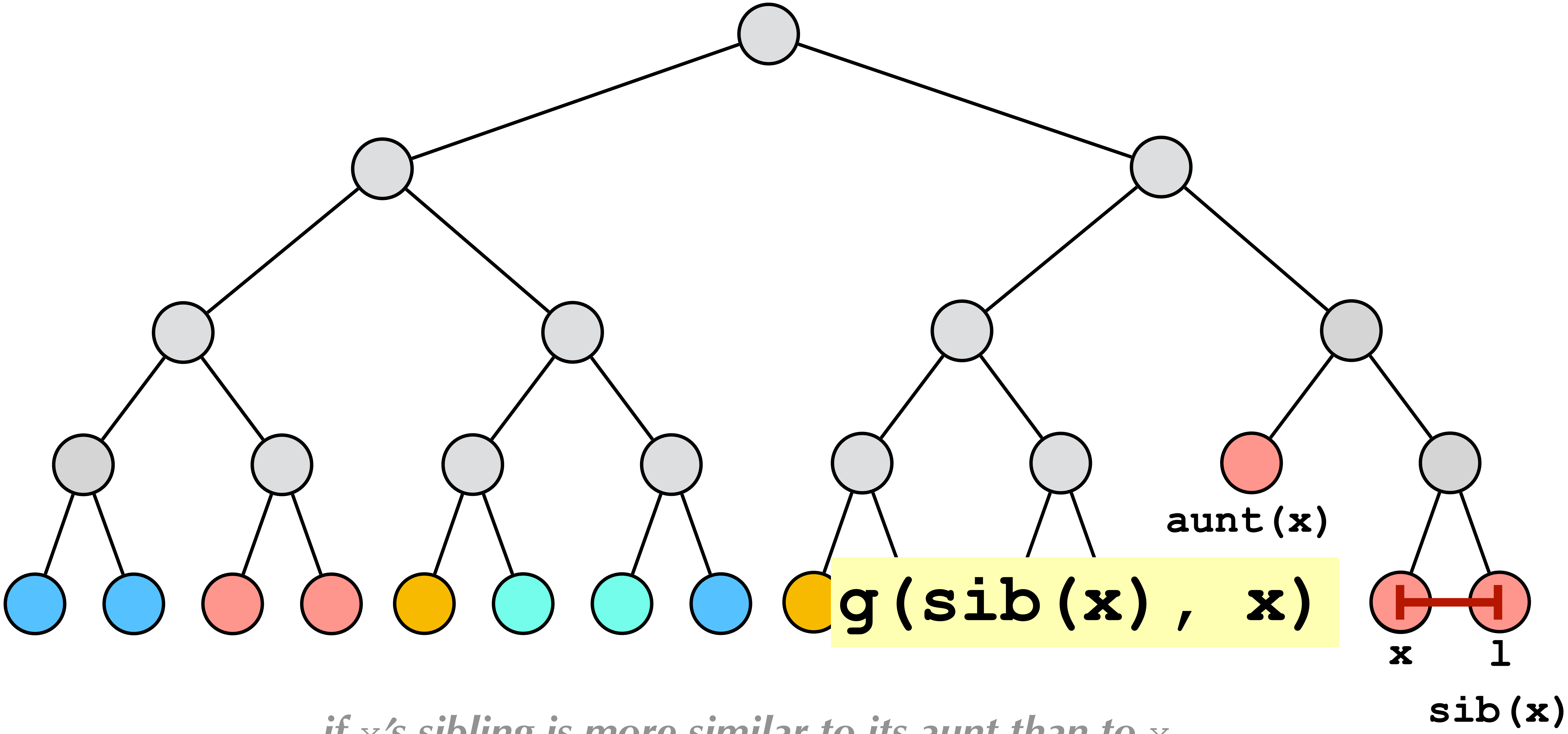
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



if x's sibling is more similar to its aunt than to x…

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

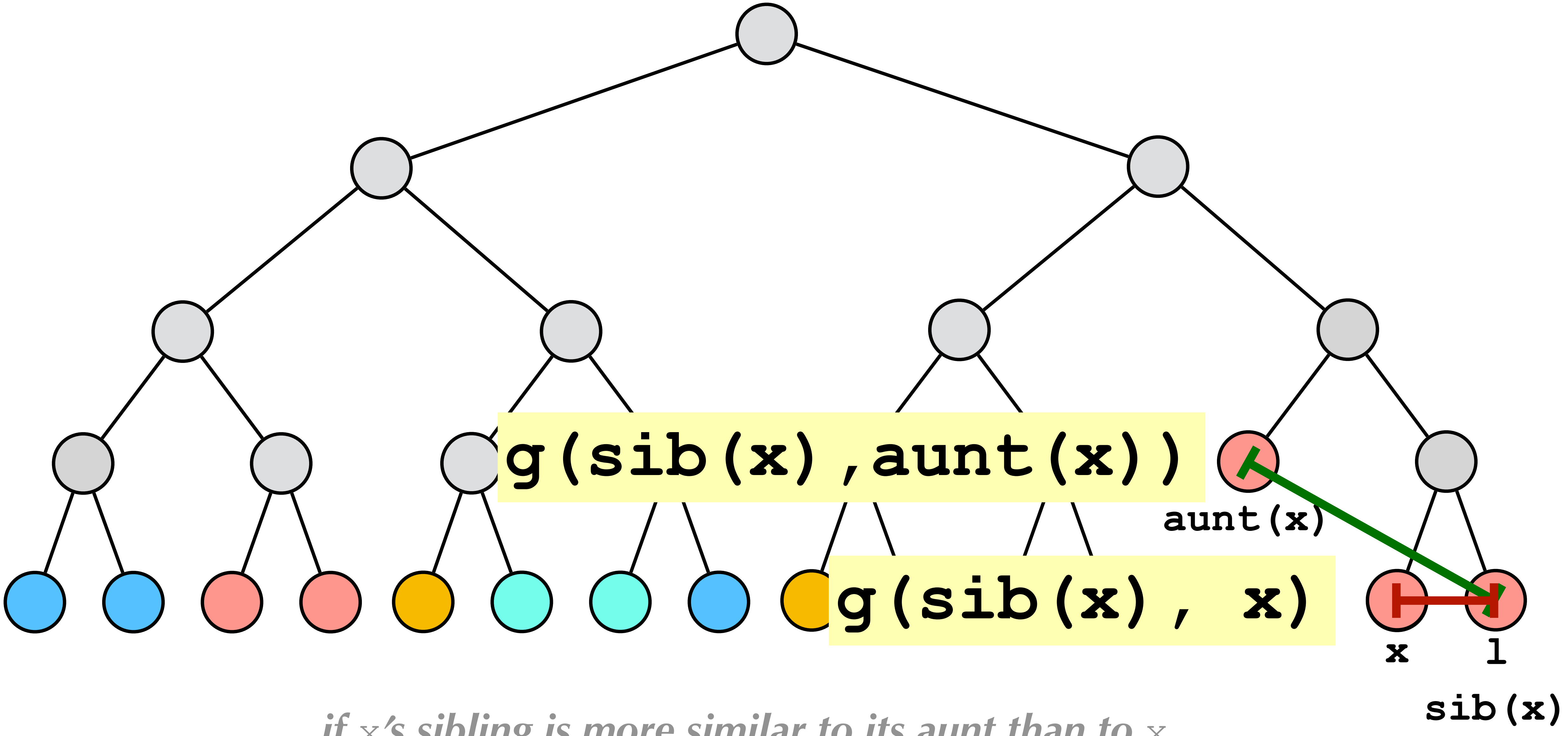
GRINCH



if x's sibling is more similar to its aunt than to x…

GRINCH

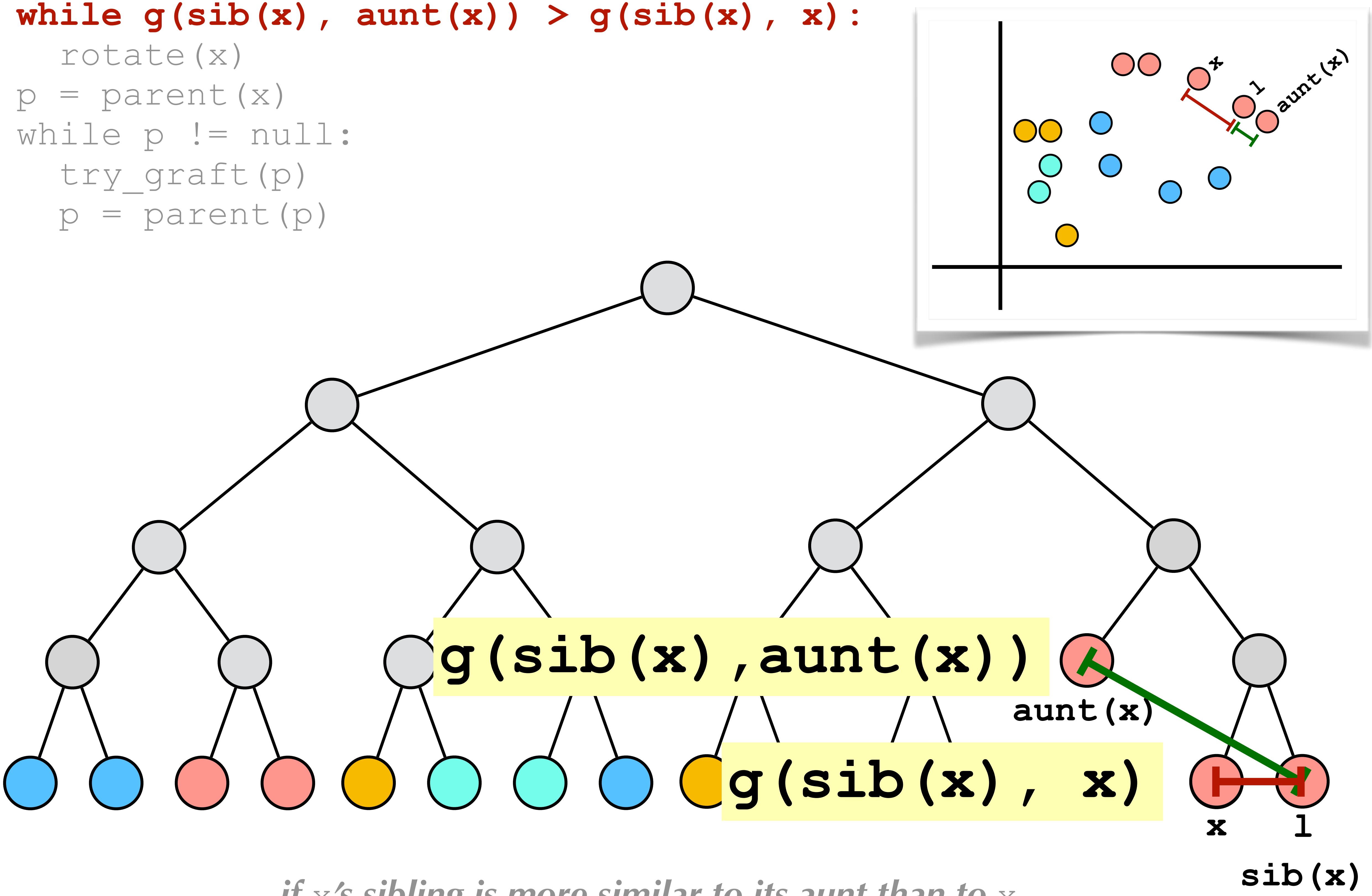
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



if x's sibling is more similar to its aunt than to x…

GRINCH

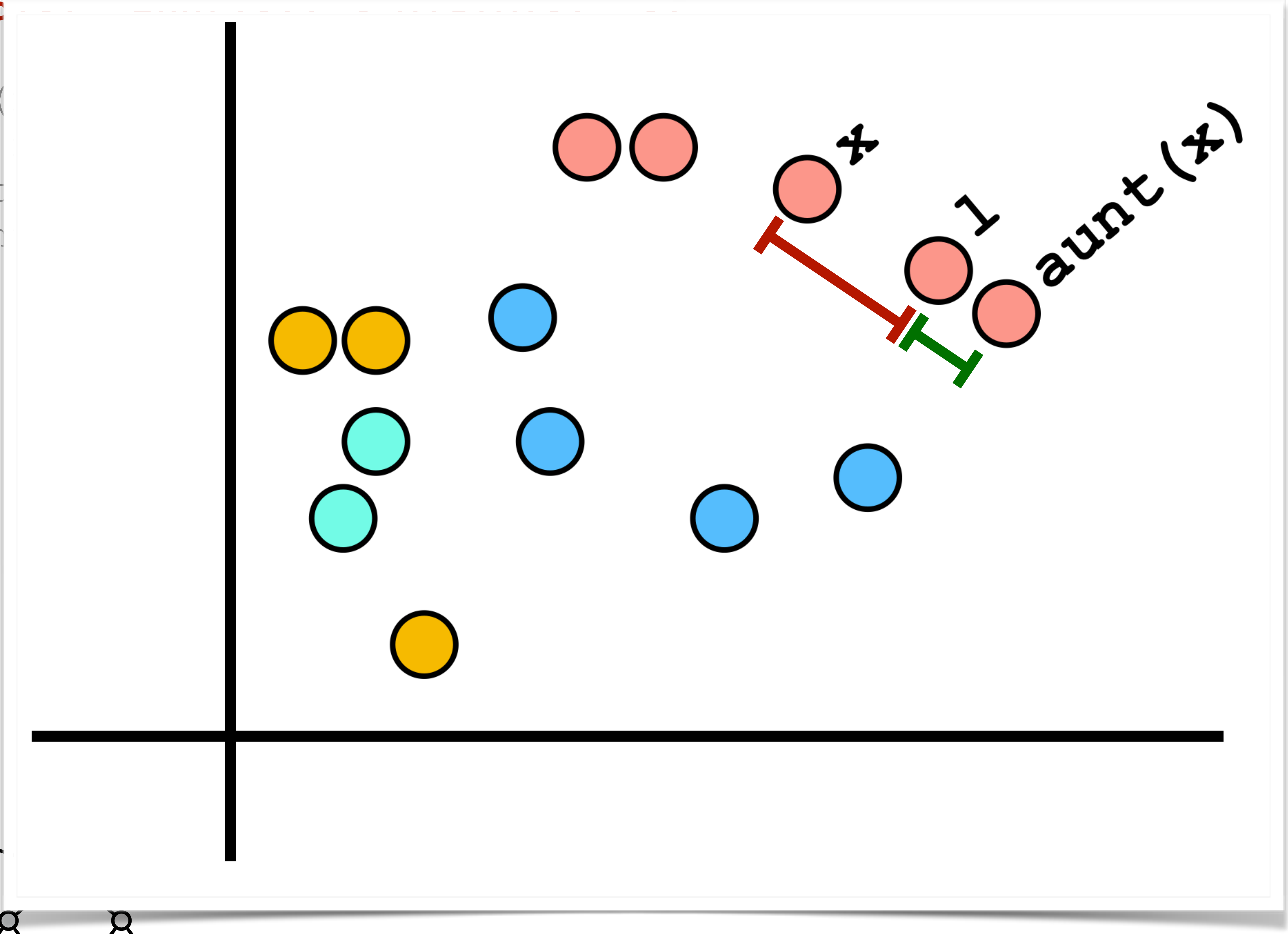
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

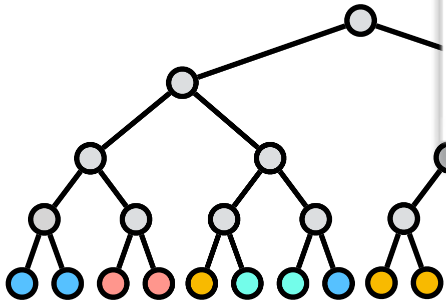


if x*'s sibling is more similar to its aunt than to* x*…*

GRINCH

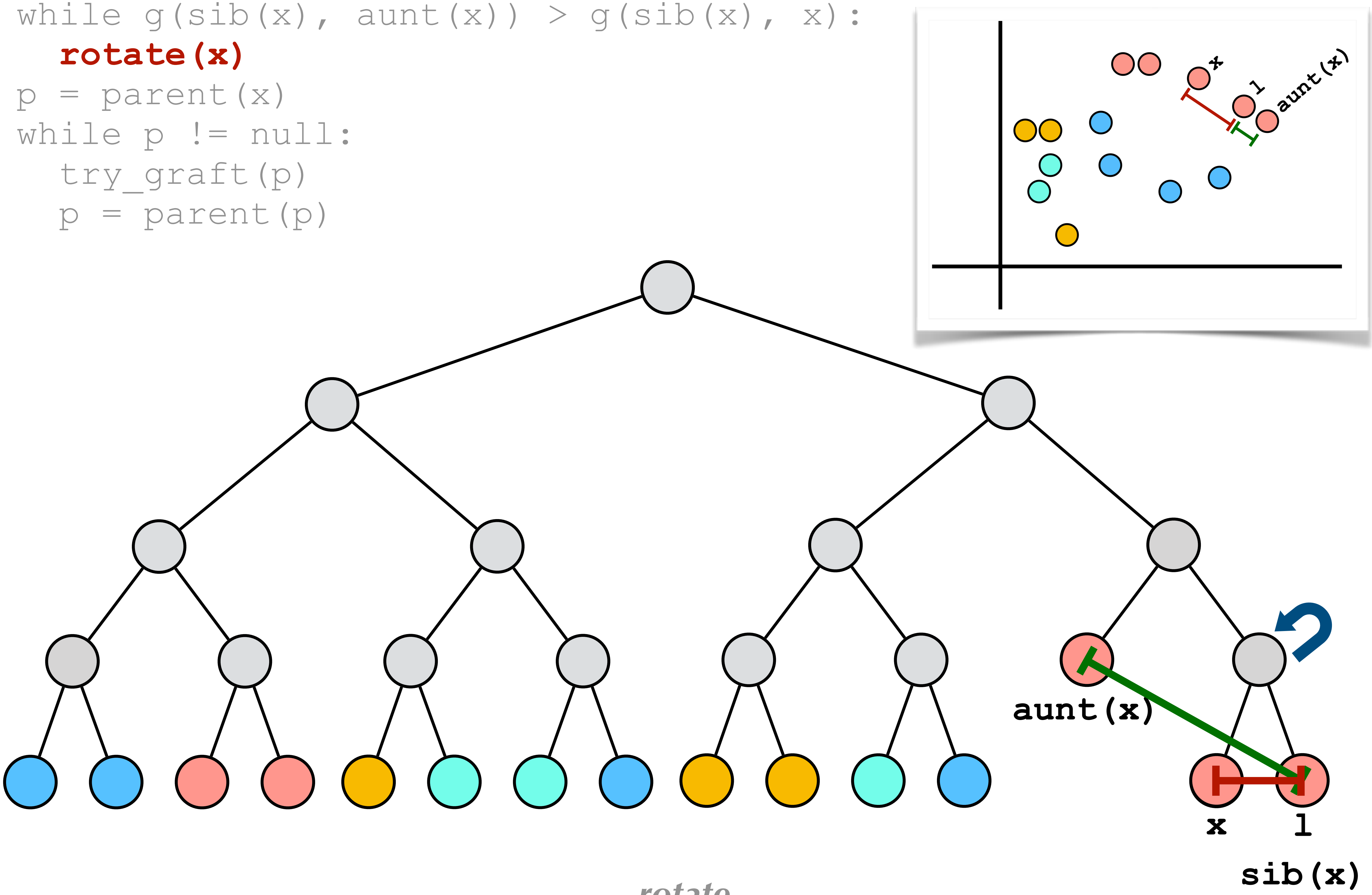
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib
    rotate(x)
  p = parent(
  while p !=
    try_graft
    p = paren
```





GRINCH

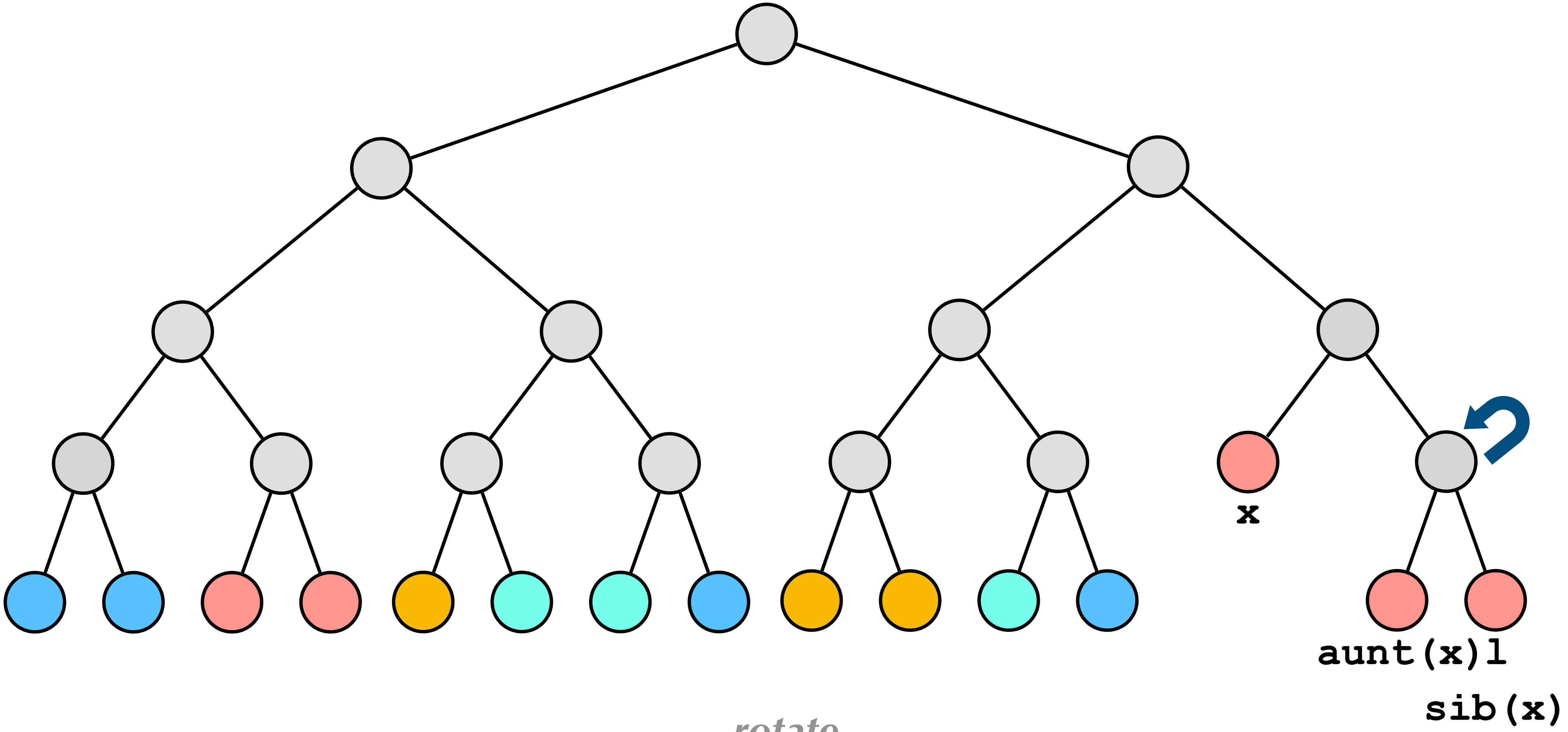
```
def insert(x, g):
    l = nearest_neighbor(x)
    p = make_sib(l, x)
    while g(sib(x), aunt(x)) > g(sib(x), x):
        rotate(x)
    p = parent(x)
    while p != null:
        try_graft(p)
        p = parent(p)
```



...rotate

GRINCH

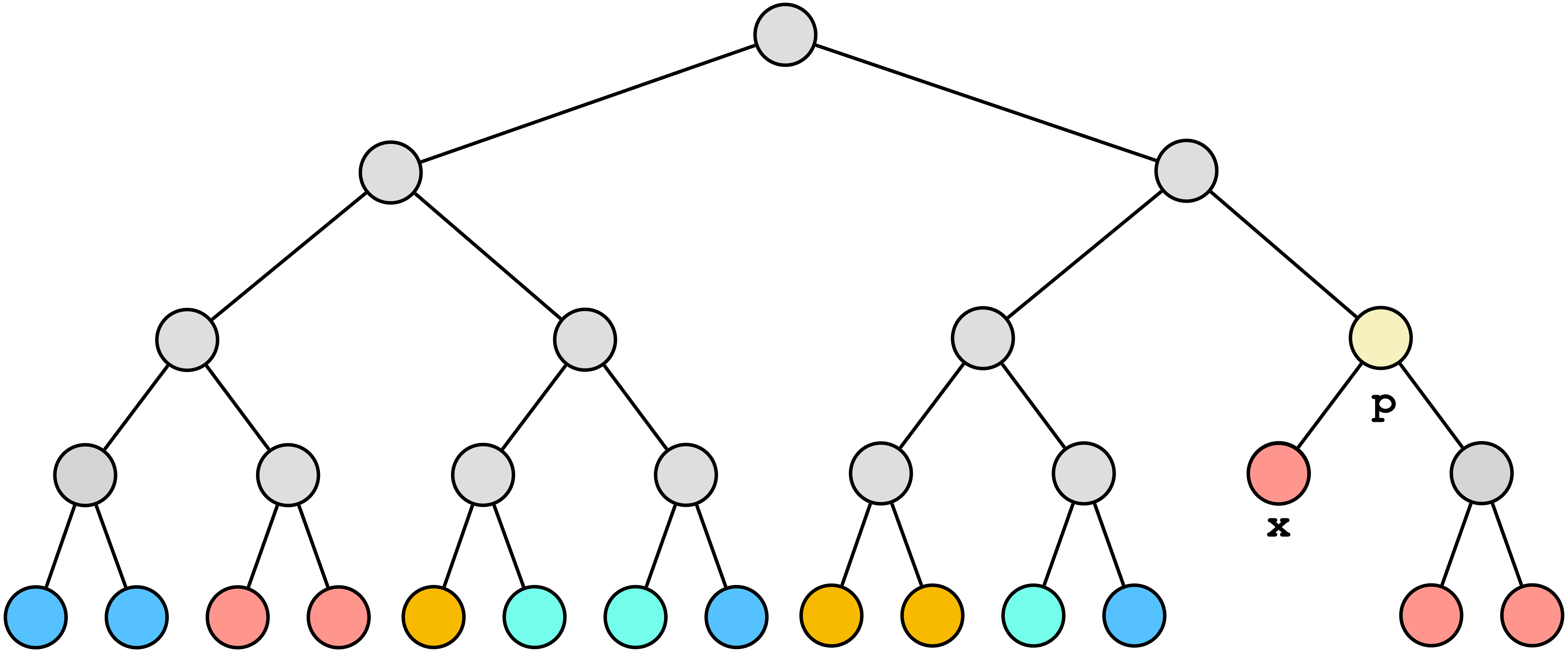
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



…rotate

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

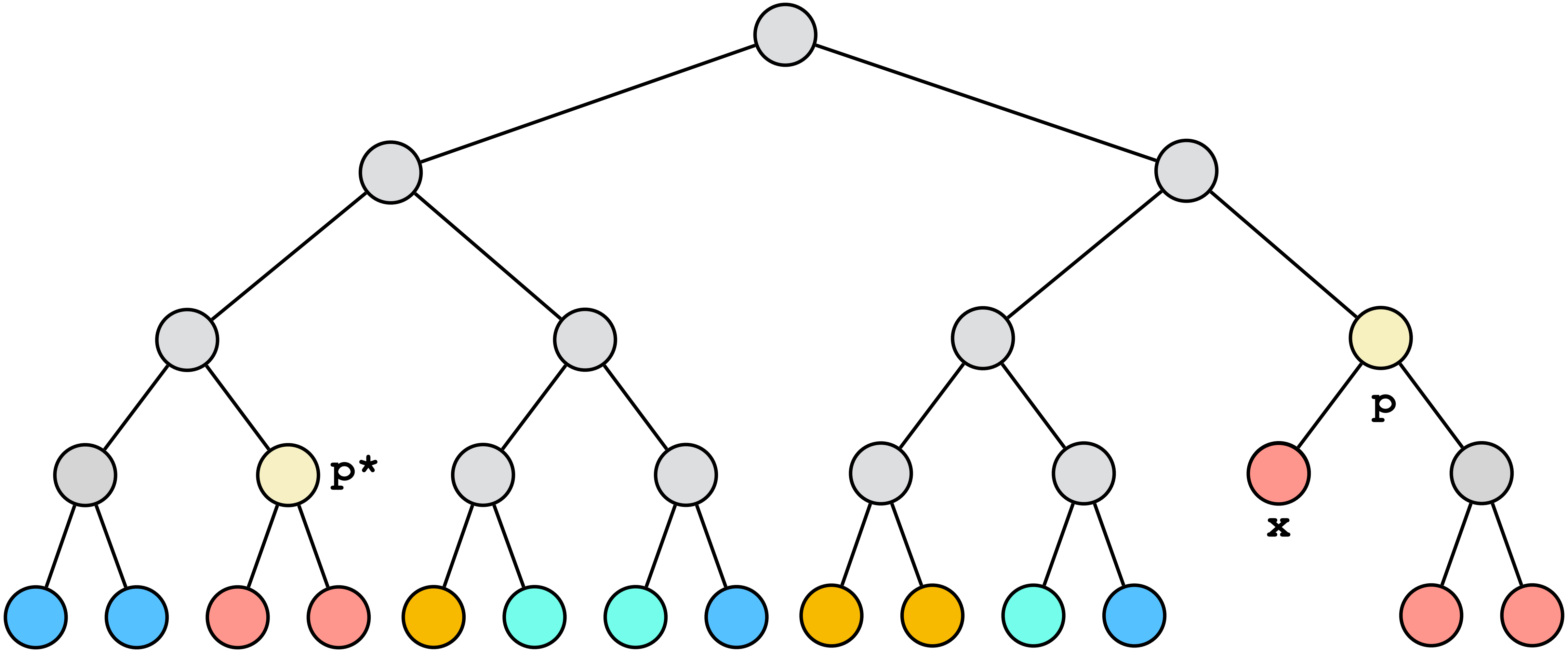
GRINCH



now consider, p*, the parent of* x*, and attempt a graft*

GRINCH

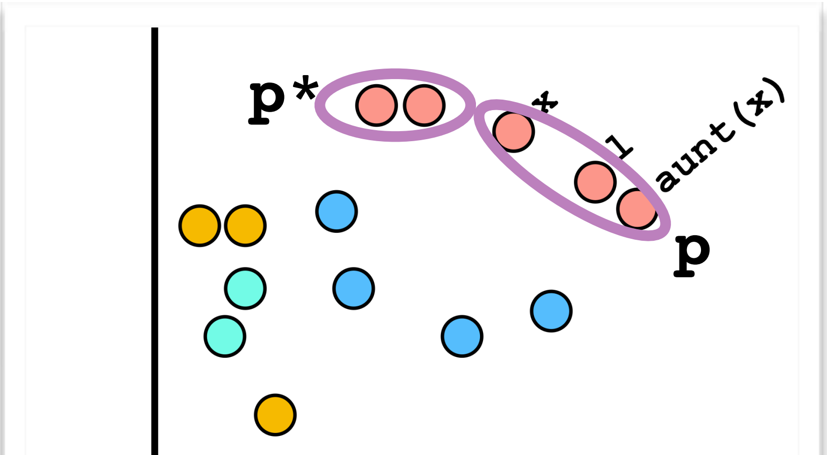
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

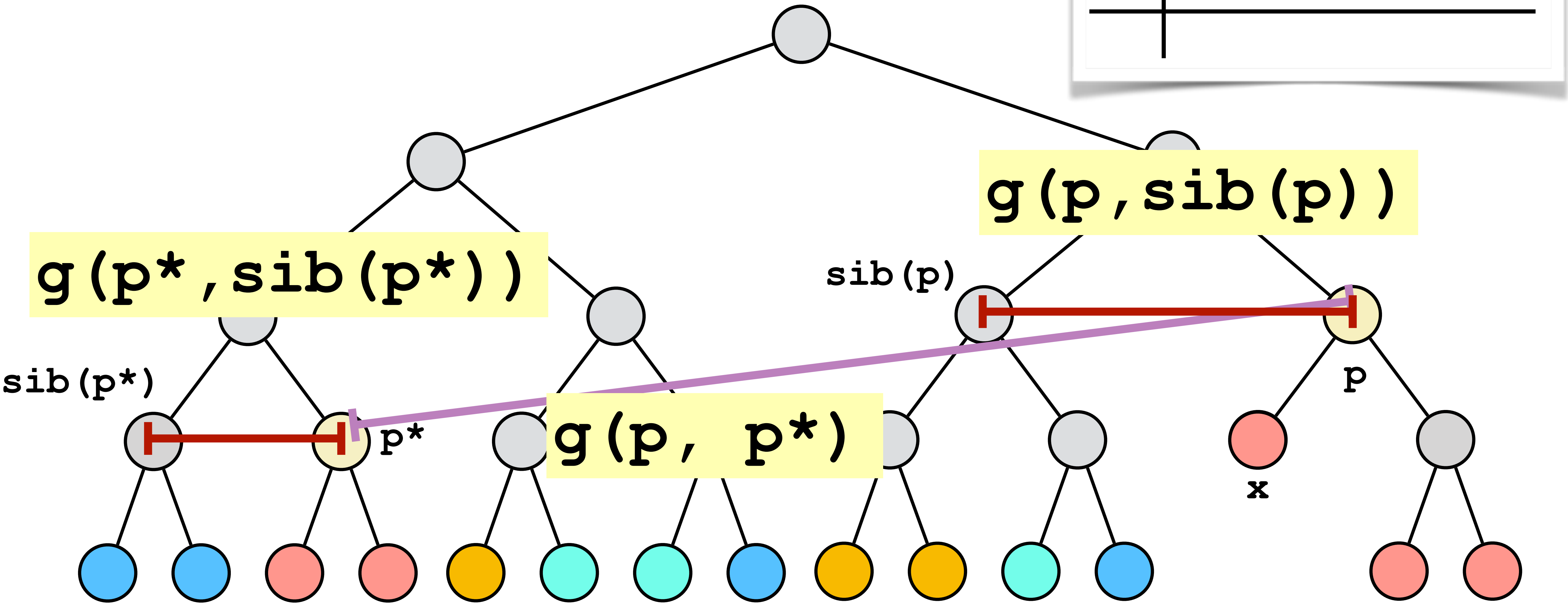


to do so, find its nearest neighbor…

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

GRINCH

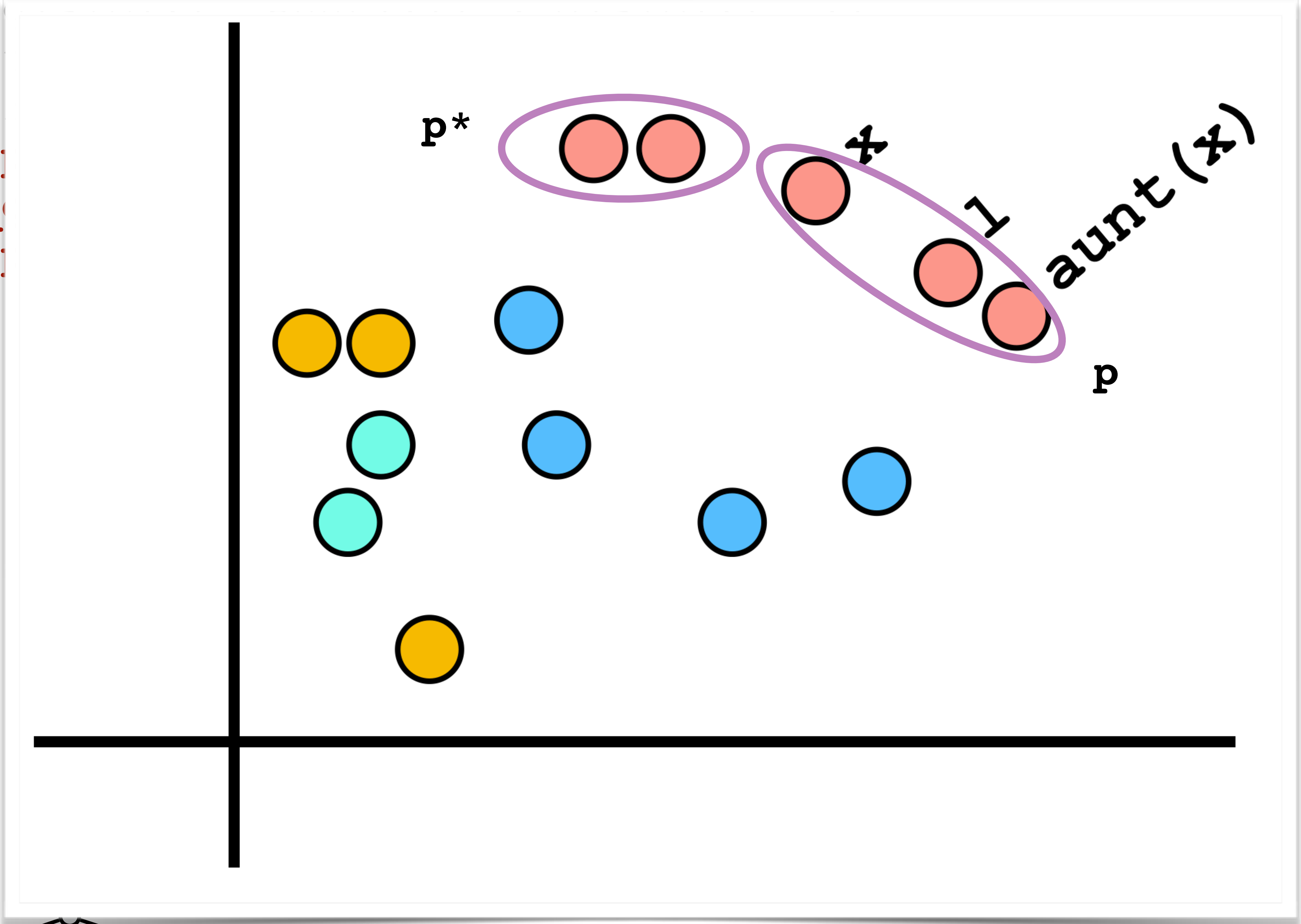


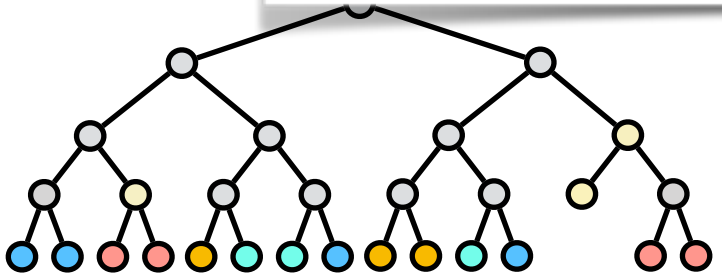


…and test: g(p, p*) > max[g(p, sib(p)), g(p*, sib(p*)]

GRINCH

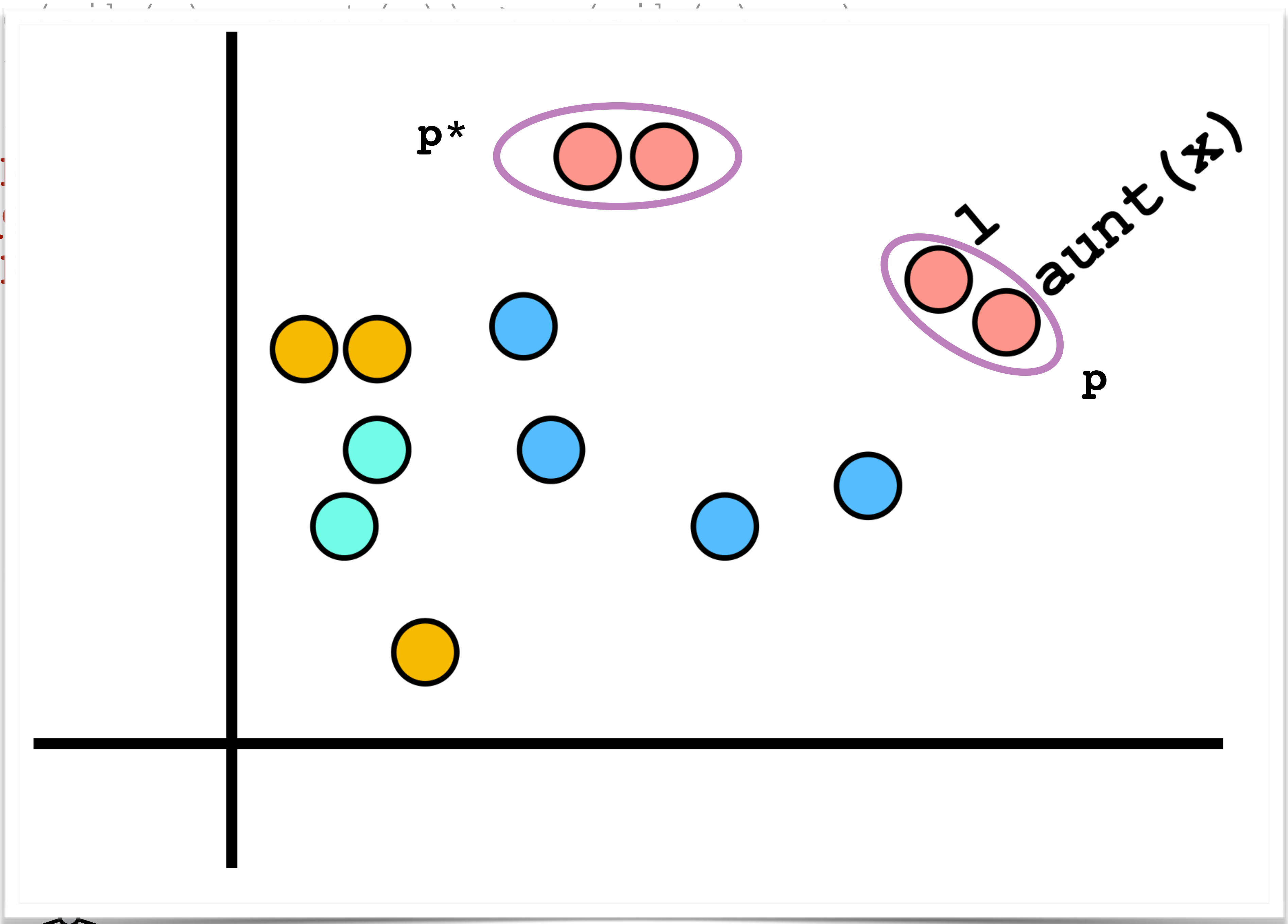
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while
    rota
  p = pa
  while
    try_
    p =
```

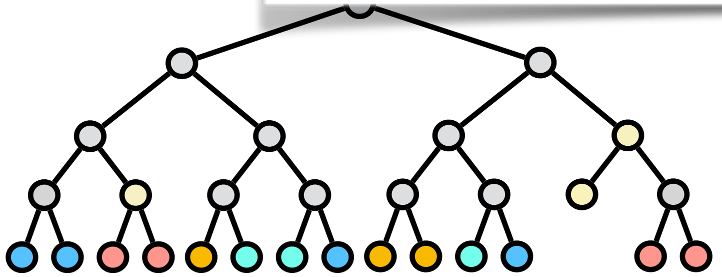




GRINCH

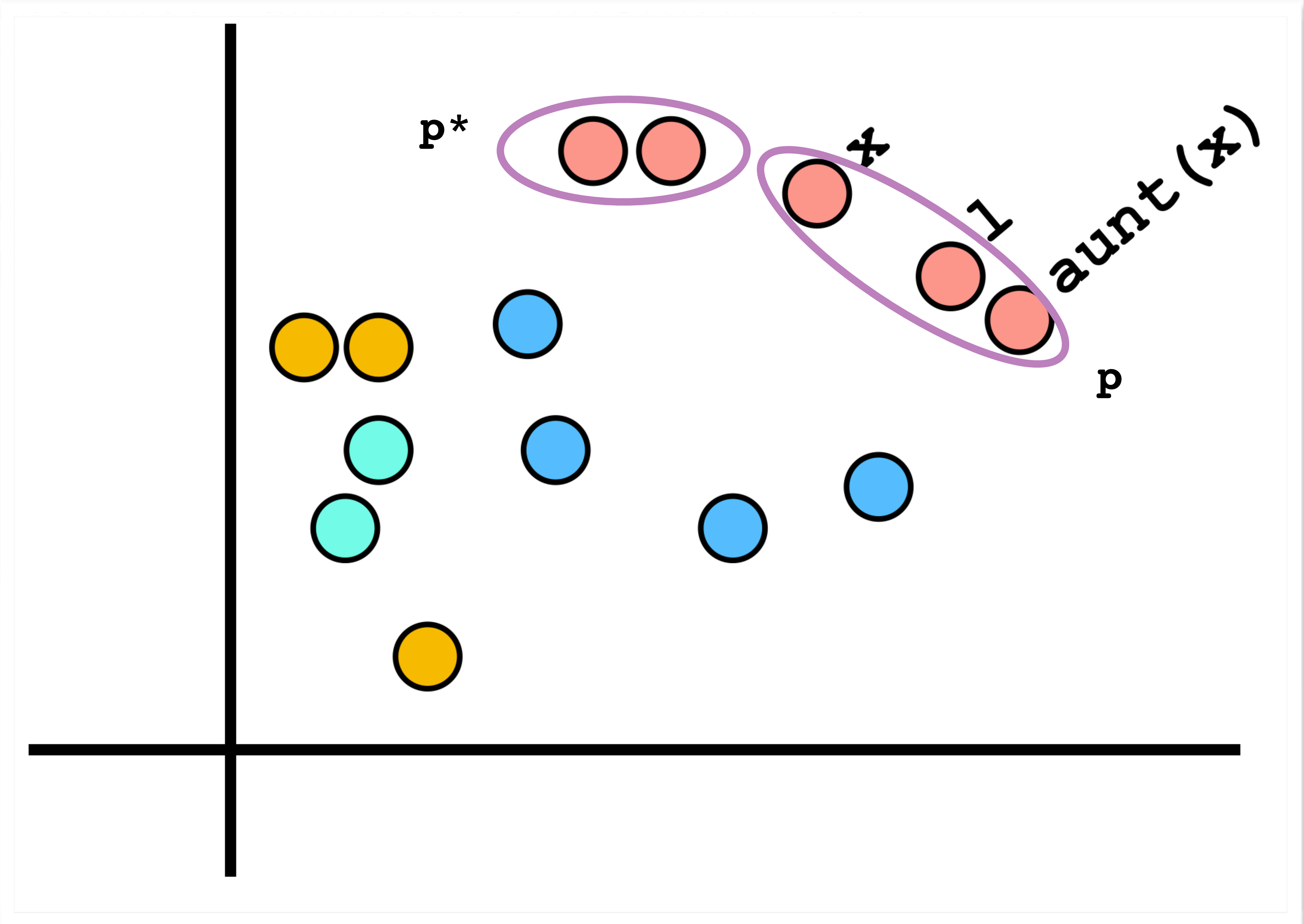
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while
    rota
  p = pa
  while
    try_
    p =
```

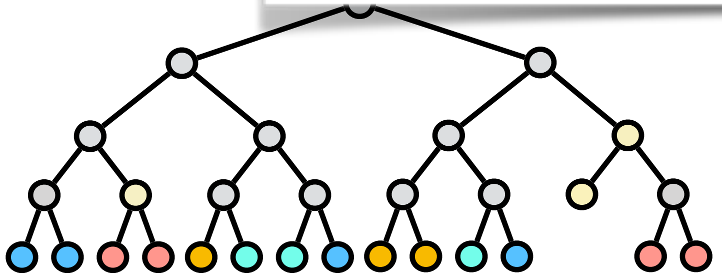




GRINCH

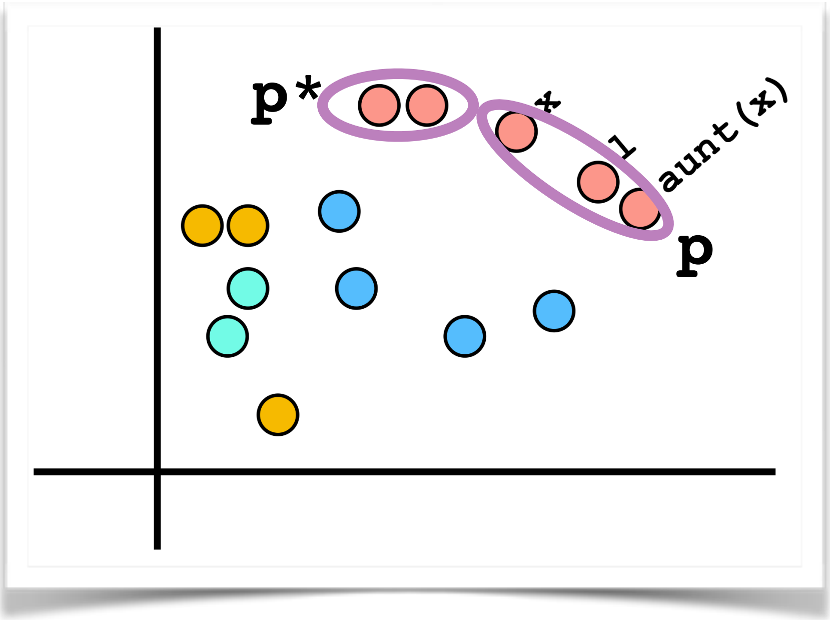
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while
    rota
  p = pa
  while
    try_
    p =
```

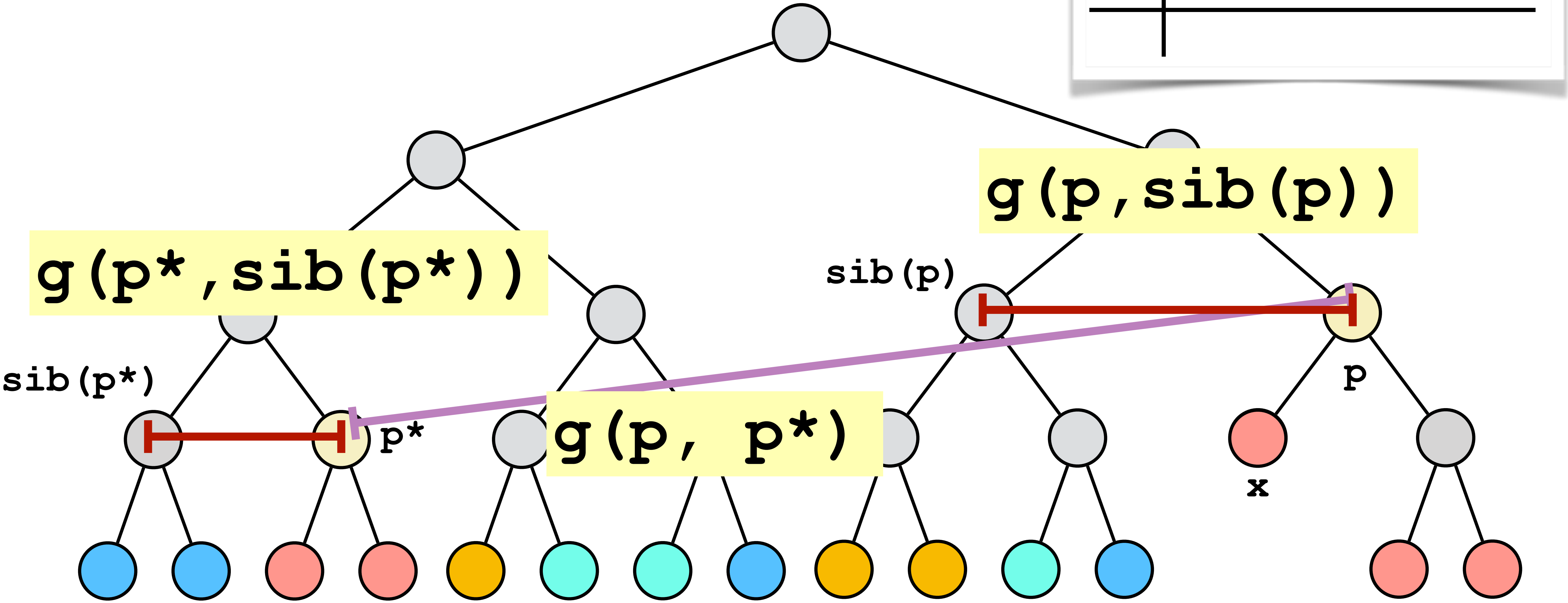




GRINCH

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

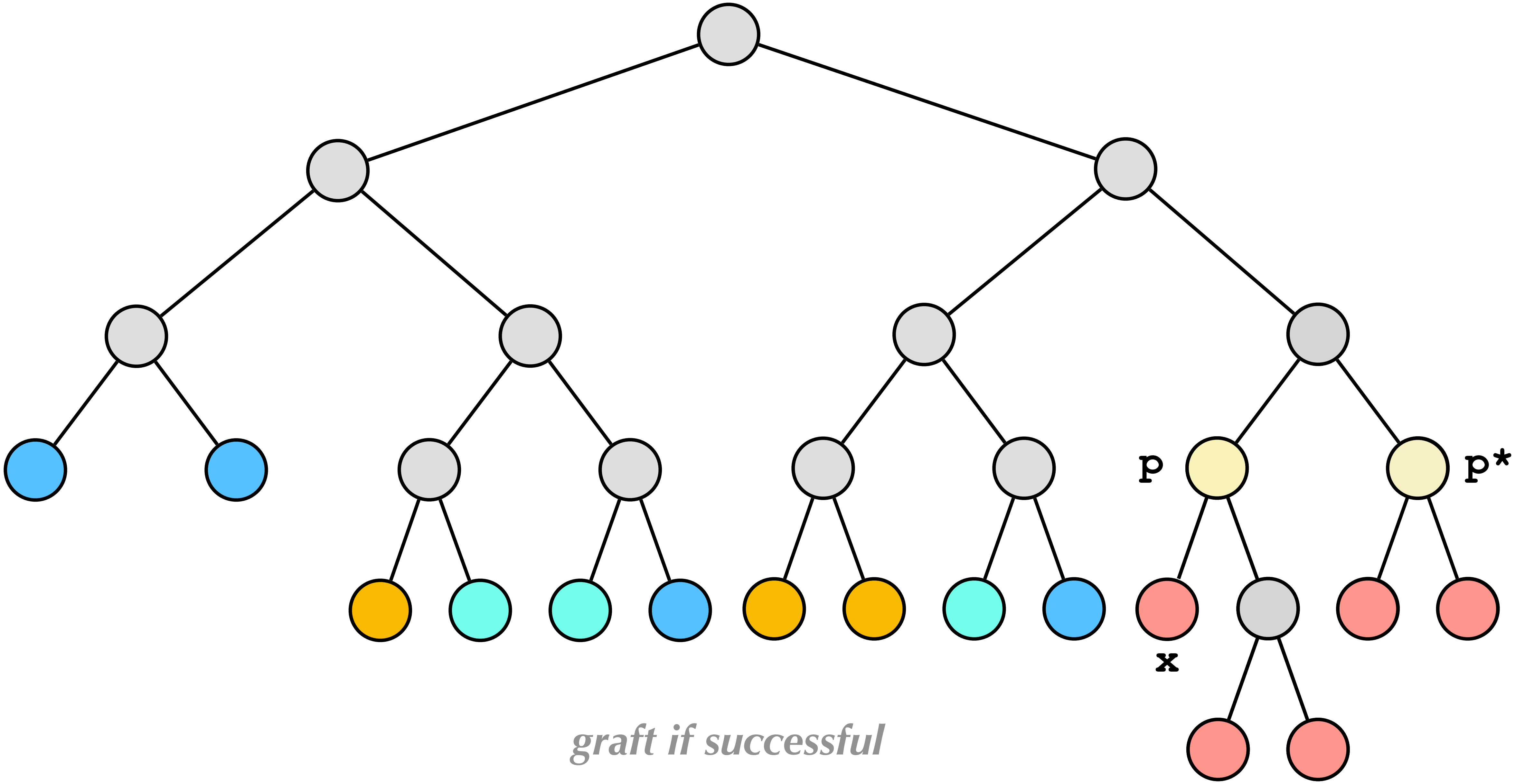




…and test: g(p, p*) > max[g(p, sib(p)), g(p*, sib(p*)]

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```

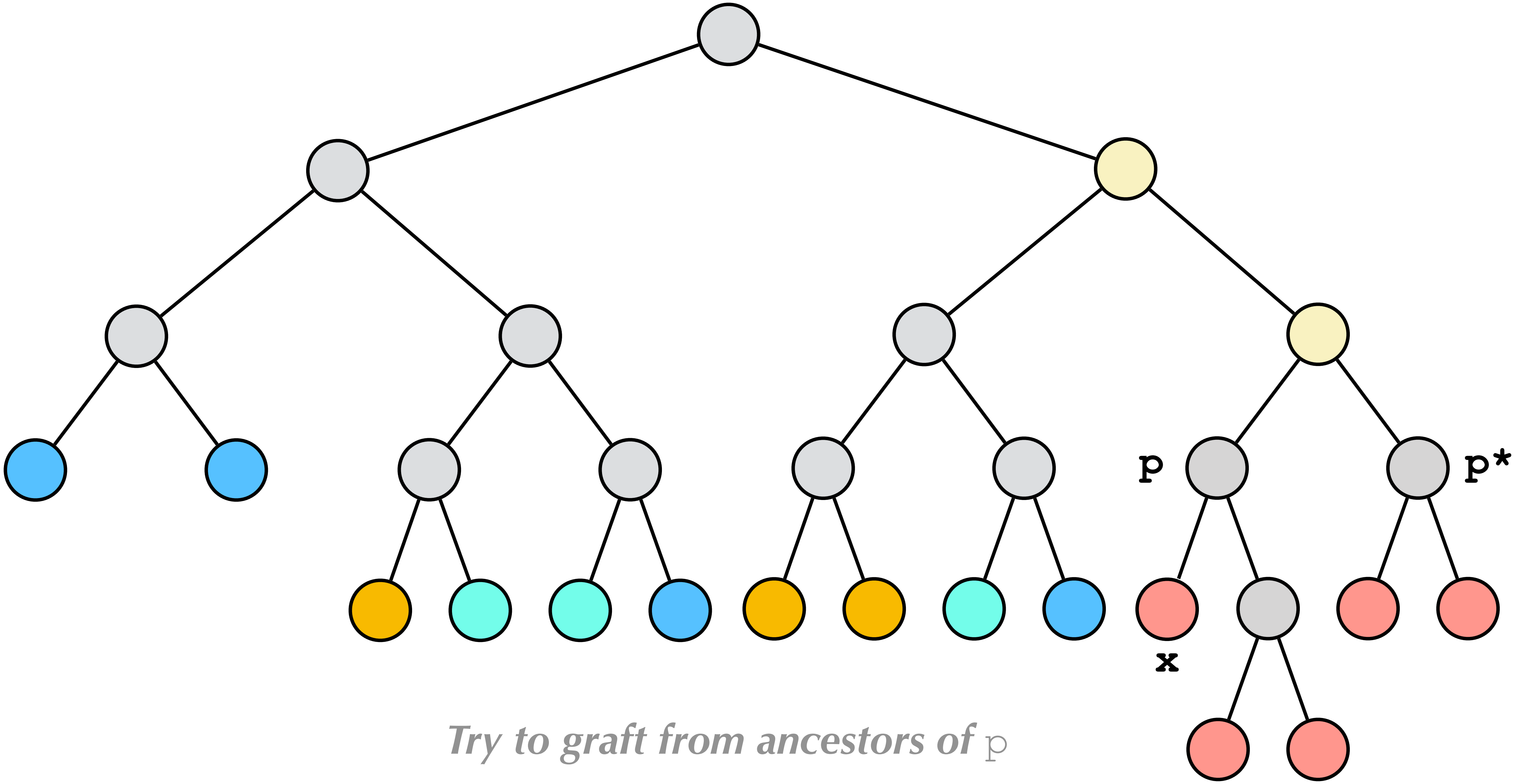
GRINCH



graft if successful

GRINCH

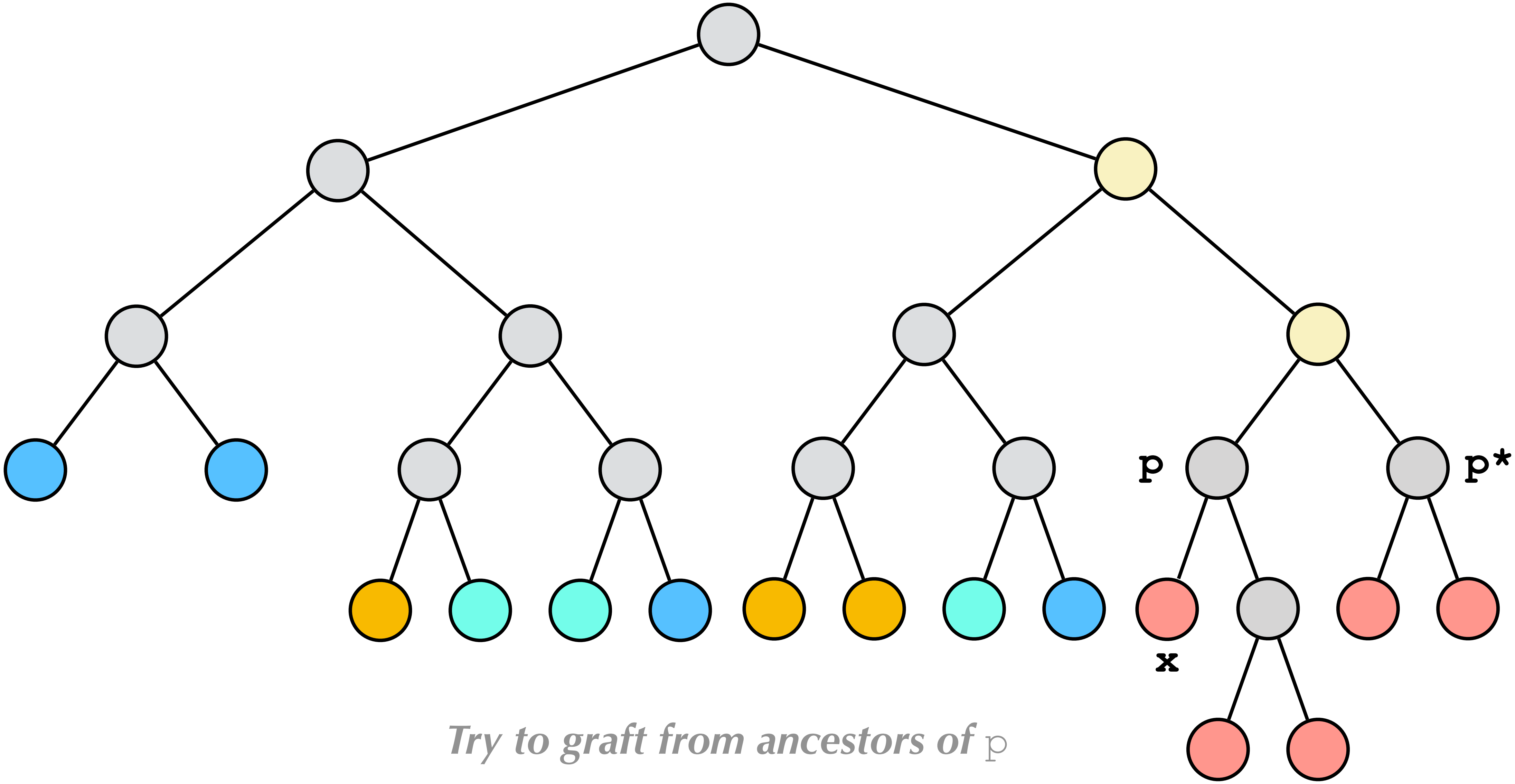
```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



Try to graft from ancestors of p

GRINCH

```
def insert(x, g):
  l = nearest_neighbor(x)
  p = make_sib(l, x)
  while g(sib(x), aunt(x)) > g(sib(x), x):
    rotate(x)
  p = parent(x)
  while p != null:
    try_graft(p)
    p = parent(p)
```



Try to graft from ancestors of p

Outline

1. Introduction

2. Proposed methodology

3. Experimental Results

4. Experimental Analysis

5. Theoretical Results

Large Scale Hierarchical Clustering Experiments

We compare GRINCH to:

ONLINE GRINCH without rotate or graft procedures

ROTATE GRINCH without the graft procedure

PERCH [Kobren et al, 2017] Efficient, highly performant bounding box-based incremental method

Mini-batch HAC Streaming variant of agglomerative clustering

HAC Highly performant, but not scalable bottom up hierarchical agglomerative algorithm

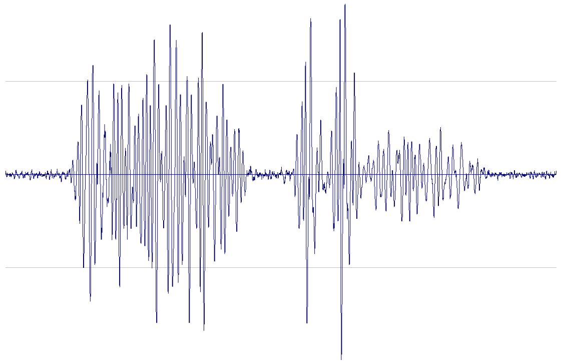
Large Scale Hierarchical Clustering Experiments

We evaluate GRINCH on:

ALOI

100K Points
1000 Clusters

Speaker



36K Points
5K Clusters

ImageNet

50K Subset
1000 Clusters

100K Subset
17K Clusters

Covertype

500K Points
7 Clusters

Large Scale Hierarchical Clustering Experiments

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI
MB-HAC	0.30 ± 0.002
PERCH	0.44 ± 0.004

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI
MB-HAC	0.30 ± 0.002
PERCH	0.44 ± 0.004
ONLINE	0.435 ± 0.004
ROTATE	0.476 ± 0.004

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI
MB-HAC	0.30 ± 0.002
PERCH	0.44 ± 0.004
ONLINE	0.435 ± 0.004
ROTATE	0.476 ± 0.004
GRINCH	**0.504 ± 0.002**

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI	Speaker
MB-HAC	0.30 ± 0.002	0.01 ± 0.002
PERCH	0.44 ± 0.004	0.37 ± 0.002
ONLINE	0.435 ± 0.004	0.317 ±0.002
ROTATE	0.476 ± 0.004	0.407 ± 0.003
GRINCH	**0.504 ± 0.002**	0.480 ± 0.003

Large Scale Hierarchical Clustering Experiments

Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI	Speaker
MB-HAC	0.30 ± 0.002	0.01 ± 0.002
PERCH	0.44 ± 0.004	0.37 ± 0.002
ONLINE	0.435 ± 0.004	0.317 ±0.002
ROTATE	0.476 ± 0.004	0.407 ± 0.003
GRINCH	**0.504 ± 0.002**	0.480 ± 0.003
HAC	-	**0.55**

Large Scale Hierarchical Clustering Experiments

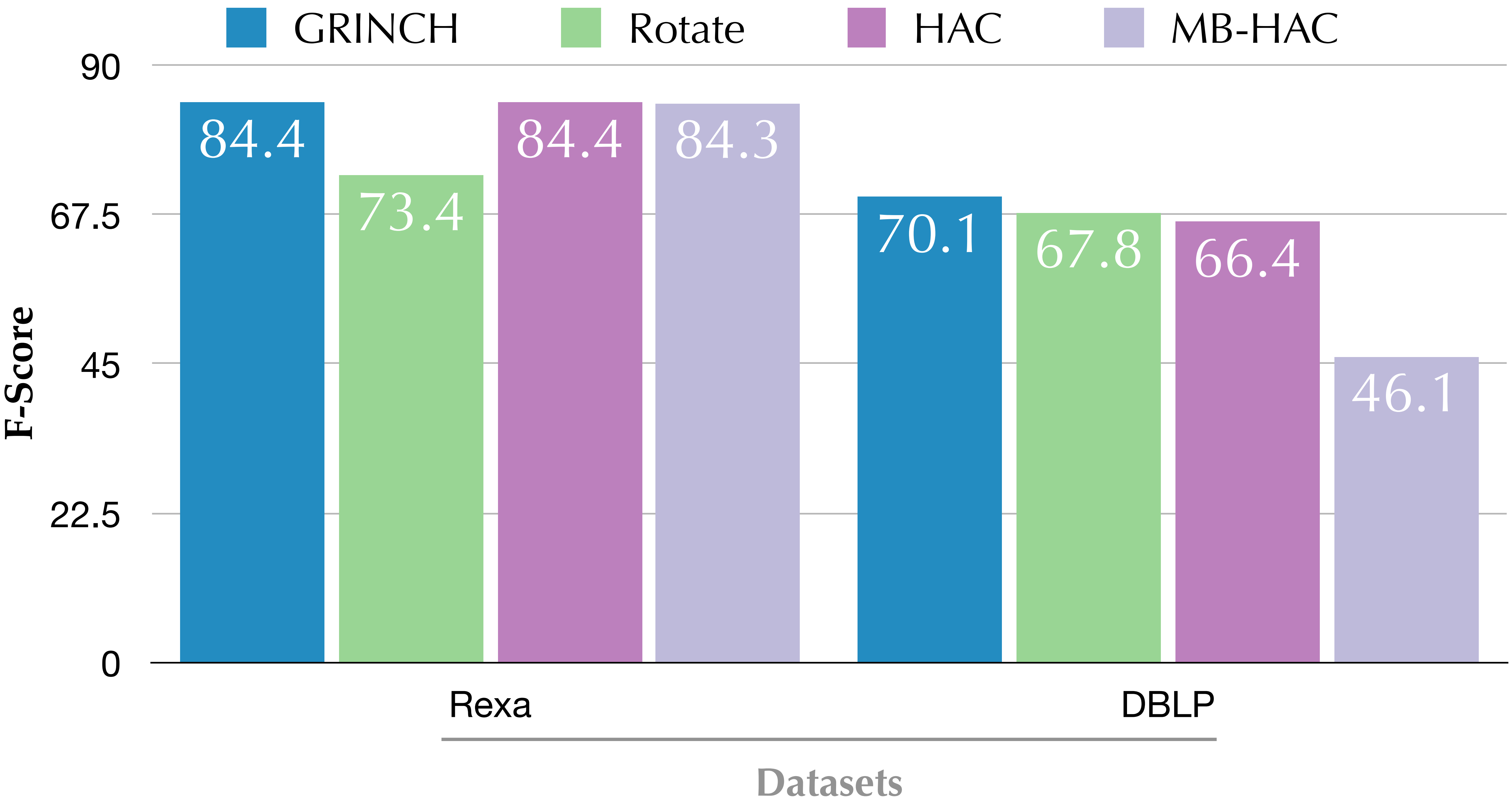
Datasets have **ground truth flat clustering**

Measure **Dendrogram Purity**

Methods use **Average Linkage**

	ALOI	**Speaker**	**ILSVRC (50K)**
MB-HAC	0.30 ± 0.002	0.01 ± 0.002	0.43 ± 0.005
PERCH	0.44 ± 0.004	0.37 ± 0.002	0.53 ± 0.003
ONLINE	0.435 ± 0.004	0.317 ±0.002	0.527 ± 0.004
ROTATE	0.476 ± 0.004	0.407 ± 0.003	0.545 ± 0.004
GRINCH	**0.504 ± 0.002**	0.480 ± 0.003	**0.557 ± 0.003**
HAC	-	**0.55**	0.54

Author Coreference



Outline

1. Introduction

2. Proposed methodology

3. Experimental Results

4. **Experimental Analysis**

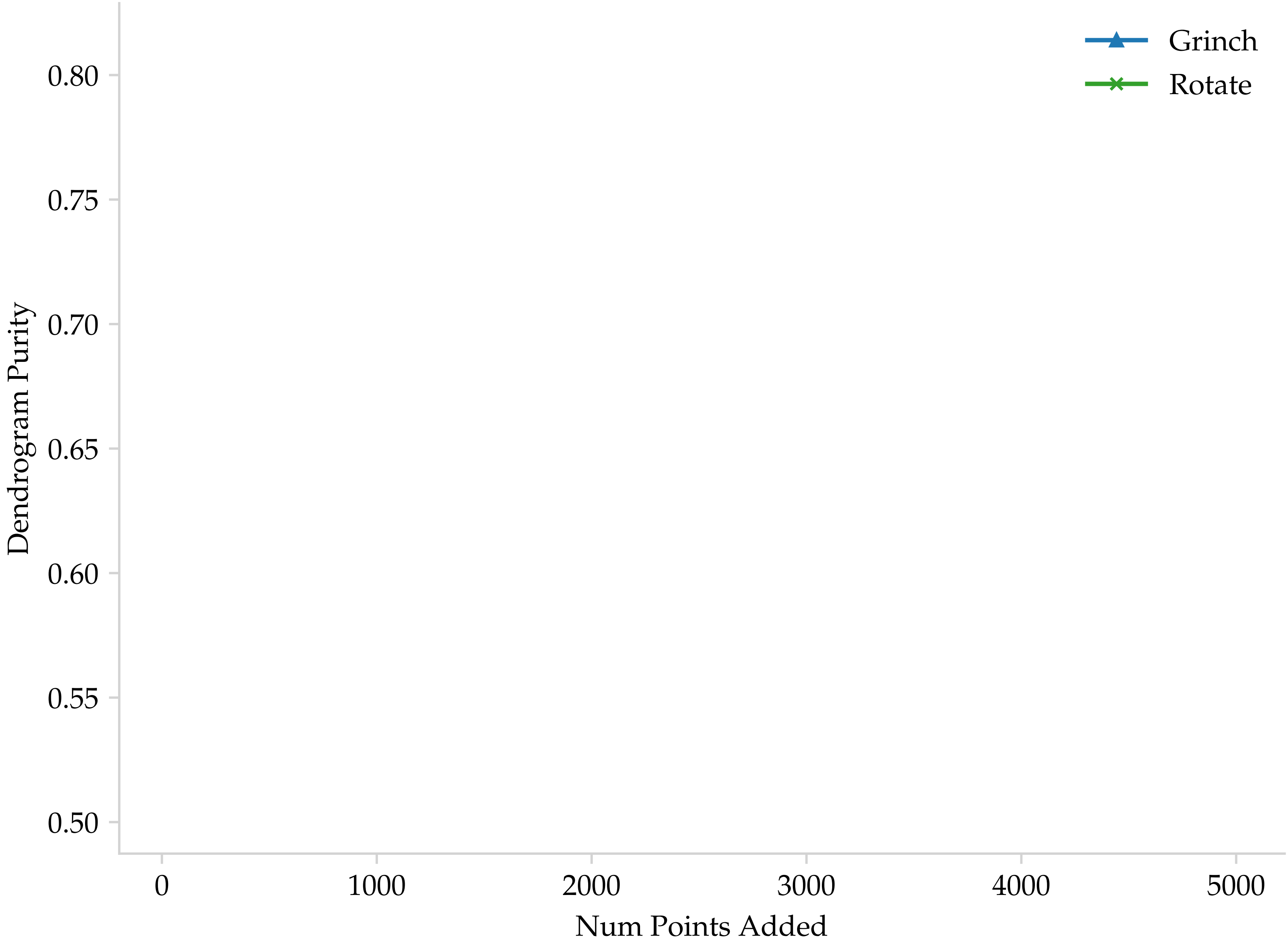
5. Theoretical Results

Importance of Grafting

Importance of Grafting

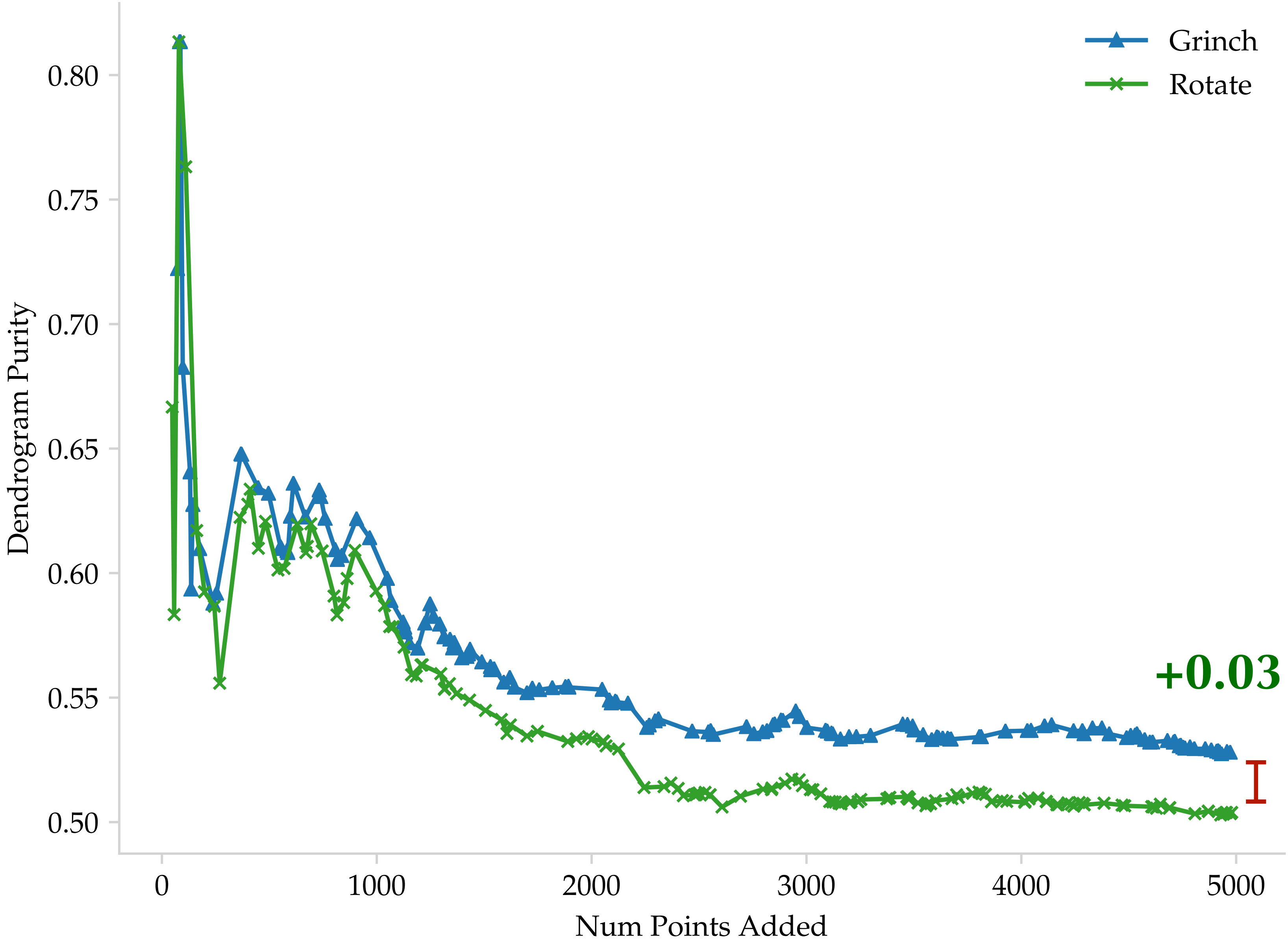
first 5000 points of ALOI dataset

Importance of Grafting



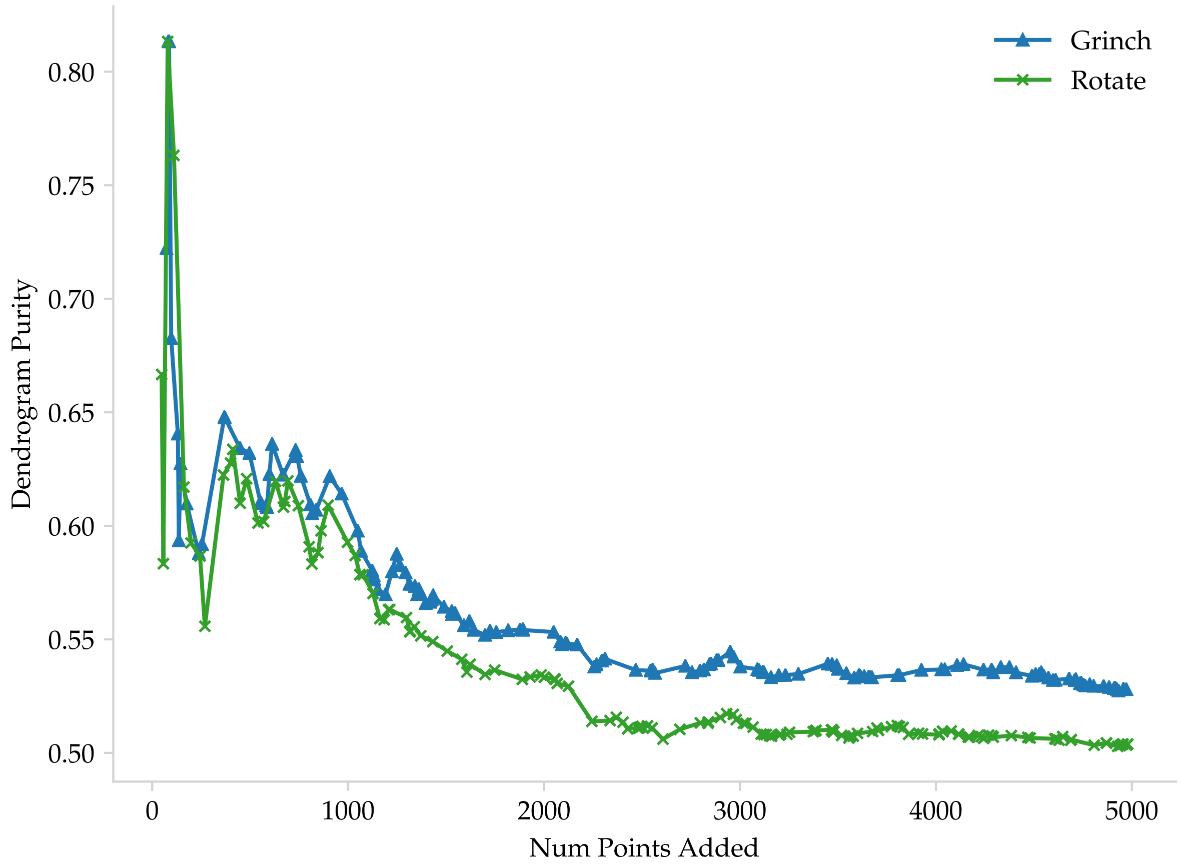
first 5000 points of ALOI dataset

Importance of Grafting



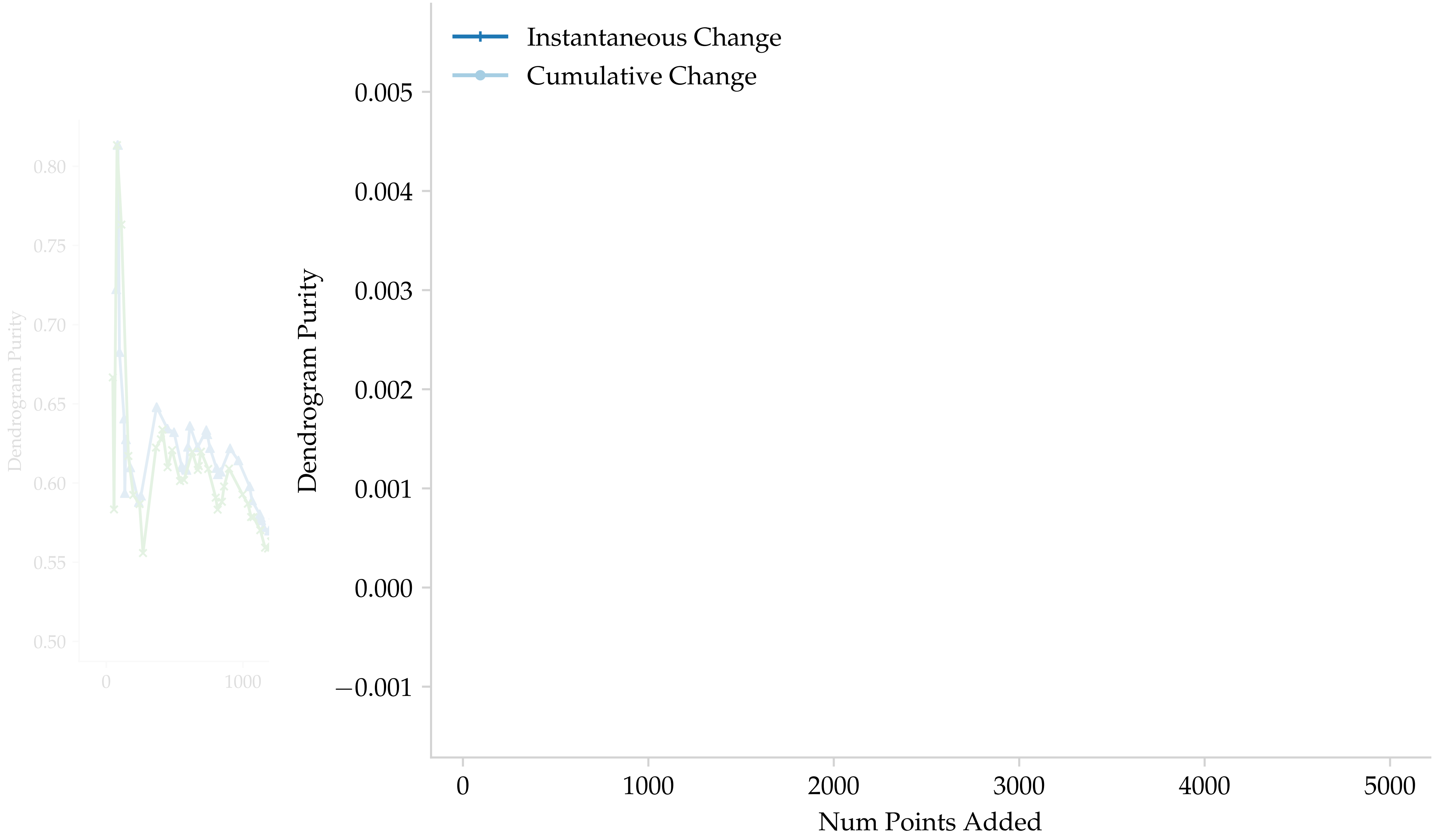
first 5000 points of ALOI dataset

Importance of Grafting



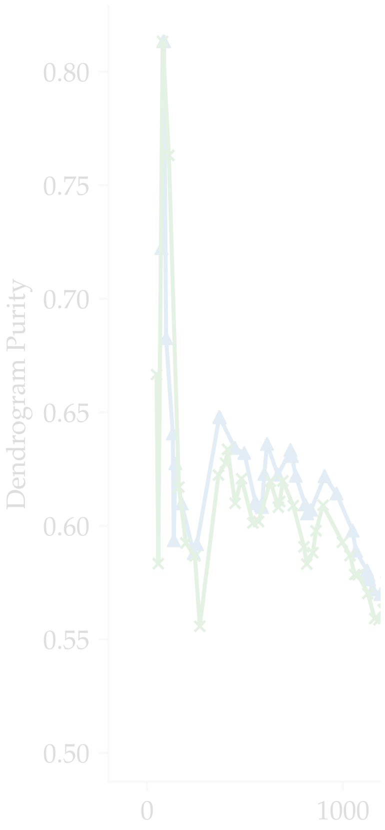
first 5000 points of ALOI dataset

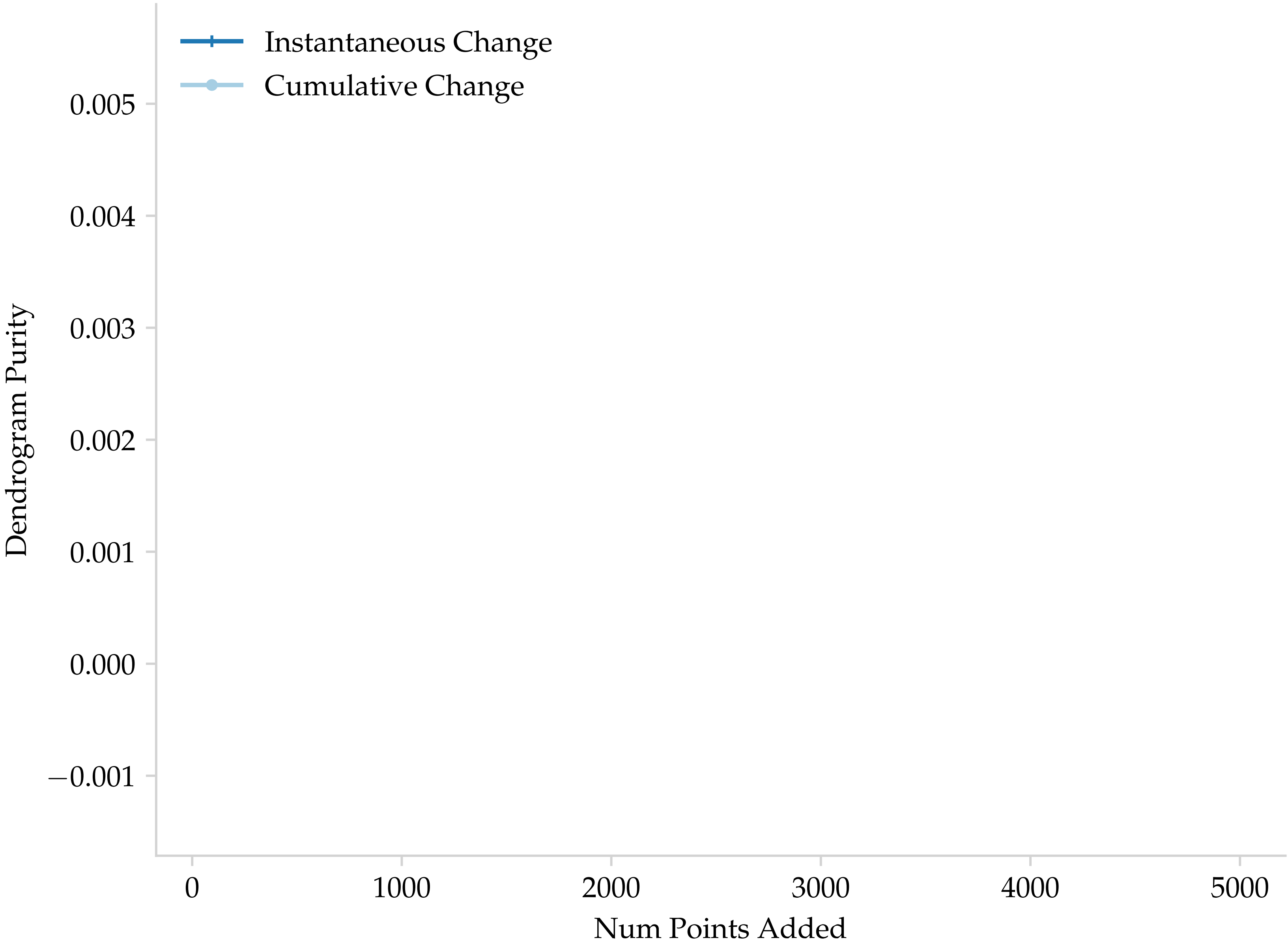
Importance of Grafting



first 5000 points of ALOI dataset

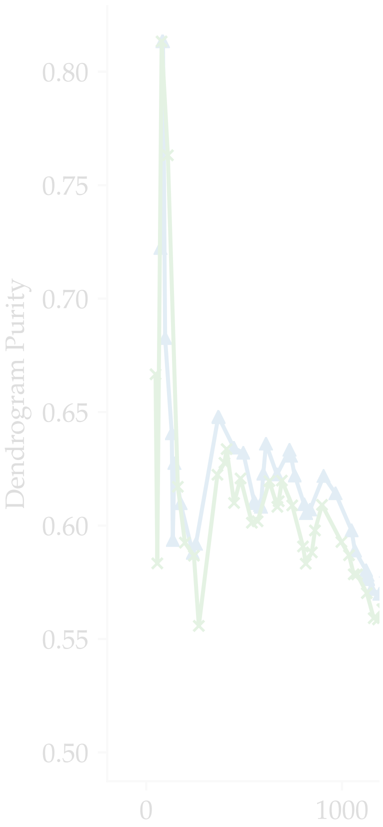
Importance of Grafting

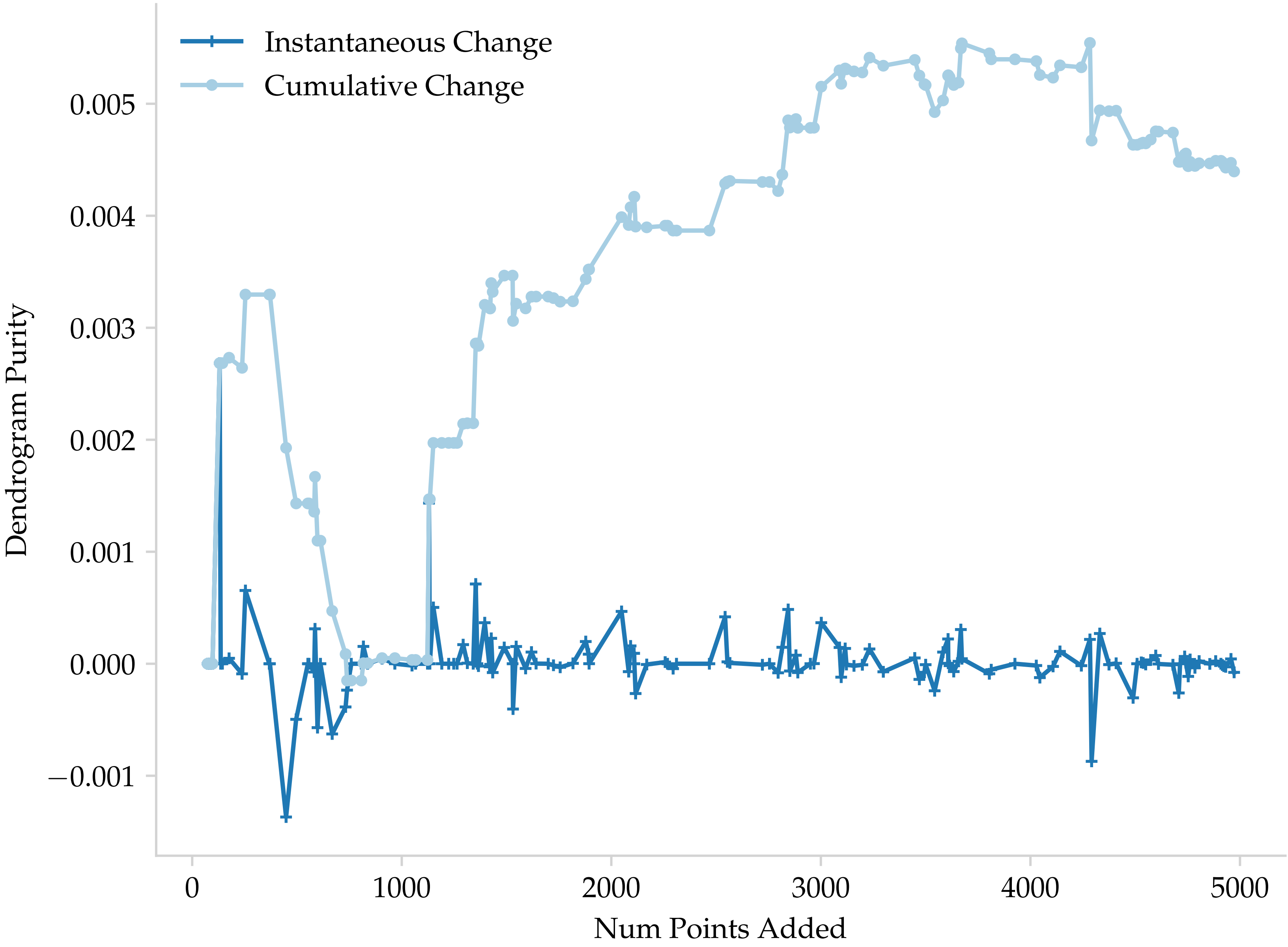




first 5000 points of ALOI dataset

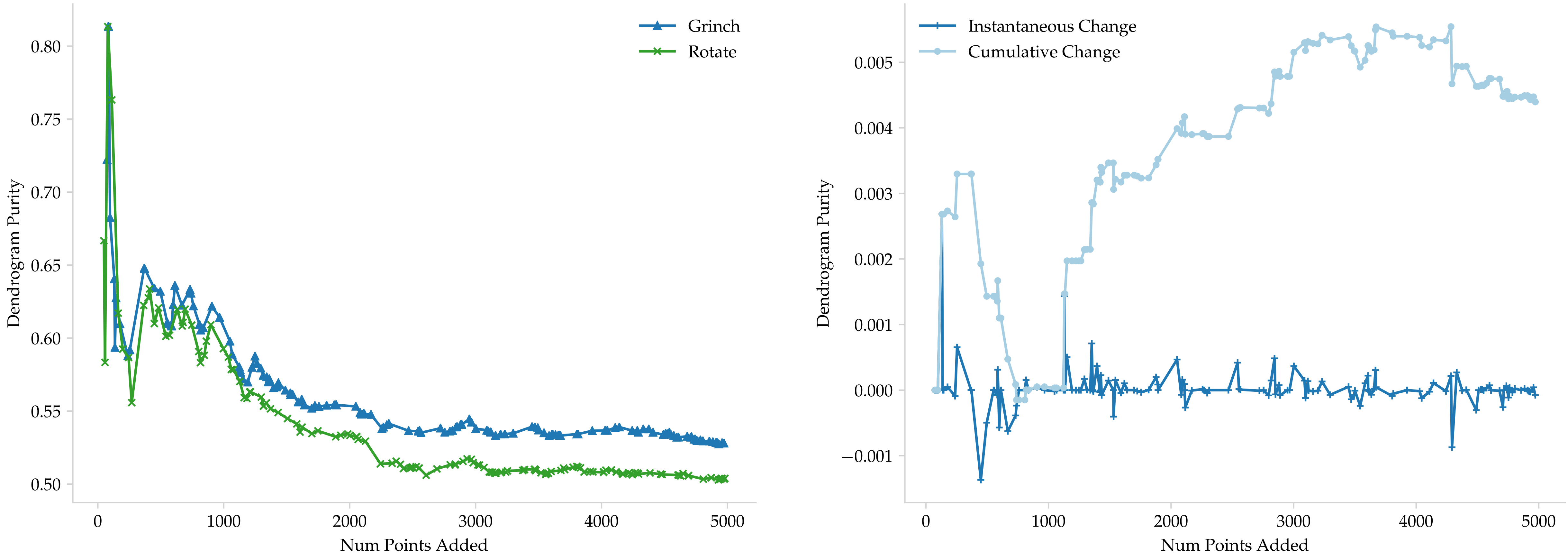
Importance of Grafting





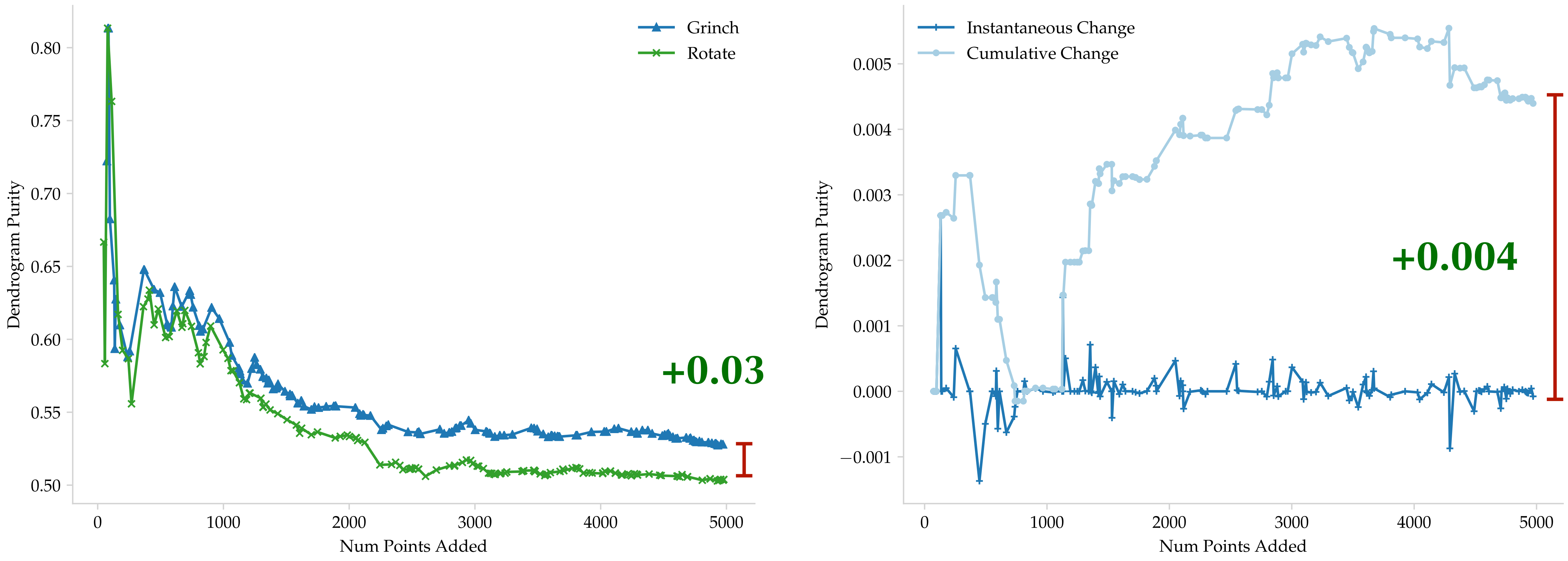
first 5000 points of ALOI dataset

Importance of Grafting



first 5000 points of ALOI dataset

Importance of Grafting



first 5000 points of ALOI dataset

Outline

1. Introduction

2. Proposed methodology

3. Experimental Results

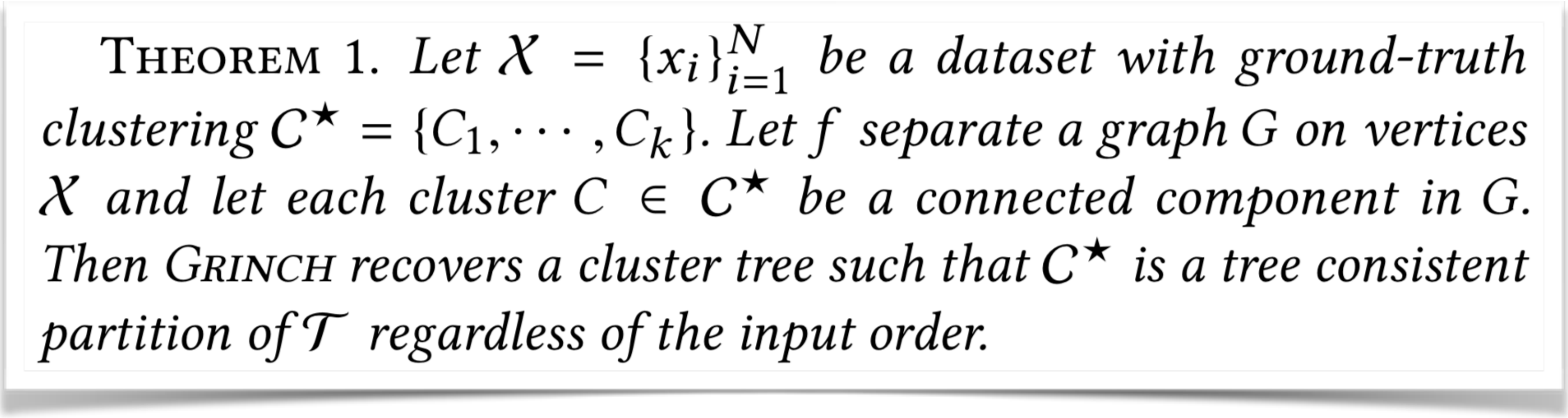
4. **Experimental Analysis**

5. Theoretical Results

Theoretical Results

New separation assumption— ***model-based separation***: **significantly more general** than typical assumptions.

We prove that for datasets satisfying model-based separation, GRINCH will recover a hierarchical clustering with **dendrogram purity equal to 1.0** regardless of **input order.**

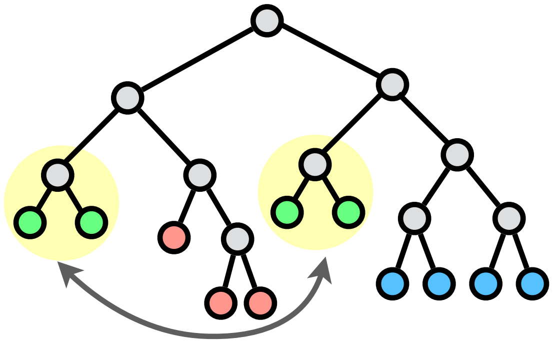


THEOREM 1. *Let $\mathcal{X} = \{x_i\}_{i=1}^{N}$ be a dataset with ground-truth clustering $\mathcal{C}^\star = \{C_1, \cdots, C_k\}$. Let f separate a graph G on vertices $\mathcal{X}$ and let each cluster $C \in \mathcal{C}^\star$ be a connected component in G. Then* GRINCH *recovers a cluster tree such that $\mathcal{C}^\star$ is a tree consistent partition of $\mathcal{T}$ regardless of the input order.*

Summary

GRINCH

Grafting and
Rotation-based
INCremental
Hierarchical
clustering



Scalable, **incremental** hierarchical clustering alternative to agglomerative clustering.

Uses novel **tree re-arrangements (rotate, graft)** to efficiently reconsider past decisions.

Empirical results validating **quality** of GRINCH's clusterings

Theoretical results proving correctness of GRINCH

Thanks to my collaborators!

Ari Kobren*

Akshay Krishnamurthy

Michael Glass

Andrew McCallum

UMass Amherst

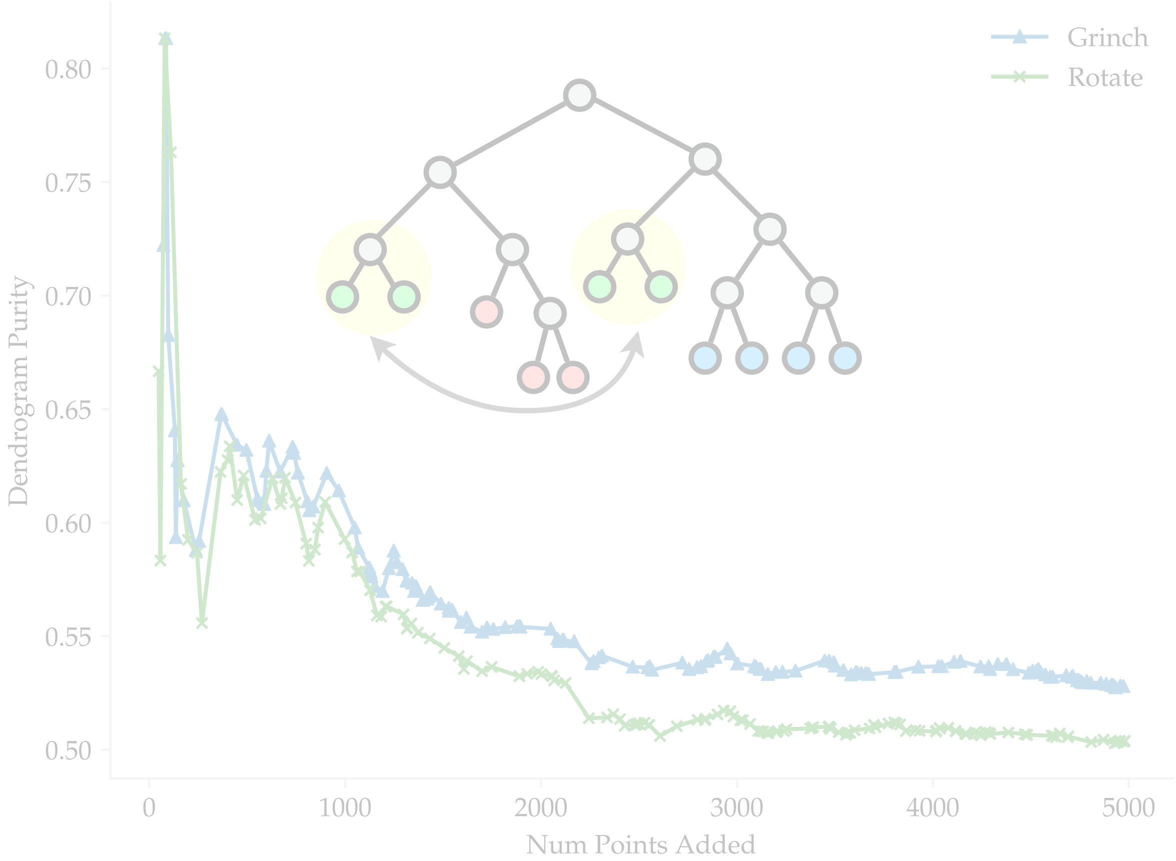
College of Information and Computer Sciences

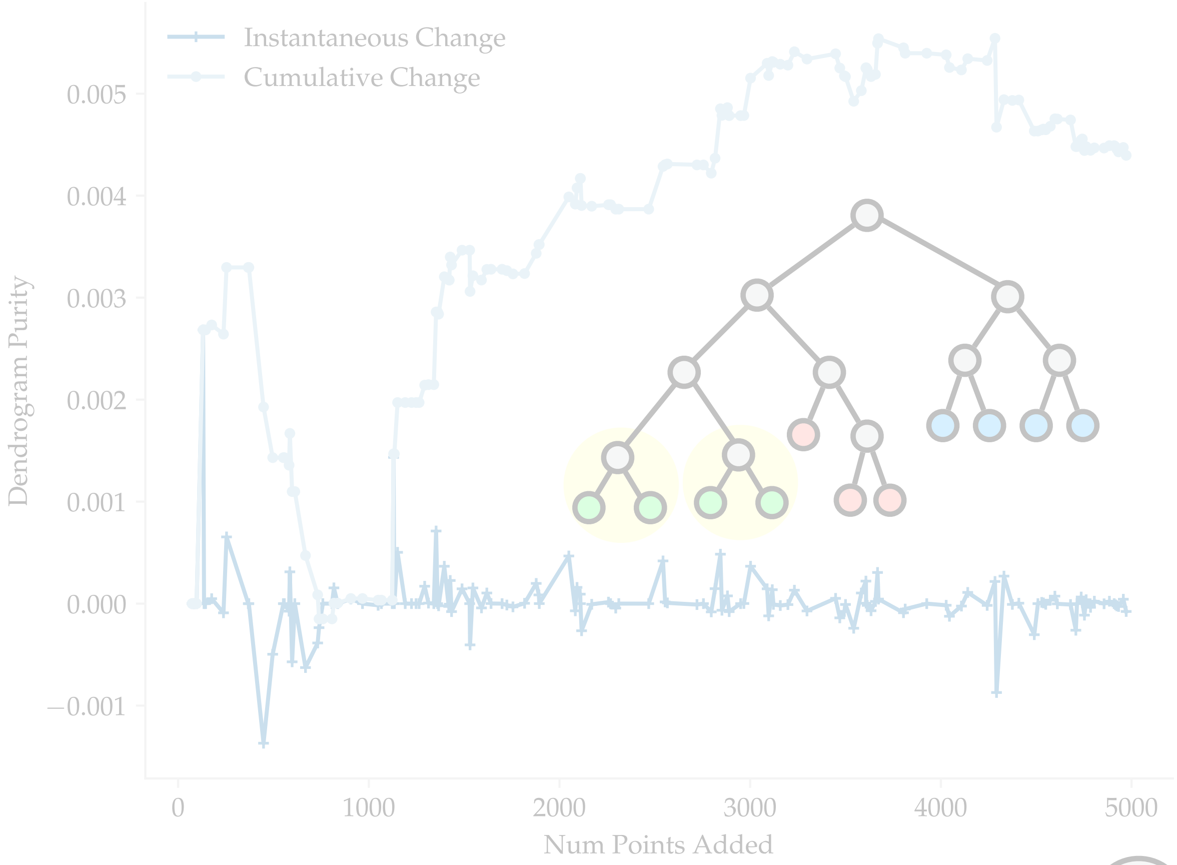
IESL

Microsoft Research

IBM

*The first two authors contributed equally.





Thanks! Questions?