fast branch-and-bound for rule lists with a symmetry-aware cache

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, Cynthia Rudin

rule lists

- one-sided decision trees
- Boolean functions for binary classification

rule lists

will the person stay inside or go outside?

- **if** (using computer) **then predict** (stay inside)
- **else if** (time is between 11PM-7AM) **then predict** (stay inside)
- **else if** (sunny **and** not raining) **then predict** (go outside)
- **else if** (temperature > 55 F) **then predict** (go outside)

else predict (stay inside)

rule lists

will the person stay inside or go outside?

if (using computer) **then predict** (stay inside)

else if (time is between 11PM-7AM) **then predict** (stay inside)

else if (sunny **and** not raining) **then predict** (go outside)

else if (temperature > 55 F) **then predict** (go outside)

else predict (stay inside)

- competitive with decision trees
- rich features (e.g., conjunctions) => expressive
- human-interpretable (industry, medicine)

learning rule lists

- typical algorithms don't do global optimization
- greedy (fast)
- Monte Carlo (slow)

goal: global optimization via branch-and-bound

selected related work:

- learning decision lists (Rivest, 1987)
- Bayesian rule lists (Letham, Rudin, McCormick, Madigan, 2015) … stroke prediction model
- scalable Bayesian rule lists (Yang, Rudin, Seltzer, 2016) … cache, efficient bit vector operations

minimize over all possible rule lists **RL** :

objective(**RL**) = error(**RL**) + c length(**RL**)

~ 0.01

- **if** (using computer) **then predict** (stay inside)
- **else if** (time is between 11PM-7AM) **then predict** (stay inside)
- **else if** (sunny **and** not raining) **then predict** (go outside)
- **else if** (temperature > 55 F) **then predict** (go outside)

else predict (stay inside)

bounds on the objective drive branch-and-bound

$$\text{objective}(\mathbf{RL}) = \text{error}(\mathbf{RL}) + c\ \text{length}(\mathbf{RL})$$

$$= \text{error}(\mathbf{prefix}) + \text{error}(\mathbf{default}) + c\ \text{length}(\mathbf{prefix})$$

- **if** (using computer) **then predict** (stay inside)
- **else if** (time is between 11PM-7AM) **then predict** (stay inside)
- **else if** (sunny **and** not raining) **then predict** (go outside)
- **else if** (temperature > 55 F) **then predict** (go outside)

else predict (stay inside)

bounds on the objective drive branch-and-bound

objective(**RL**) = error(**RL**) + *c* length(**RL**)

= error(**prefix**) + error(**default**) + *c* length(**prefix**)

>= error(**prefix**) + *c* length(**prefix**)

a rule list that starts with **prefix** can only improve by correcting errors made by the default rule lower bounds monotonically increase as prefixes grow

if (using computer) **then predict** (stay inside)
else predict (go outside)

lower bound( ) = error( ) + c

lower bounds monotonically increase as prefixes grow

● **if** (using computer) **then predict** (stay inside)

else predict (go outside)

lower bound(●) = error(●) + c

● **if** (using computer) **then predict** (stay inside)

○ **else if** (time is between 11PM-7AM) **then predict** (stay inside)

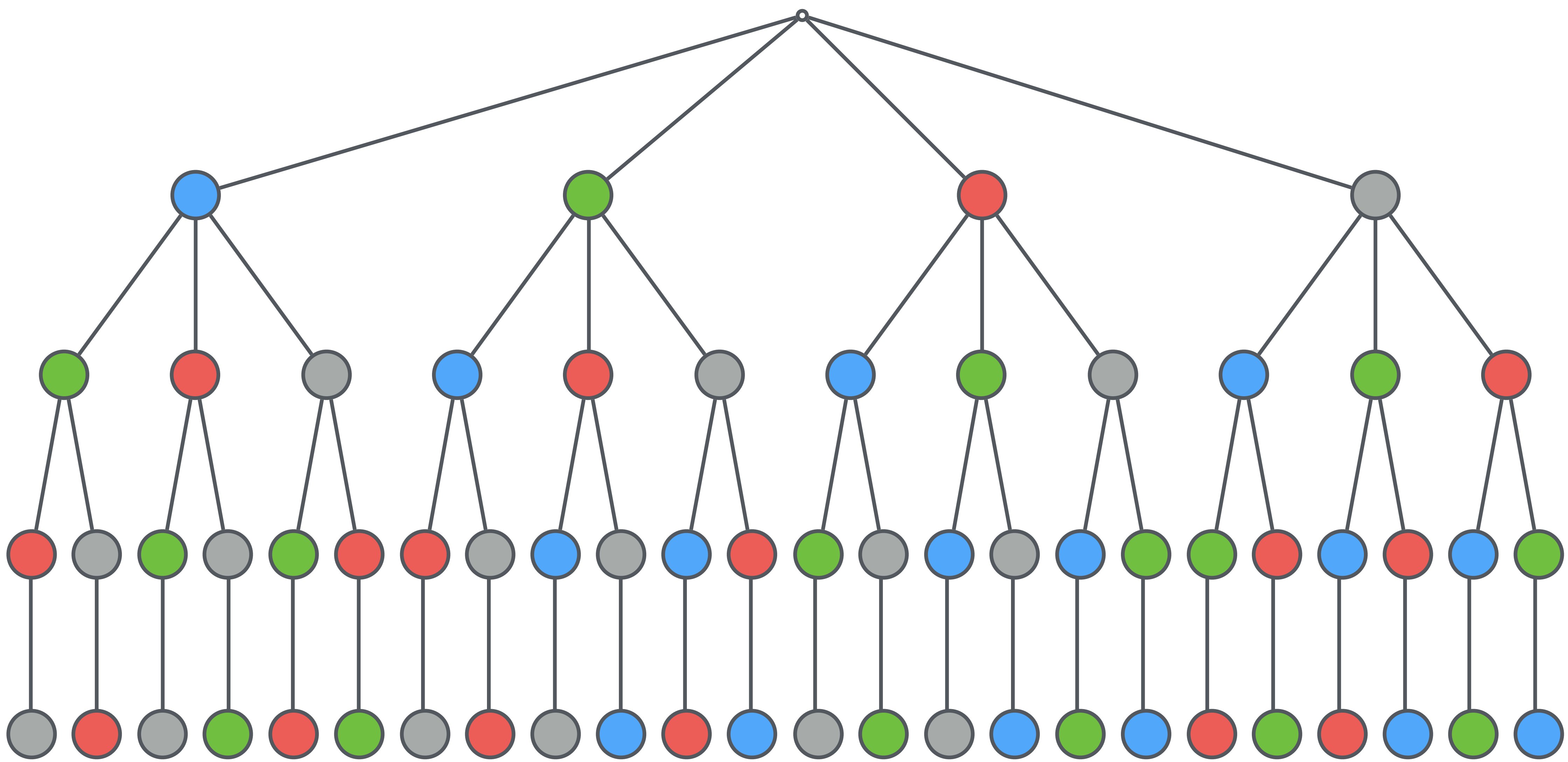
else predict (go outside)

lower bound(●, ○) = error(●) + error(○|●) + 2c

>= error(●) + c

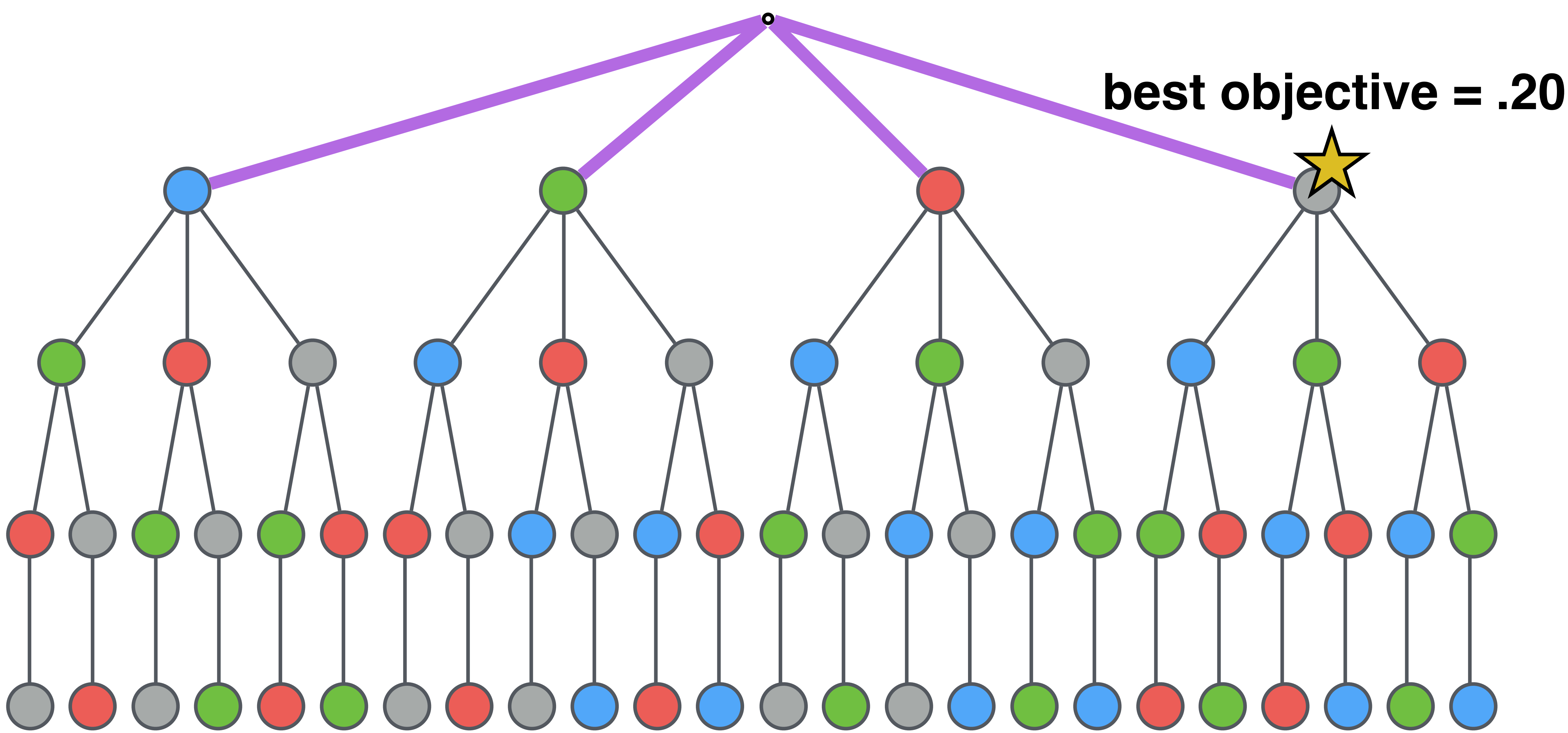
= lower bound(●)

search space = all permutations up to length 4
(paths starting from the root encode prefixes)

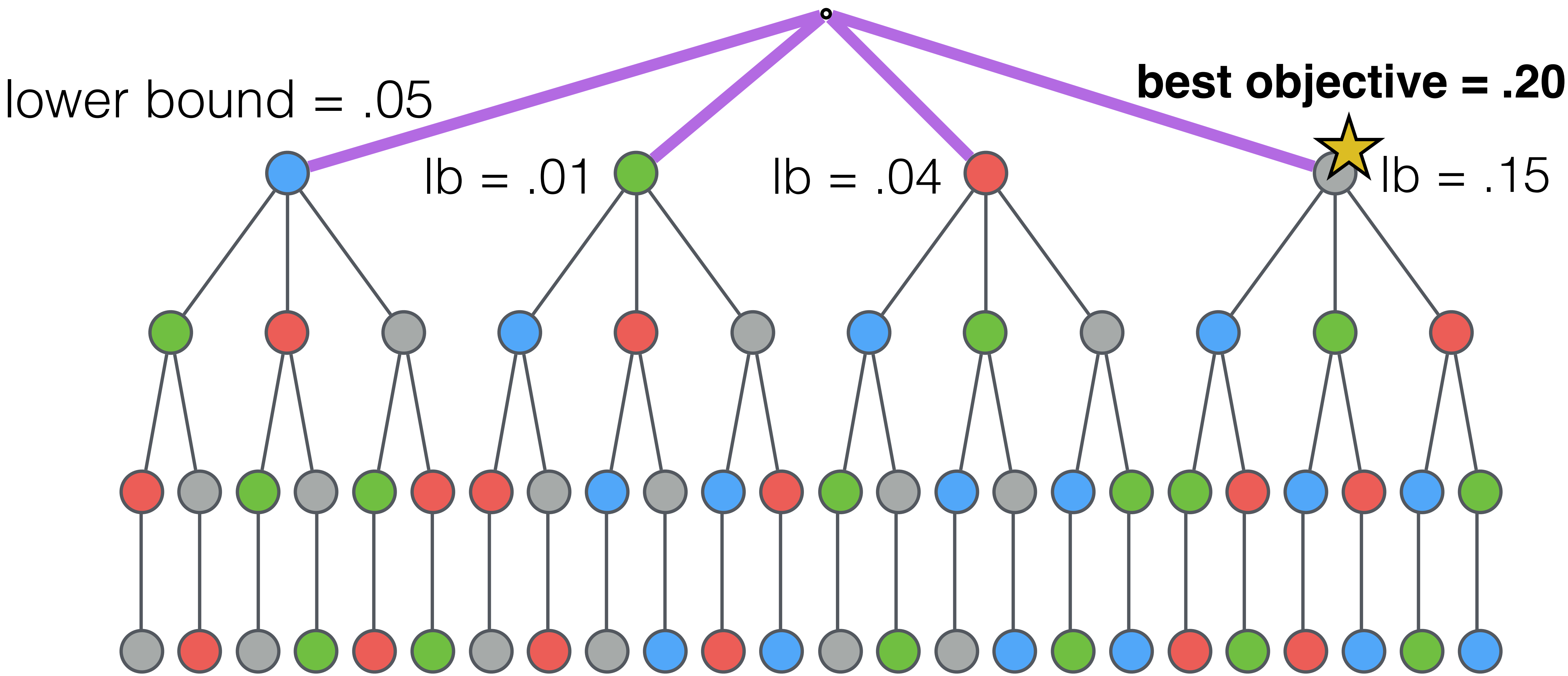


canonical branch-and-bound = breadth-first

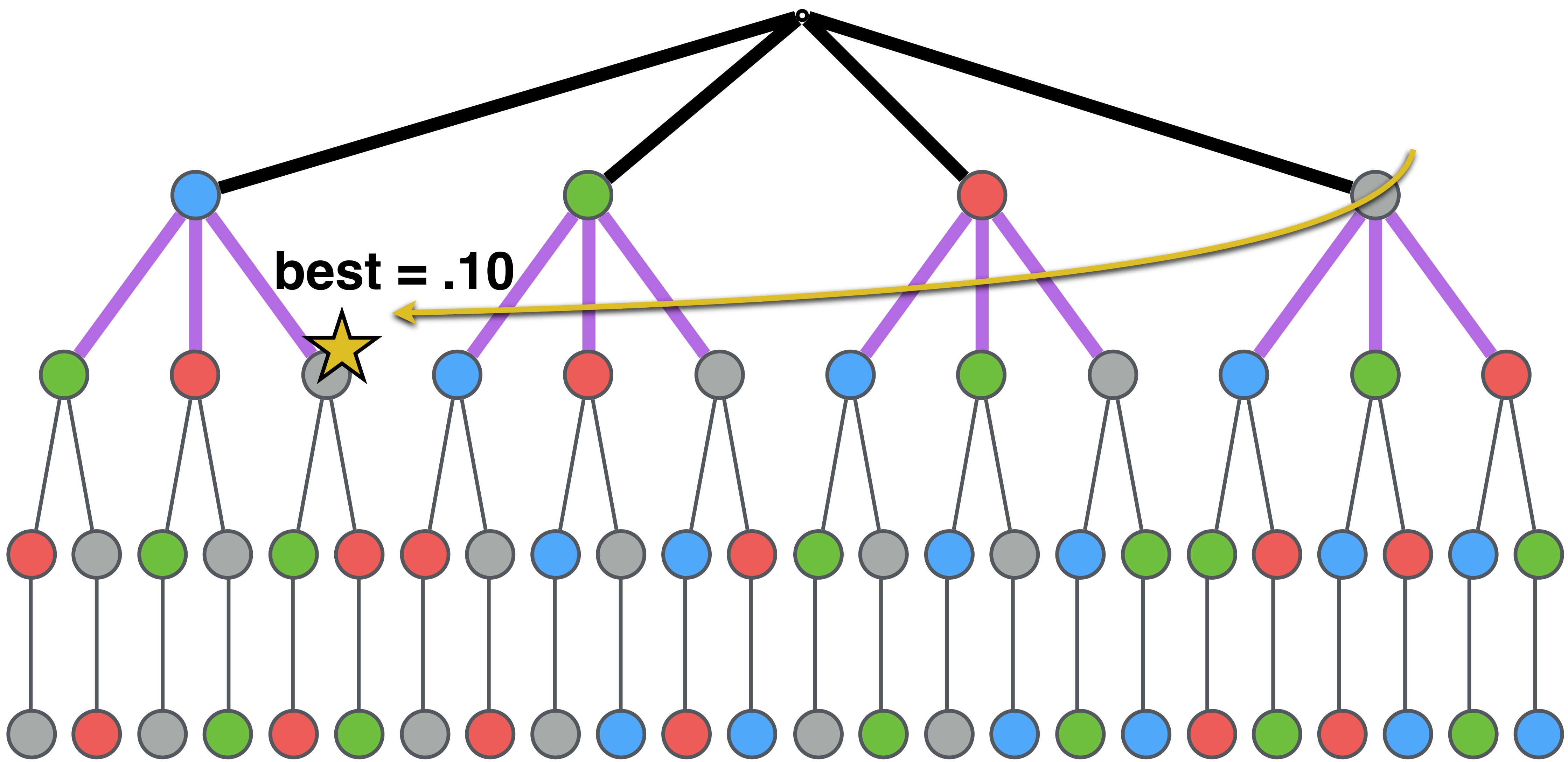
evaluate all prefixes of length 1



keep only prefixes with lower bound < best objective

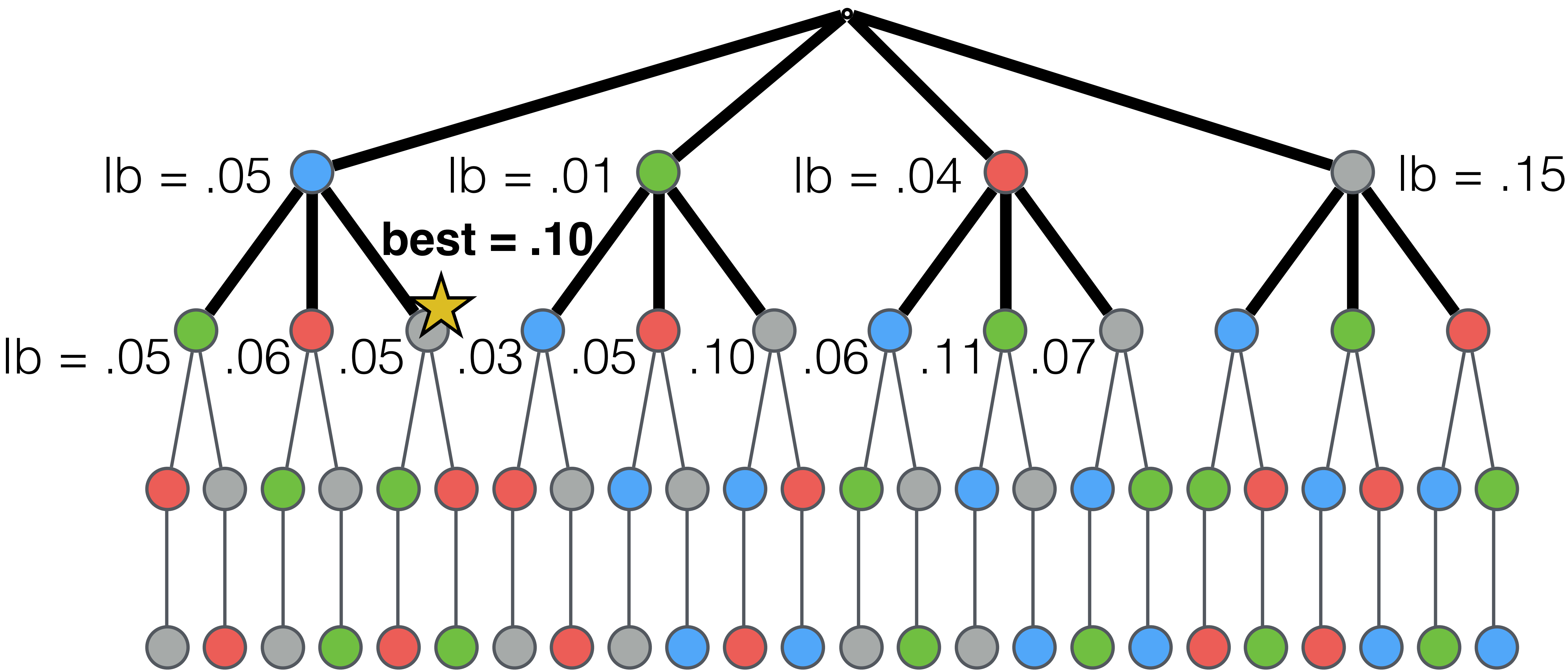


incrementally* grow to all prefixes of length 2

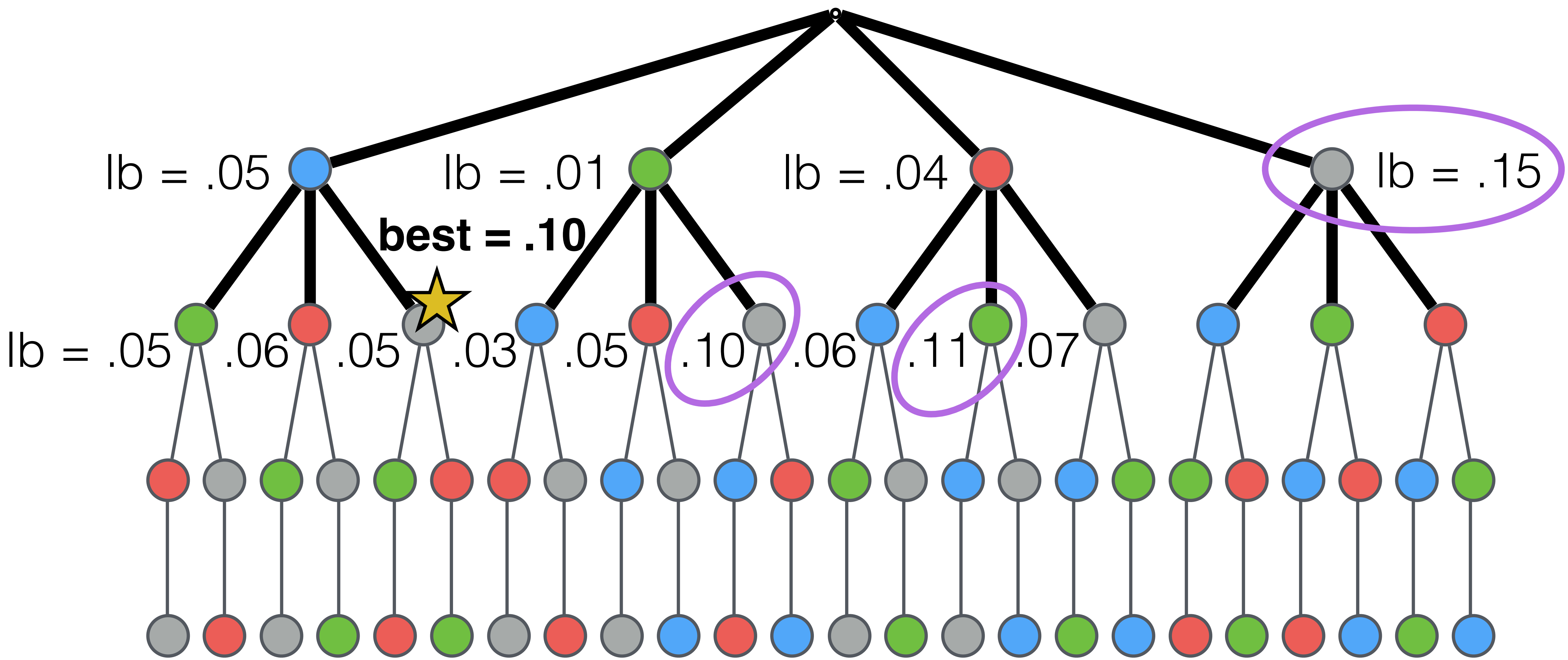


*requires efficient cache data structure (e.g., prefix tree)

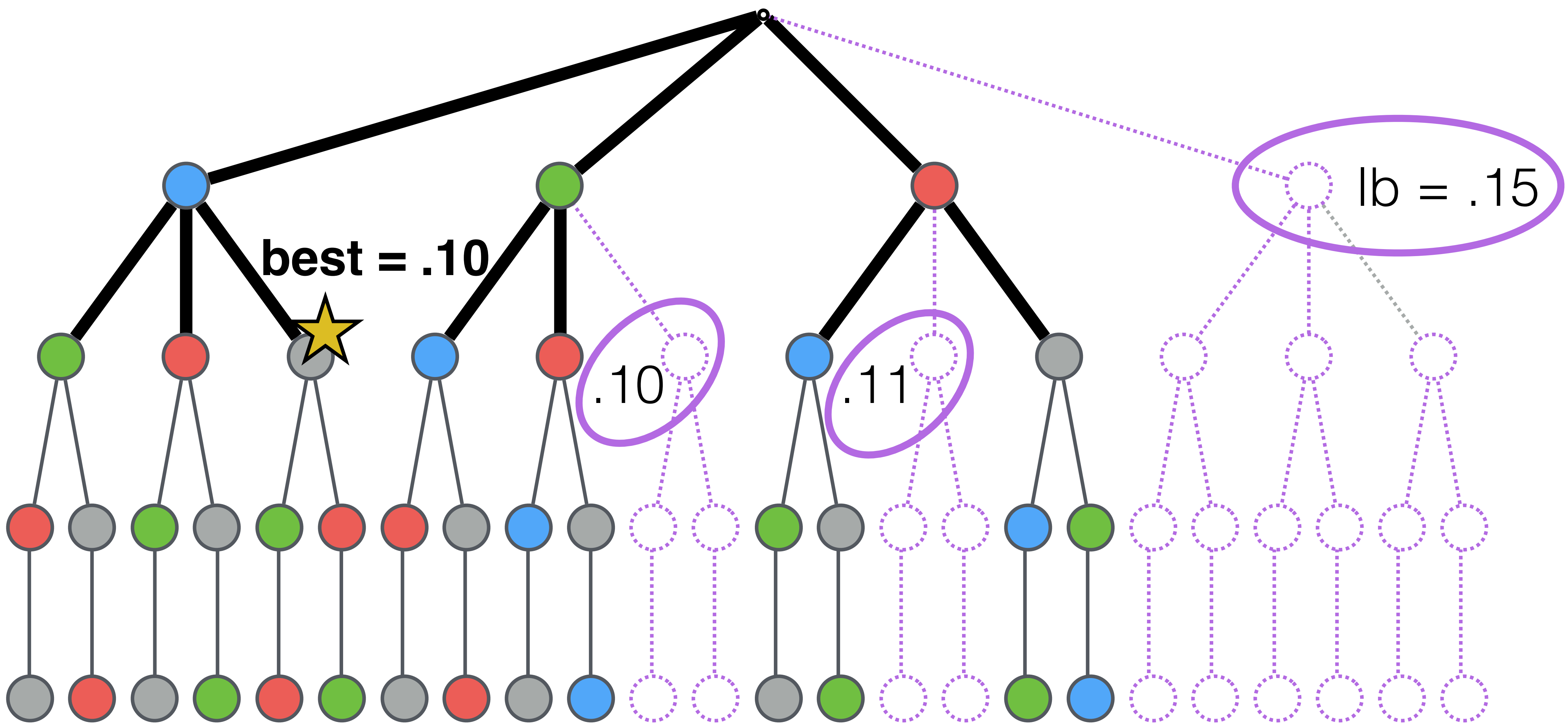
keep all prefixes with lower bound < best objective



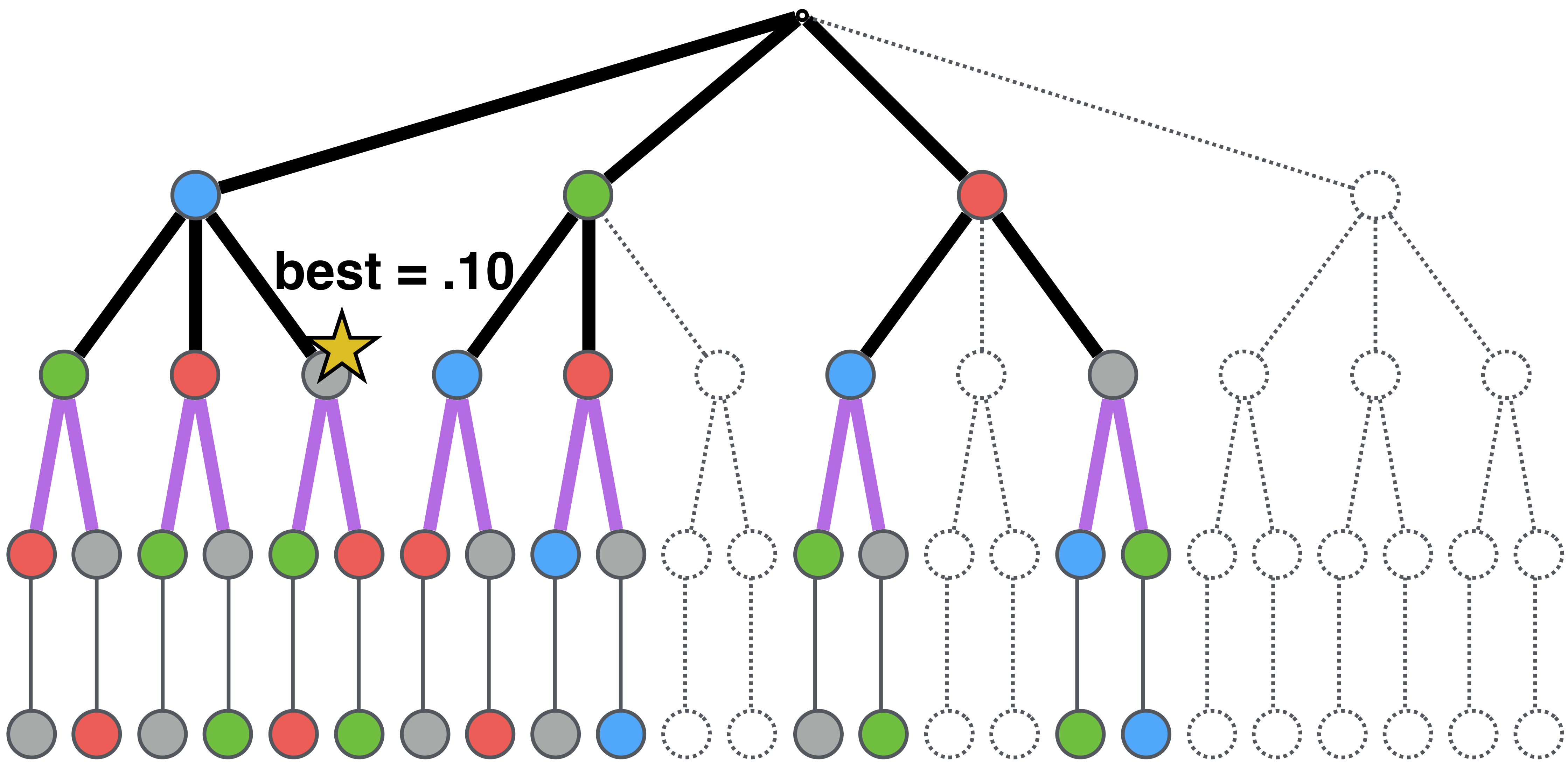
delete prefixes with lower bound >= best objective



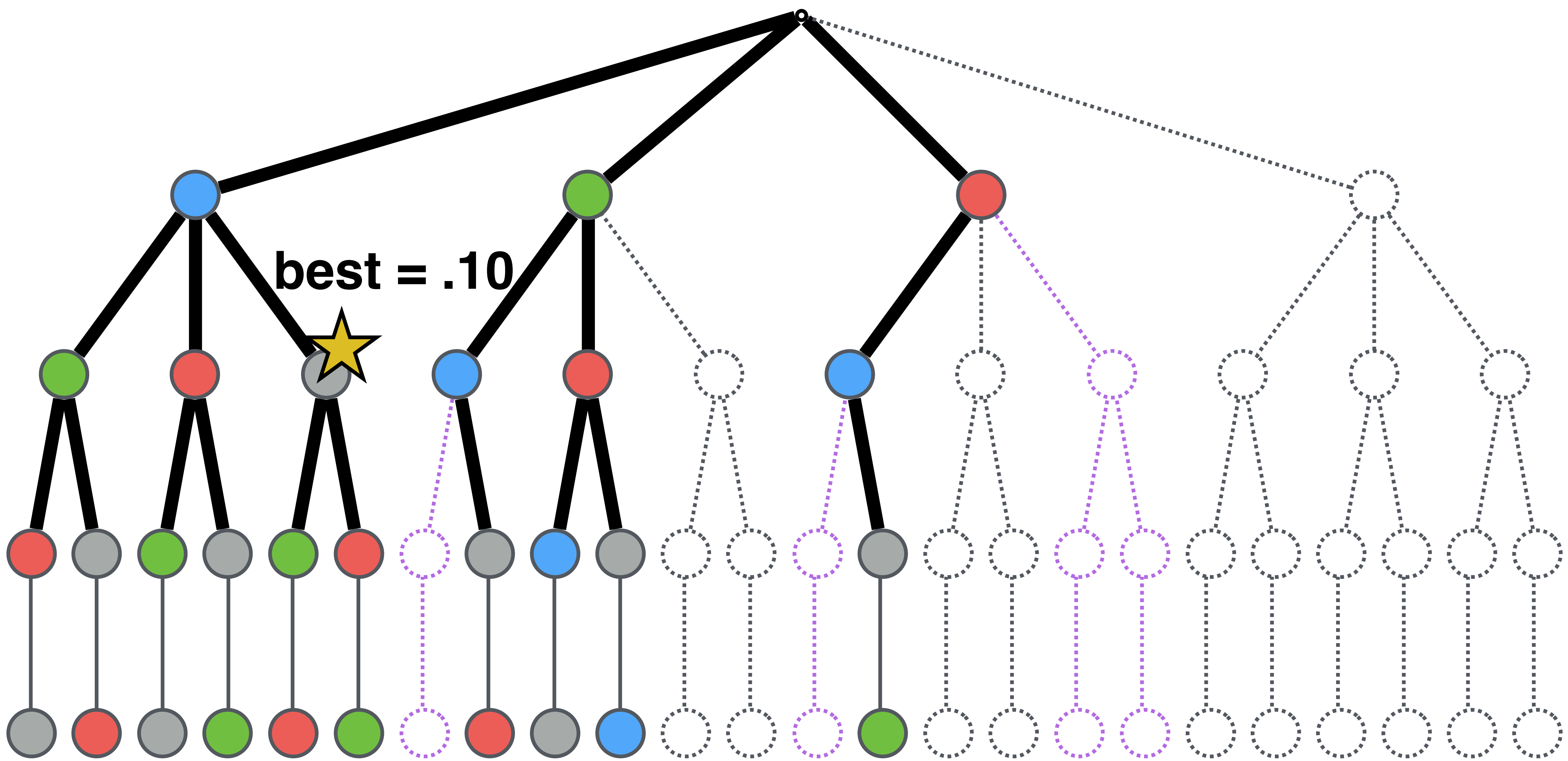
lower bounds let us prune our search space



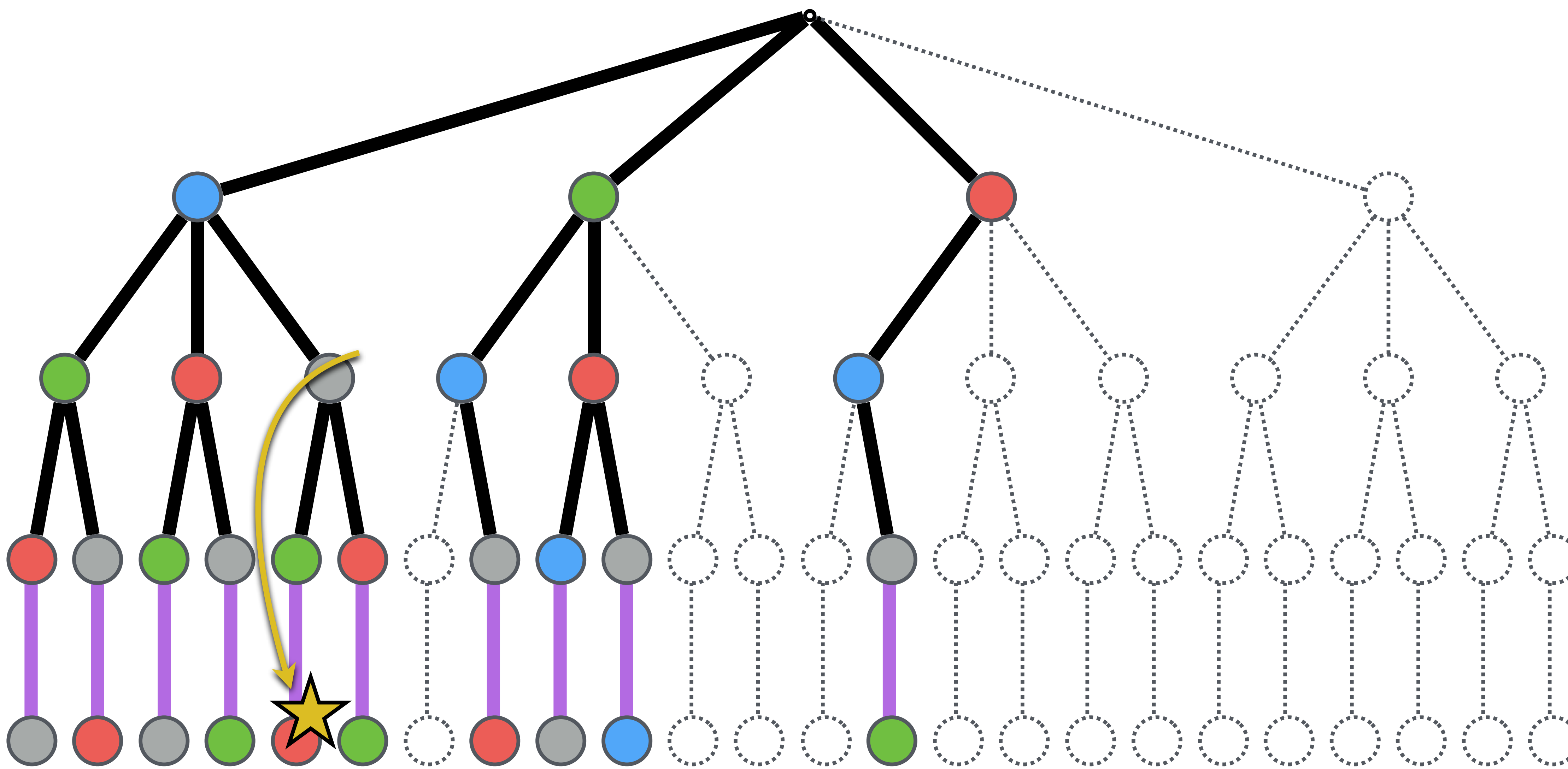
incrementally grow to all prefixes of length 3



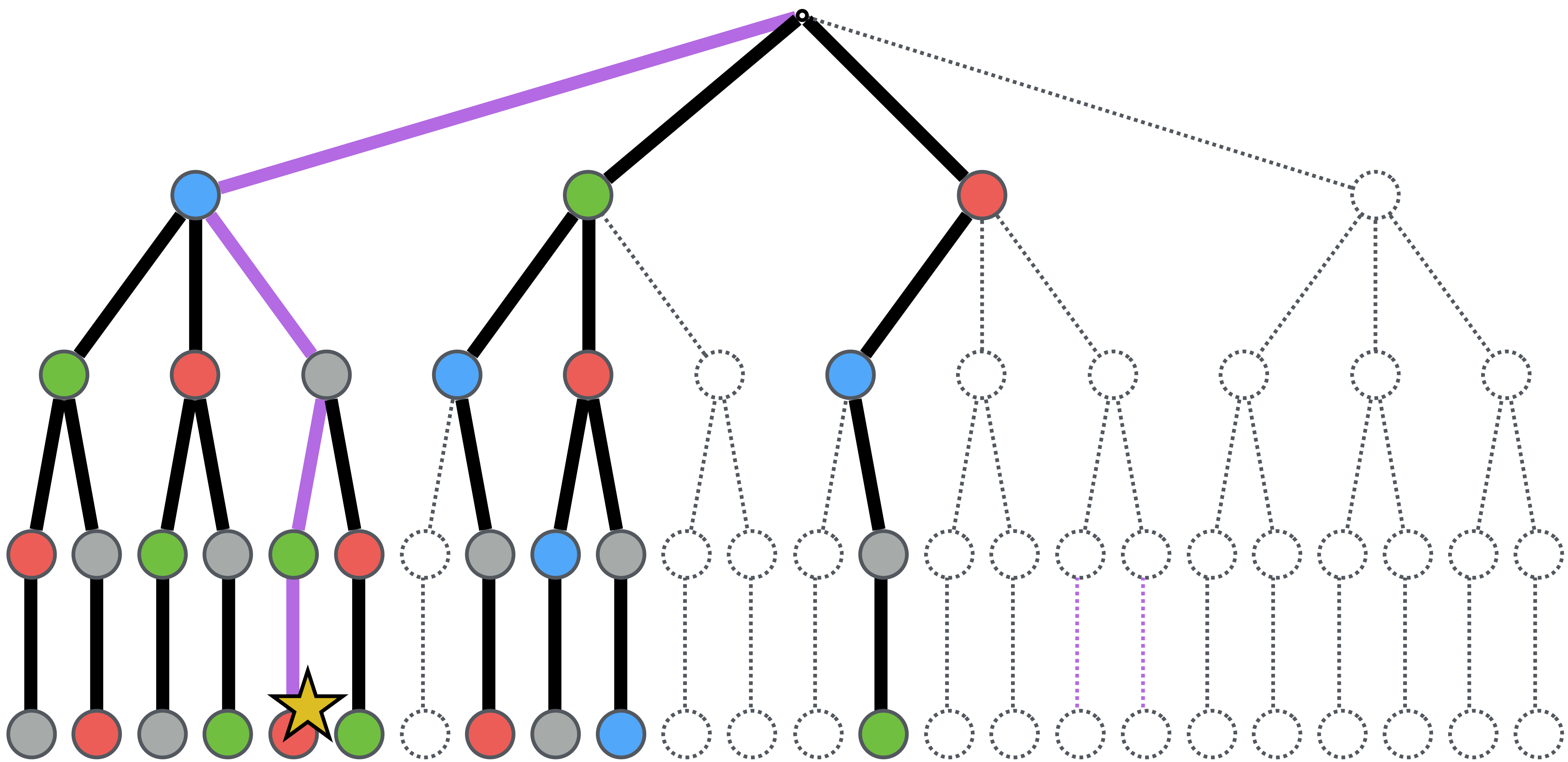
use lower bounds to prune



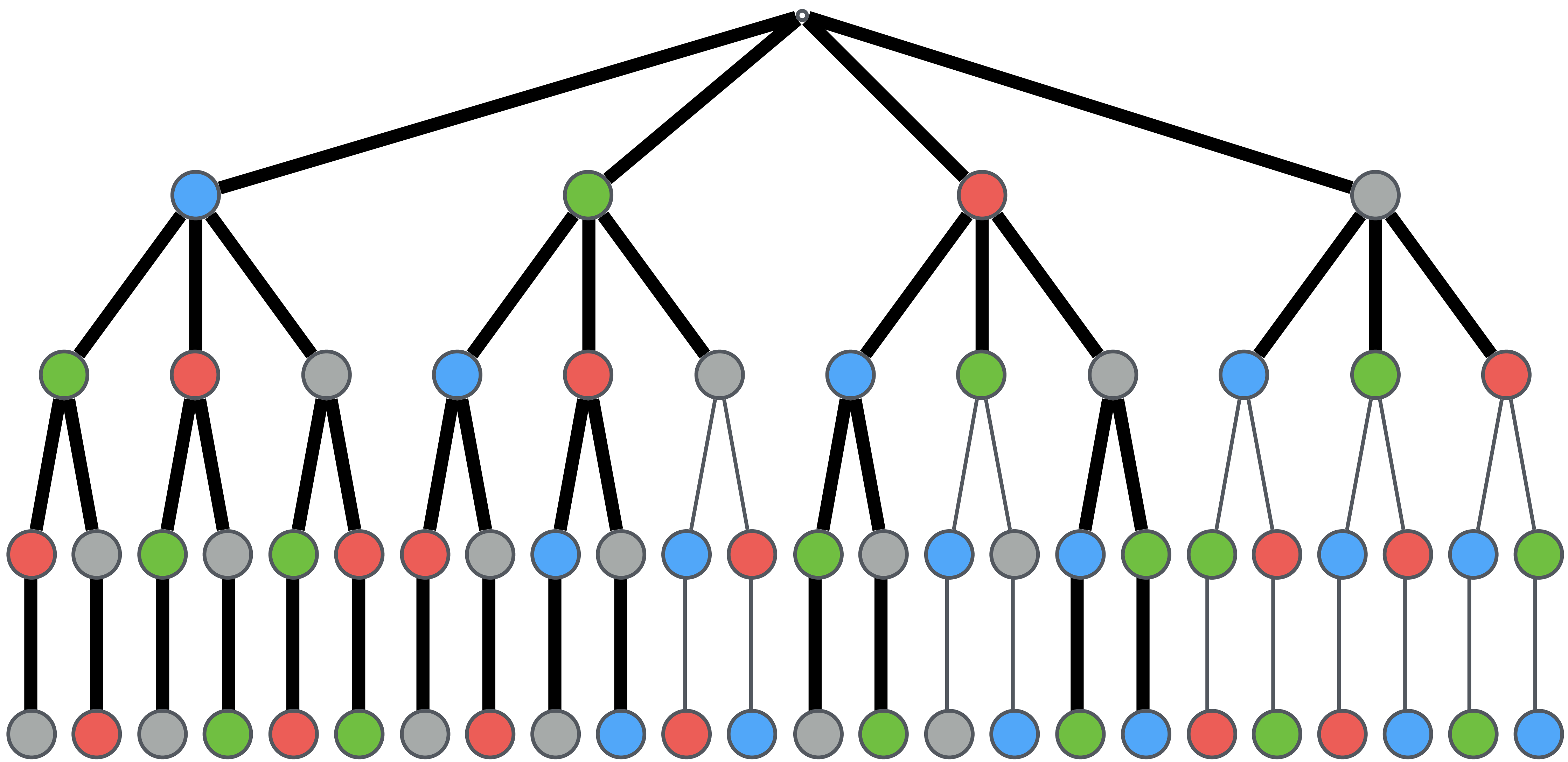
incrementally grow to prefixes of length 4



global optimization complete



summary of computations



computational gains depend on how quickly lower bounds become tight or exceed best known objective we can derive other bounds

- combine for more aggressive pruning

we can derive other bounds

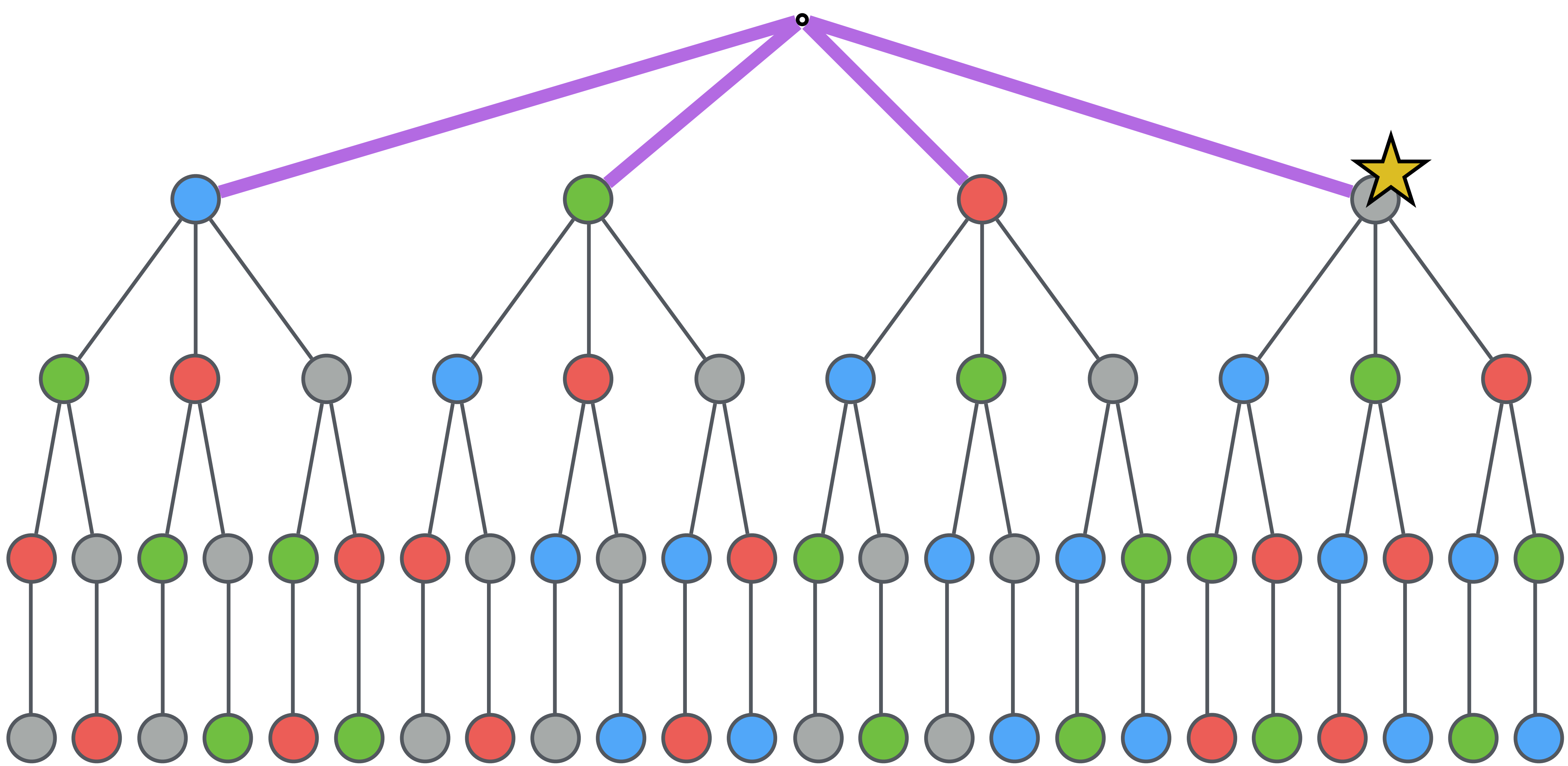
- combine for more aggressive pruning

- e.g., exploit structure based on permutations

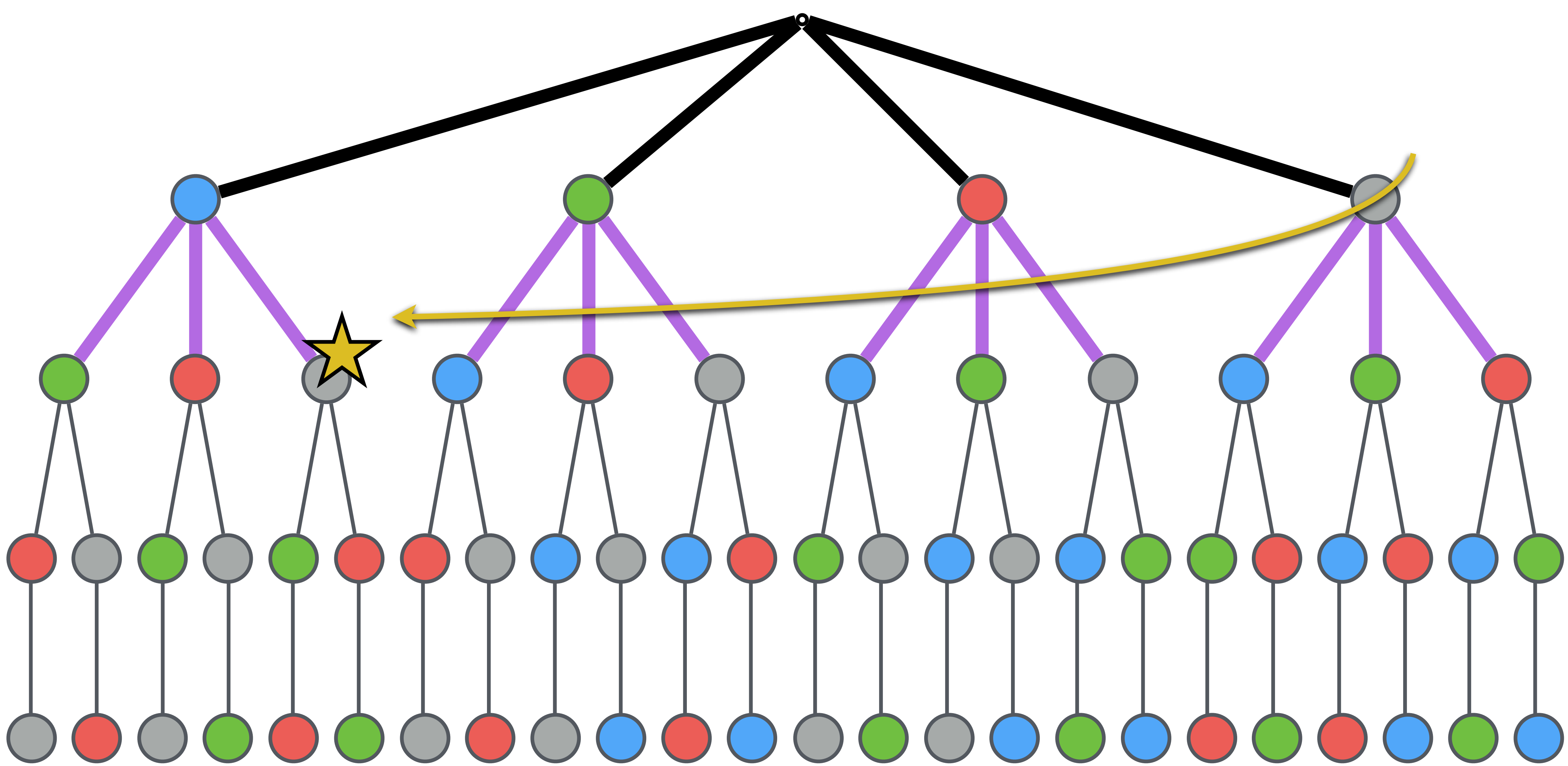
=> only extend best prefix within permutation group

- e.g., similar prefixes have similar subtrees

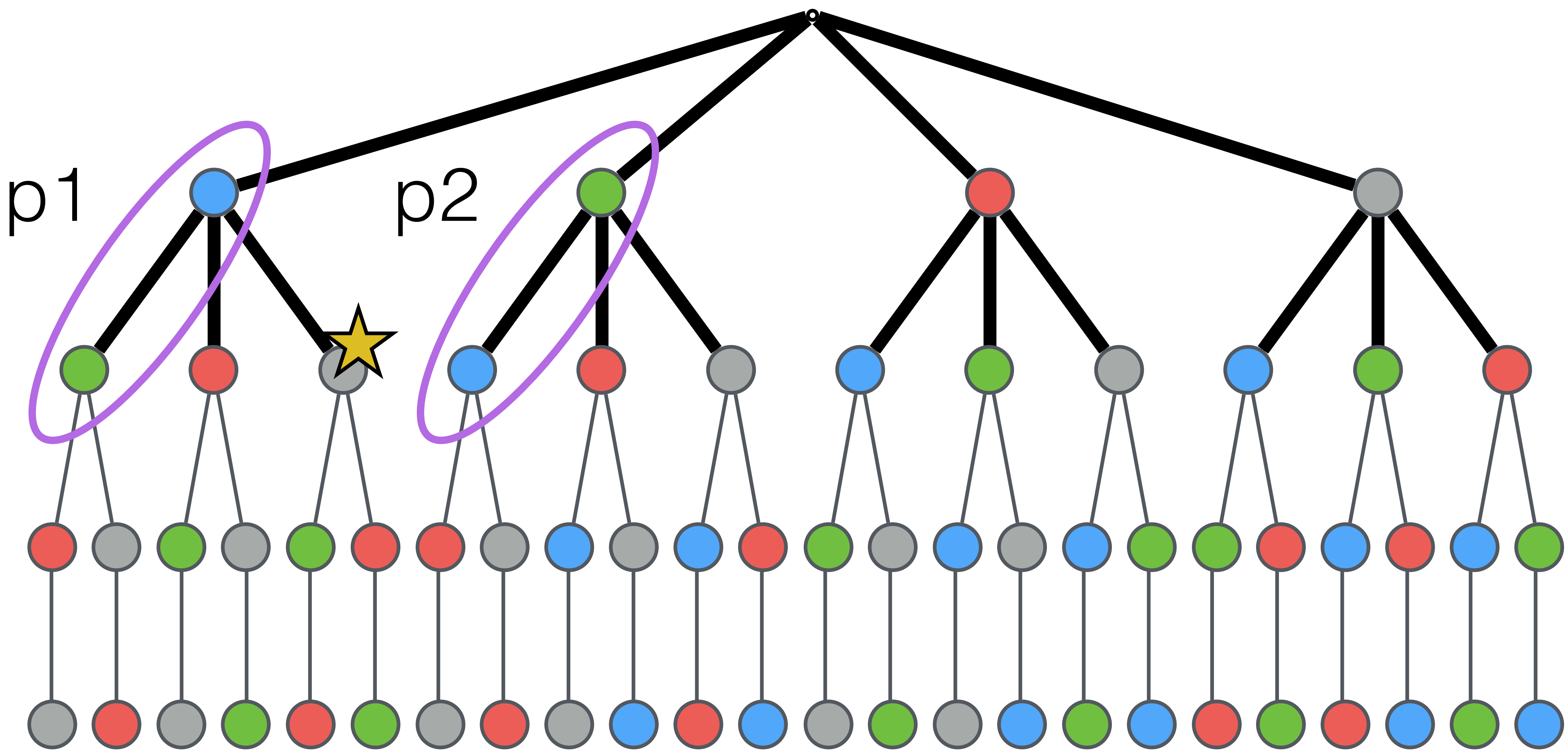
=> if one subtree is eliminated, don’t evaluate the other evaluate all prefixes of length 1



incrementally grow to all prefixes of length 2

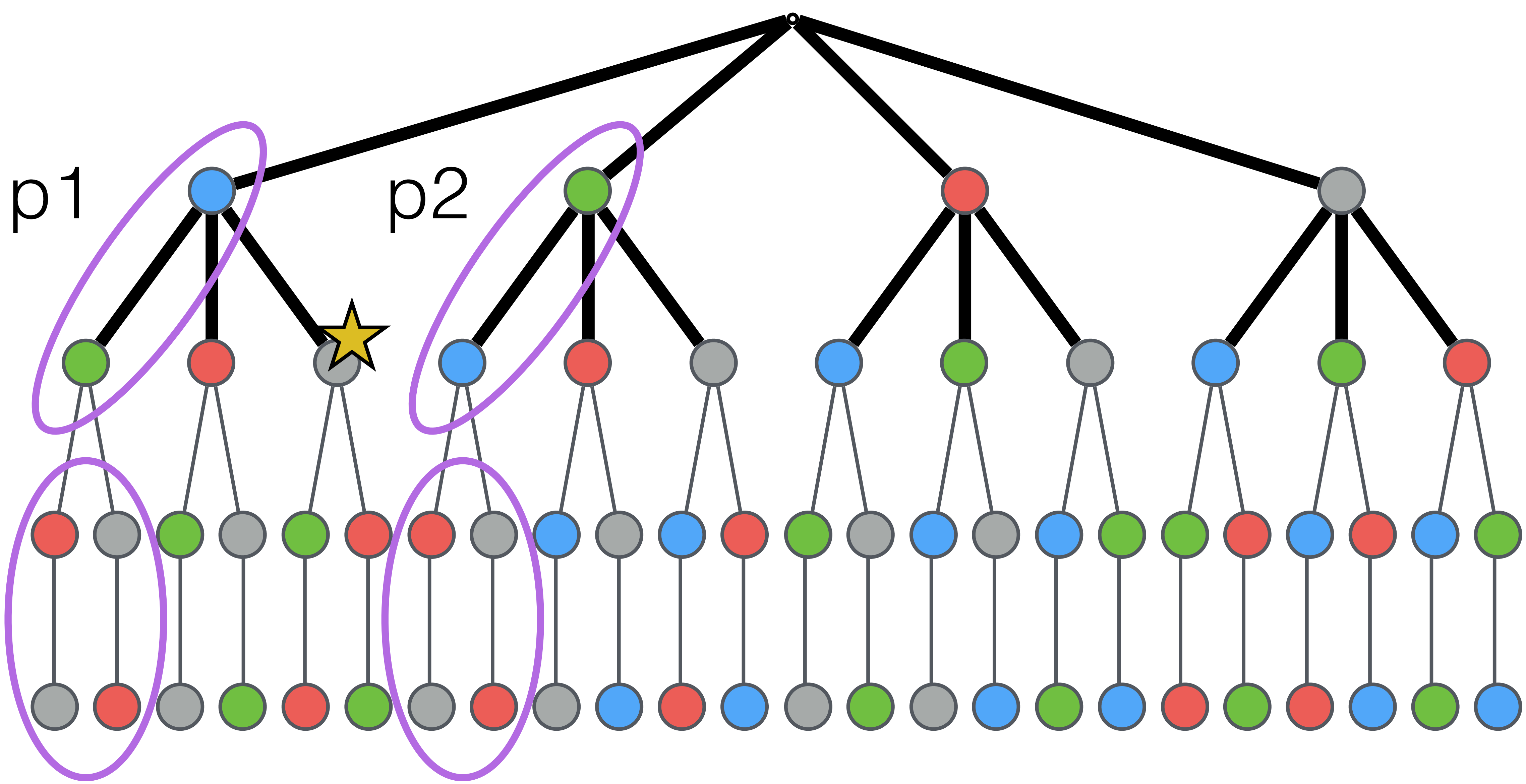


p1, p2 equivalent up to permutation

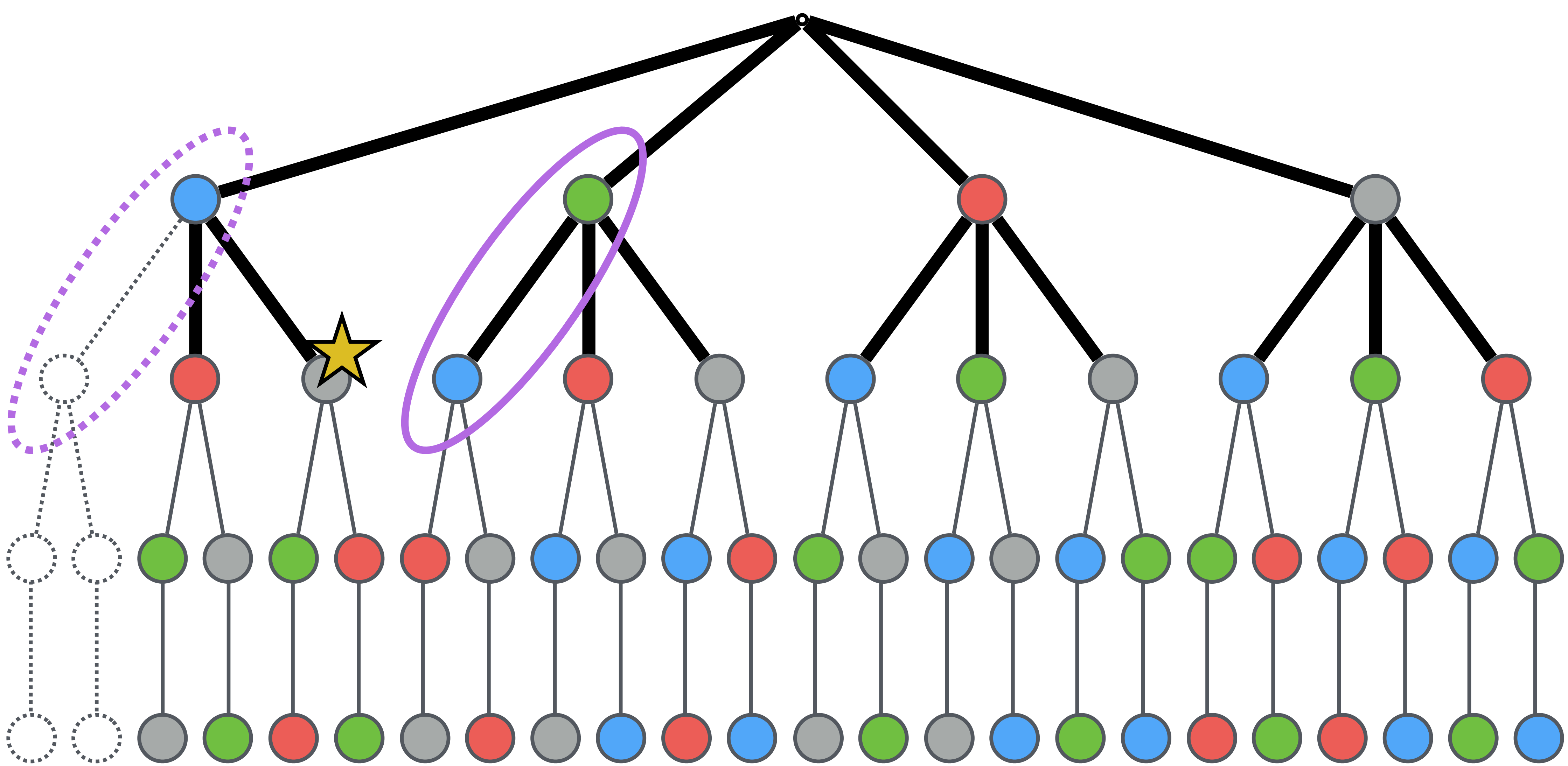


p1, p2 equivalent up to permutation

=> possible extensions have the same effect

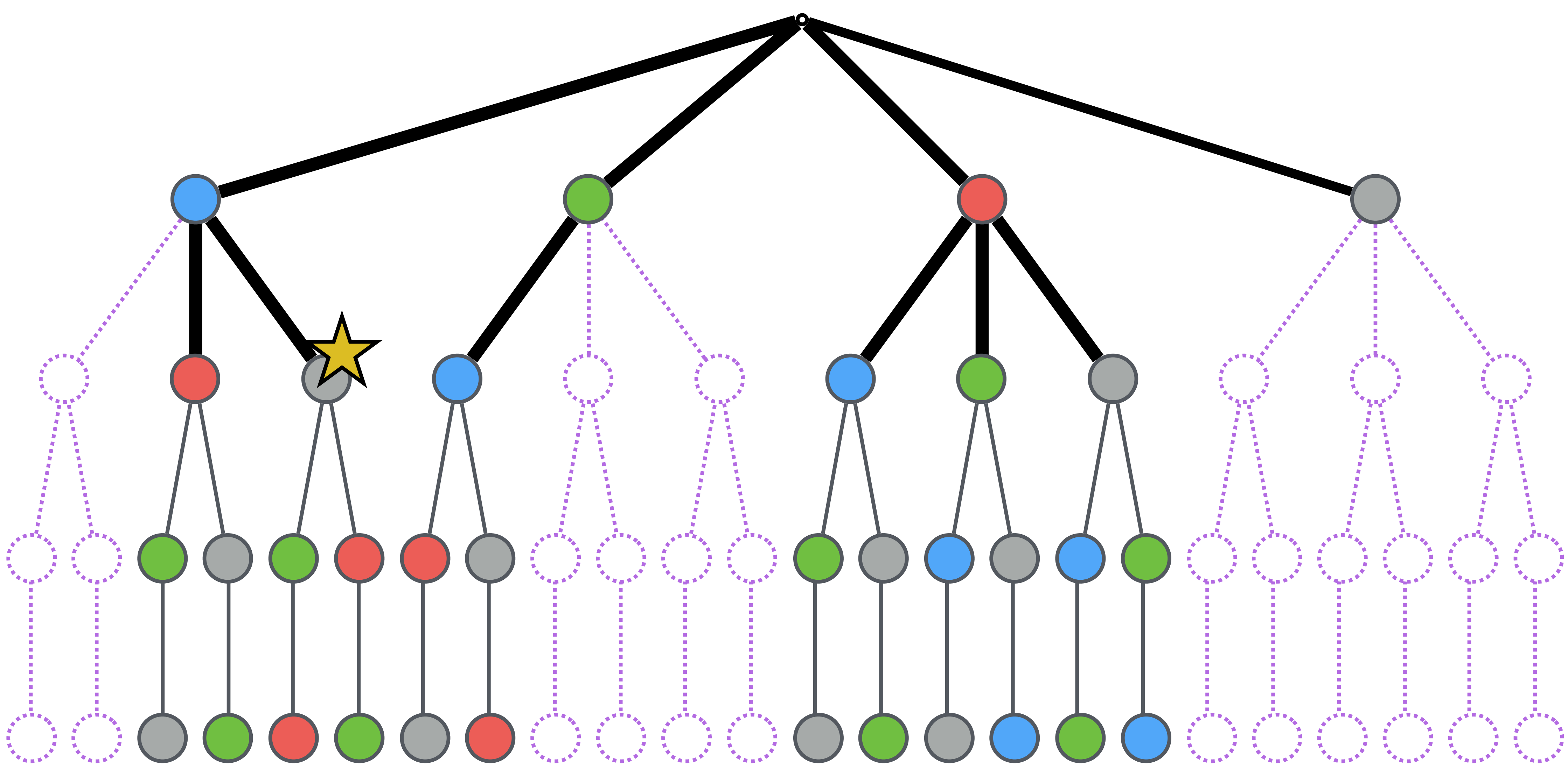


=> delete inferior permutation*

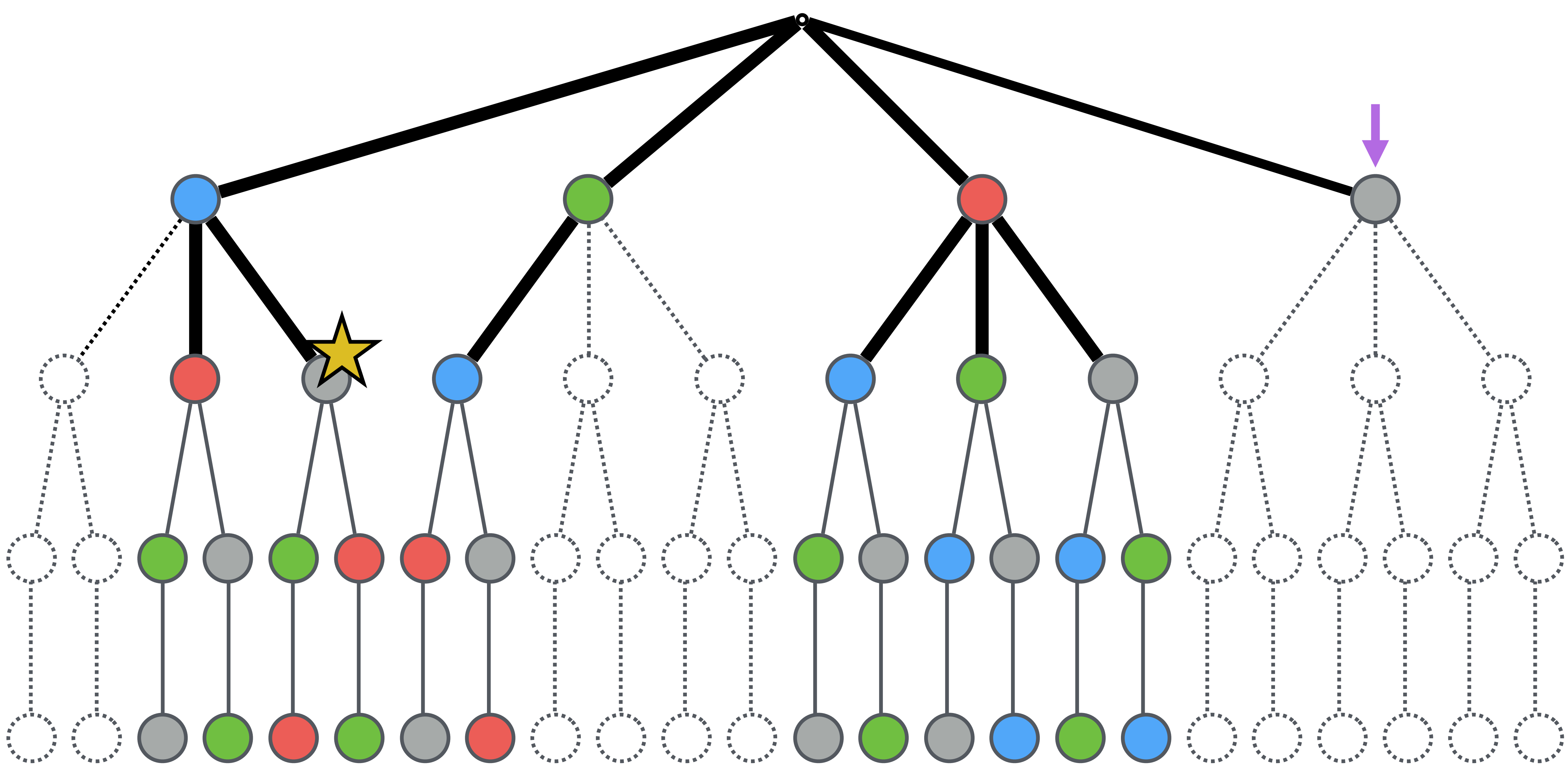


*requires permutation-aware cache queries delete all inferior permutations of length 2

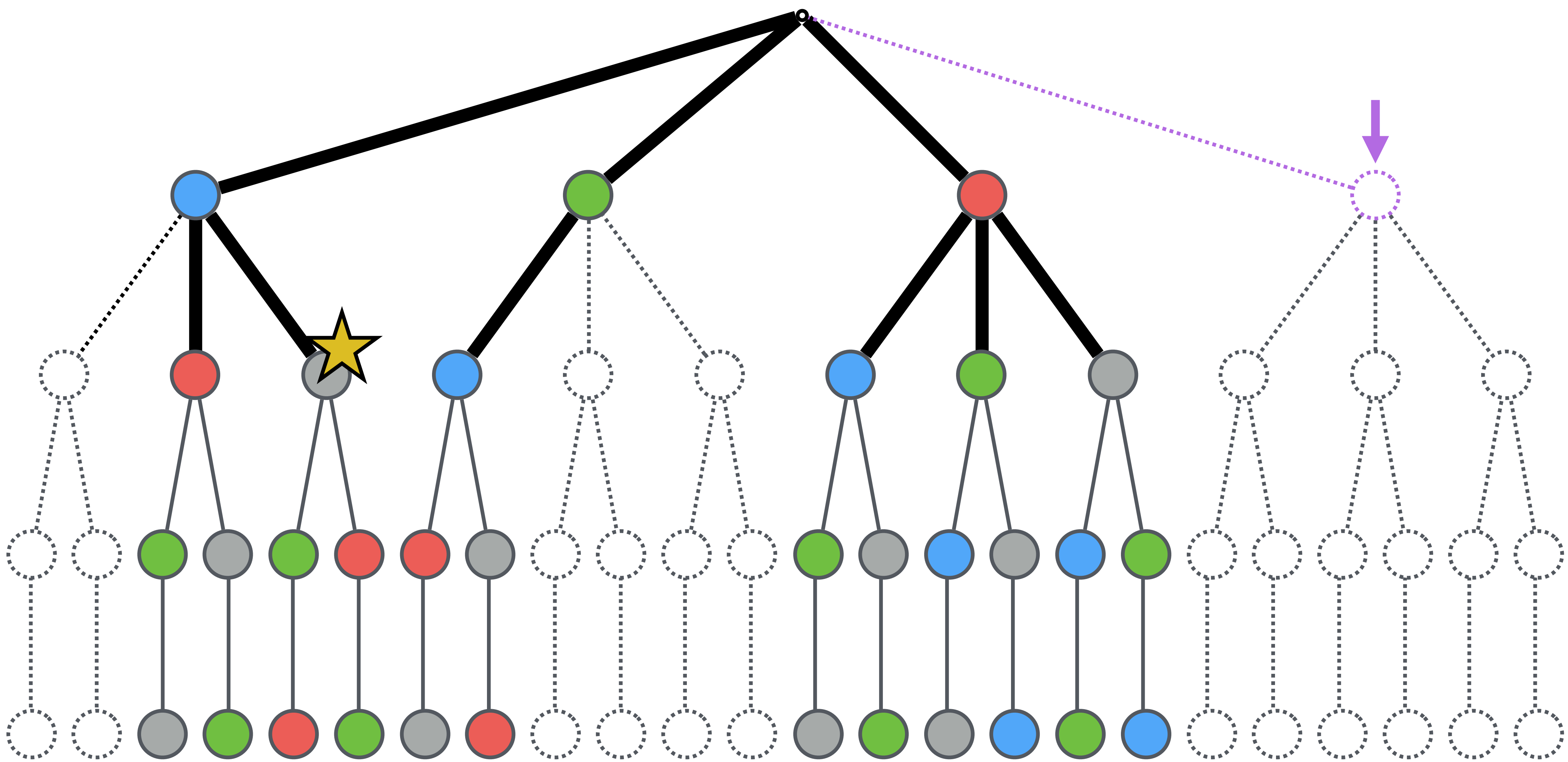
=> prunes remaining search space by 1/2



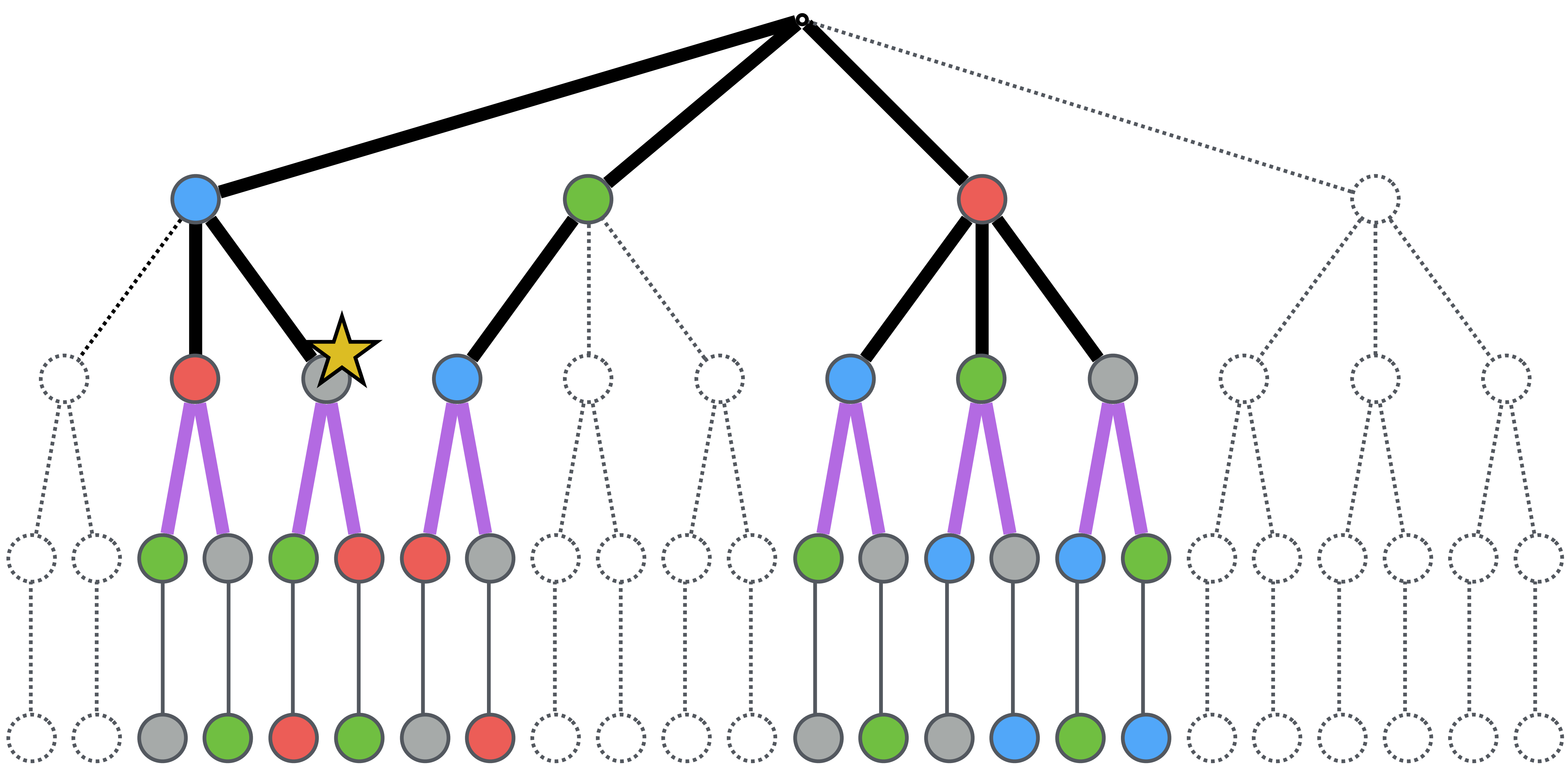
(optional: prune childless nodes)



(optional: prune childless nodes)

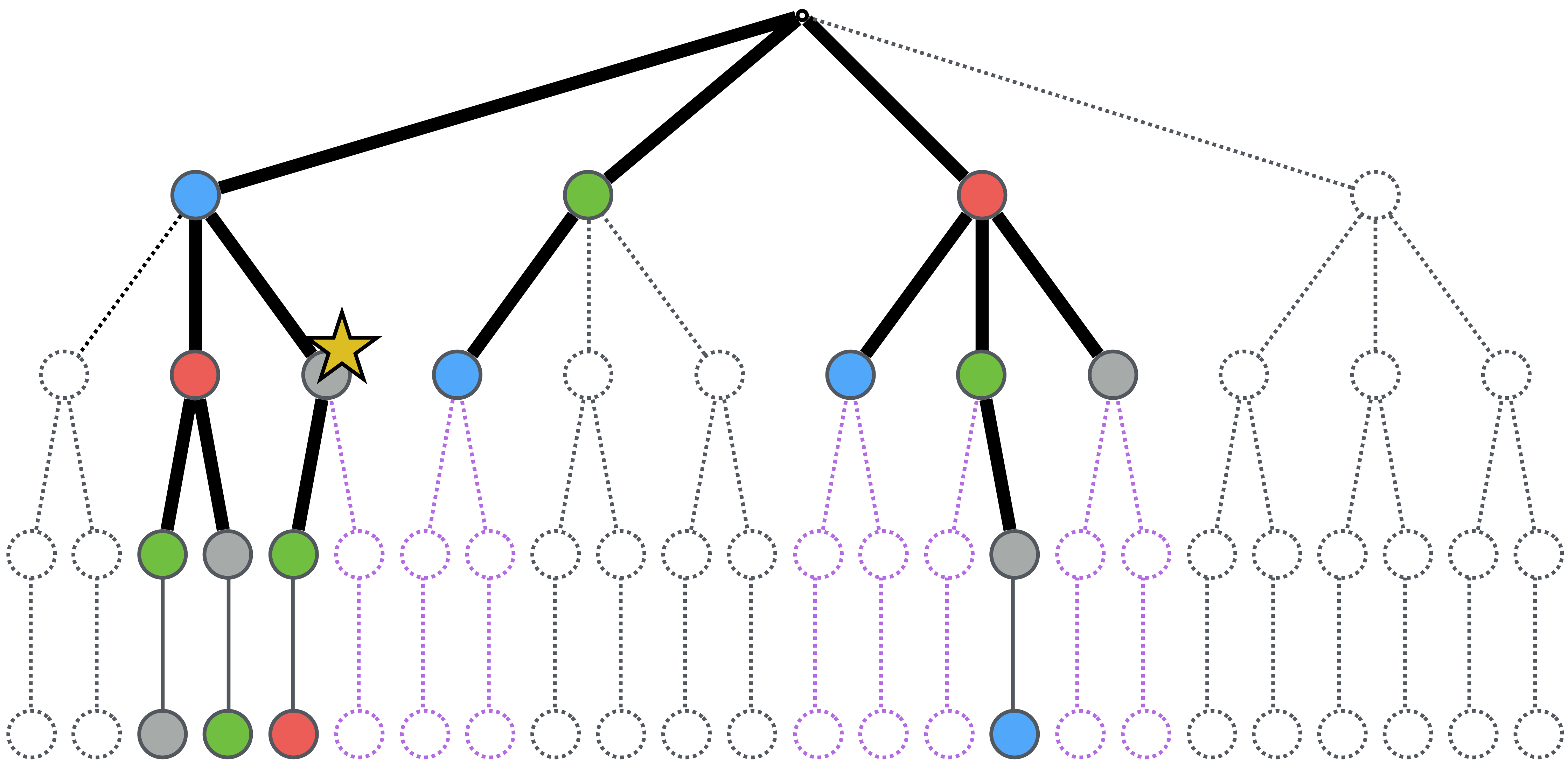


incrementally grow to prefixes of length 3

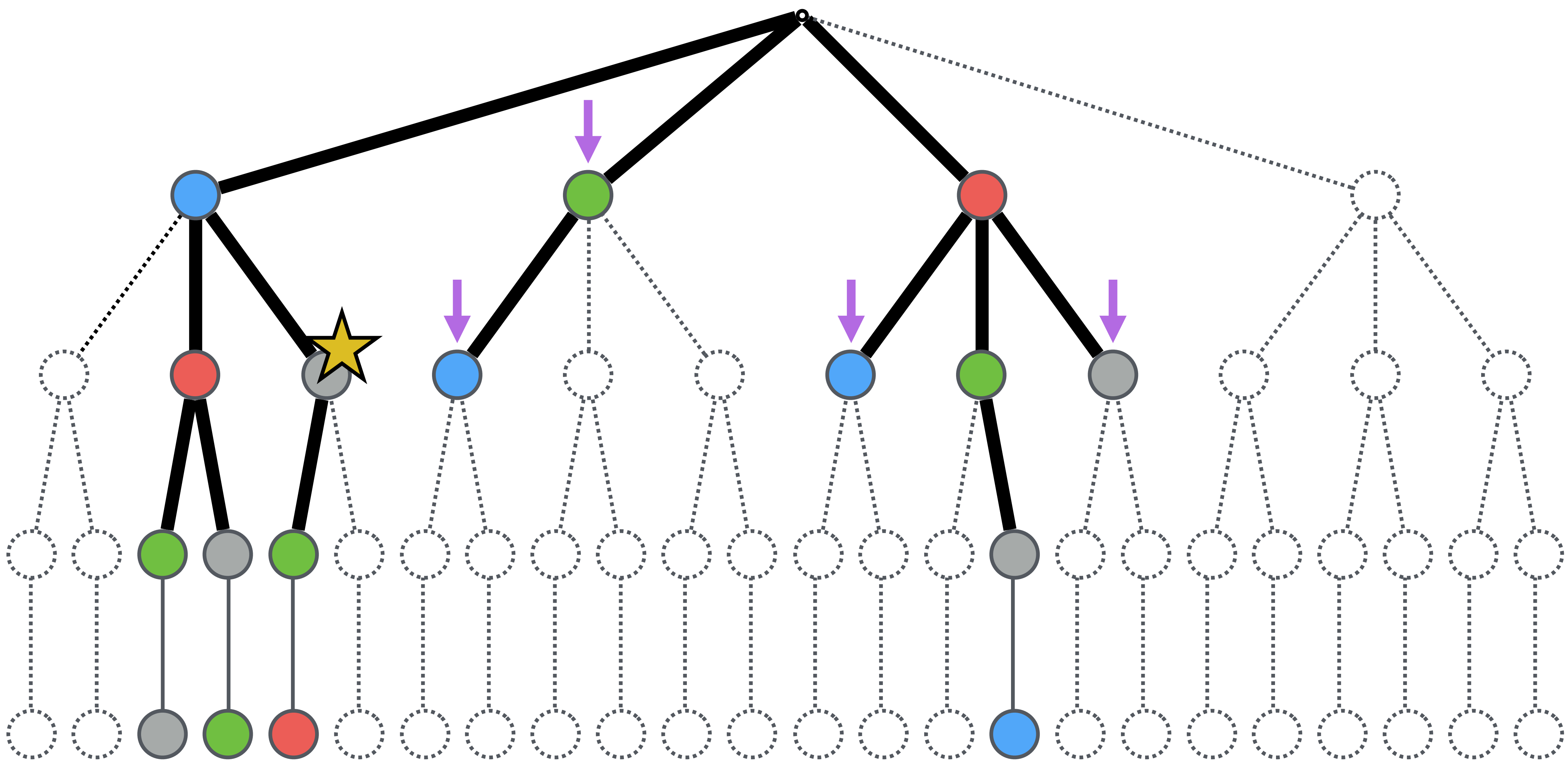


delete inferior permutations

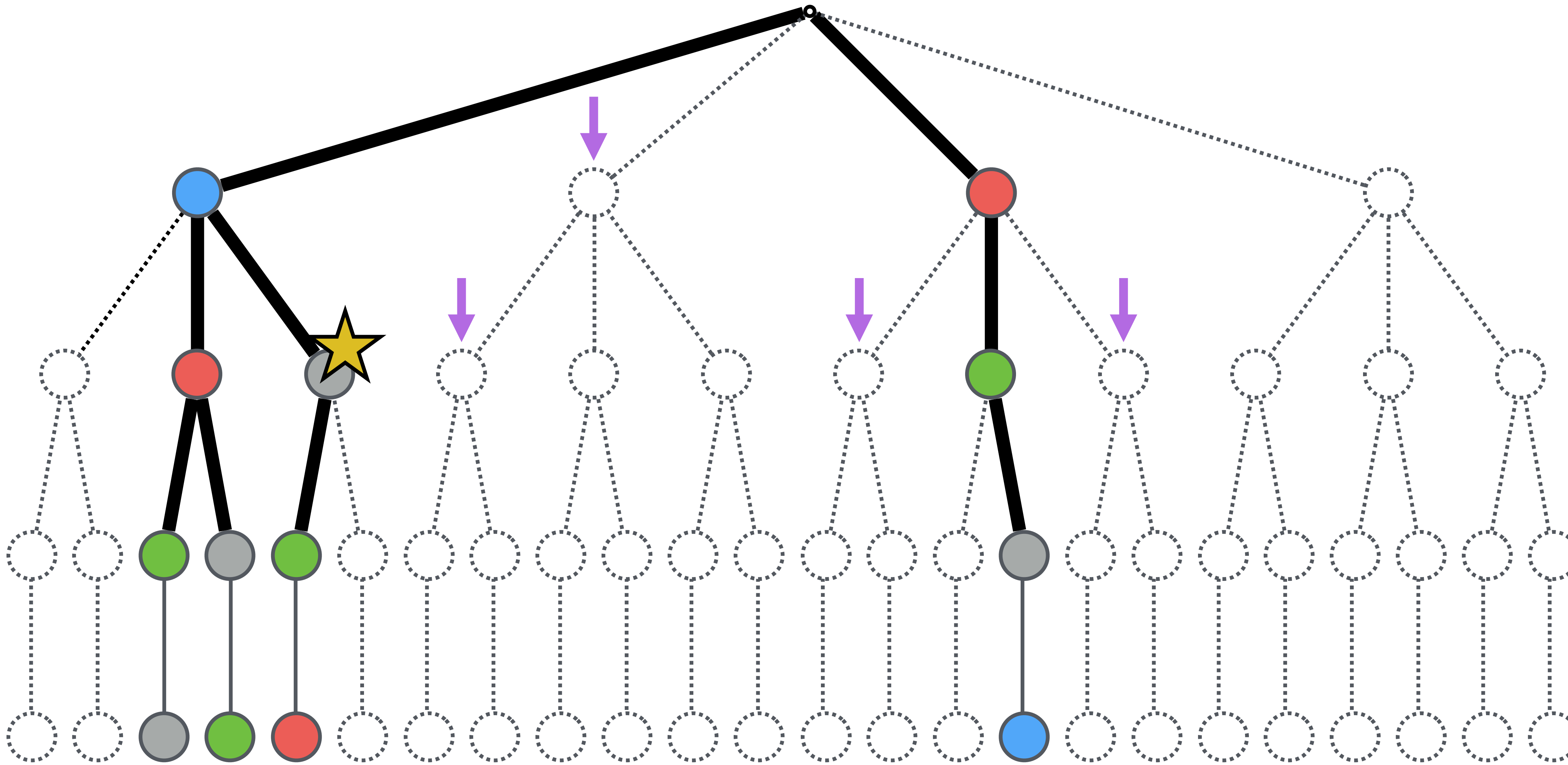
=> prunes remaining search space by 2/3



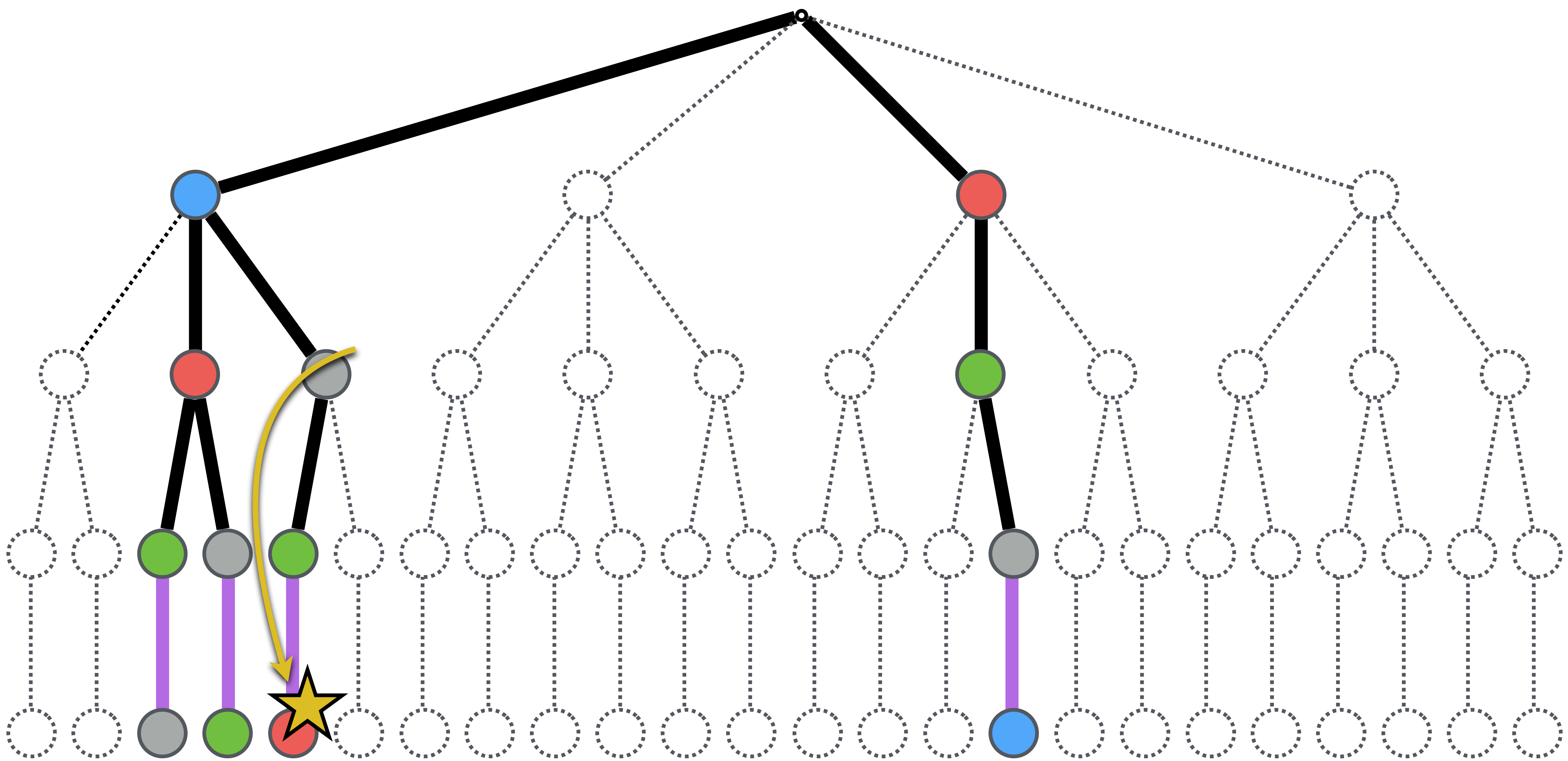
(optional: prune childless nodes)



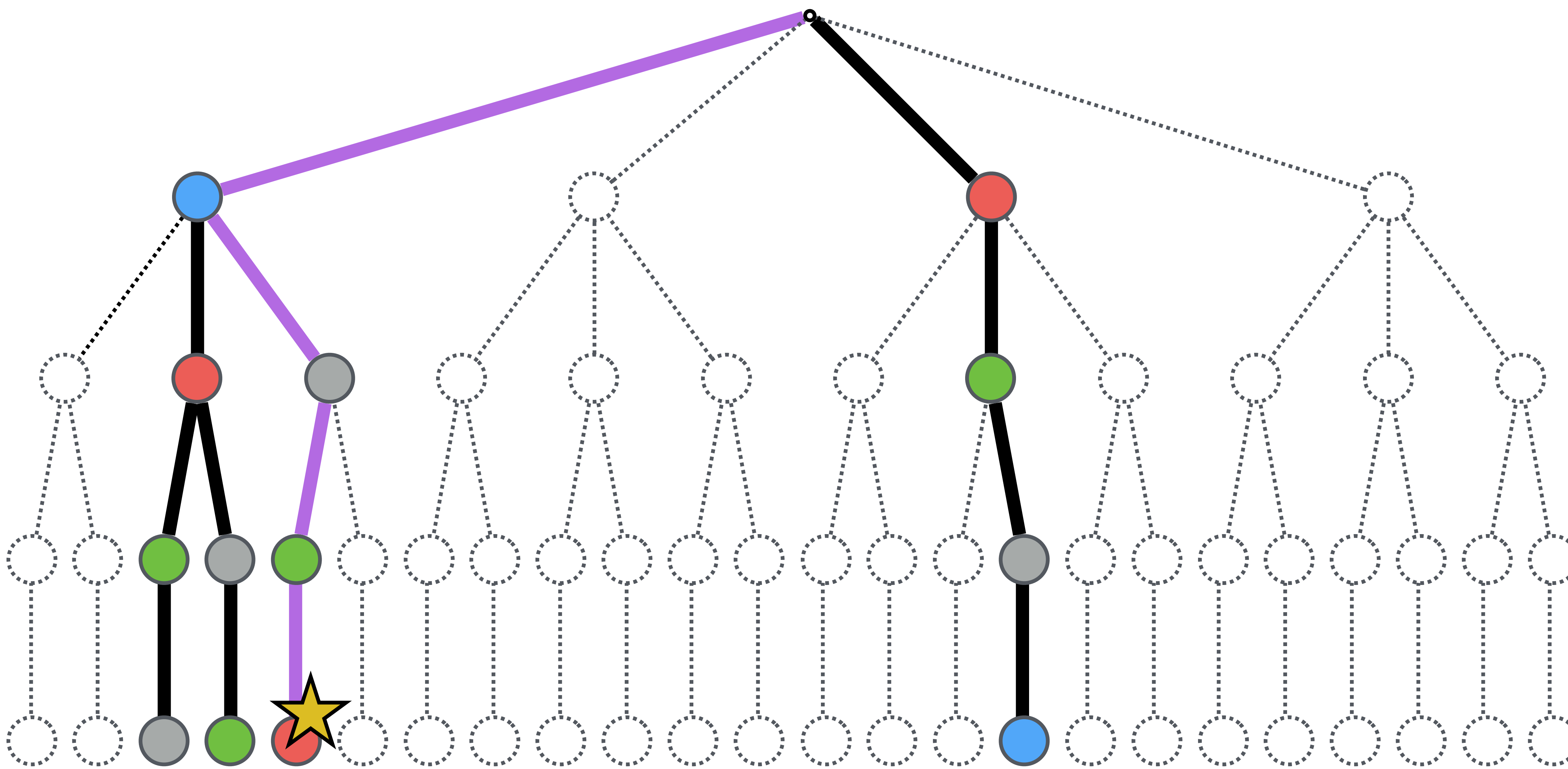
(optional: prune childless nodes)



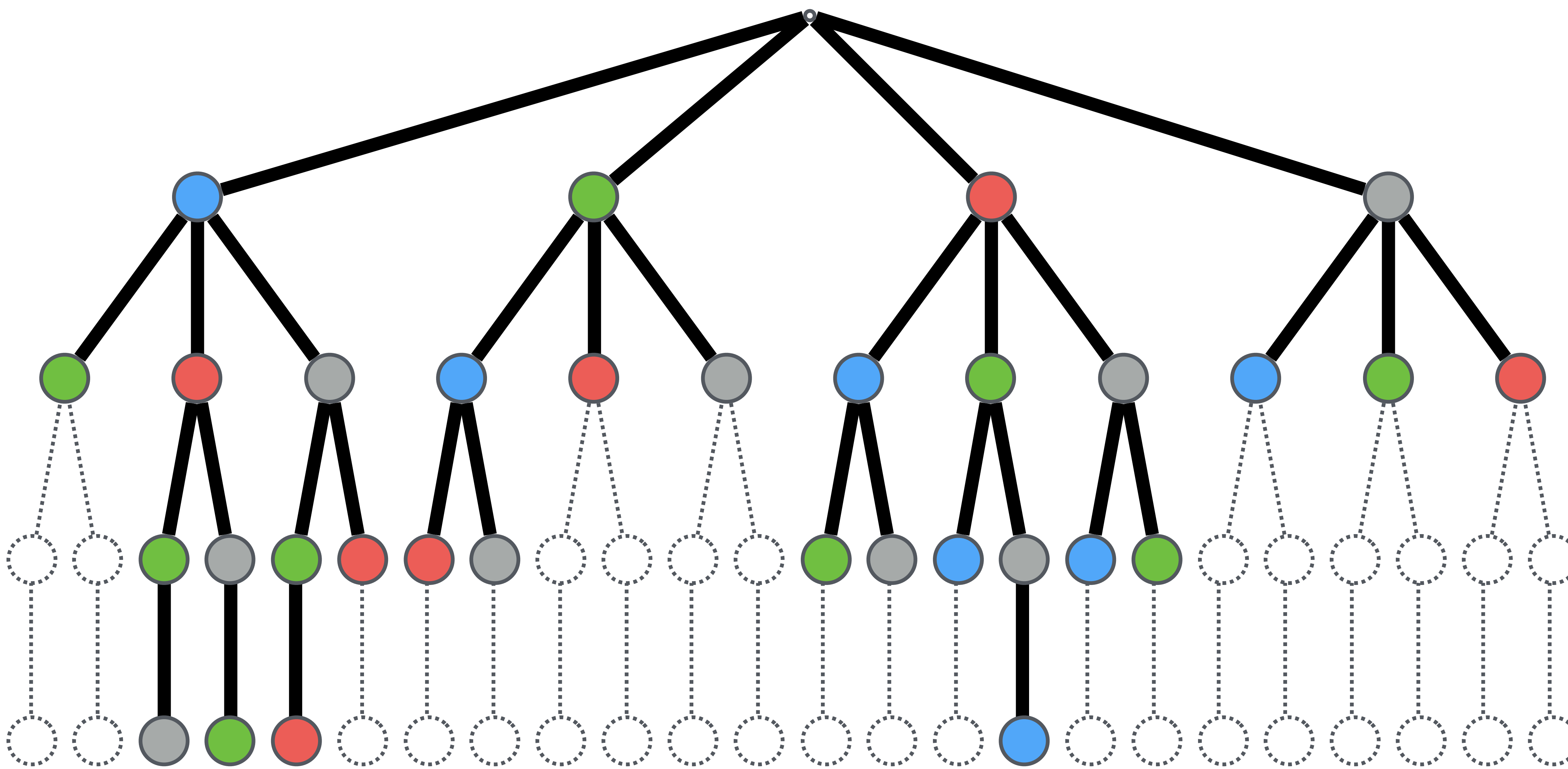
incrementally grow to prefixes of length 4



global optimization complete



summary of computations



$$\# \text{ states} = 4 + (4\text{x}3) + (4\text{x}3\text{x}2) + (4\text{x}3\text{x}2\text{x}1) = 64$$

$$\# \text{ evaluated states} = 4 + (4\text{x}3) + (4\text{x}3) + 4 = 32$$

data structures

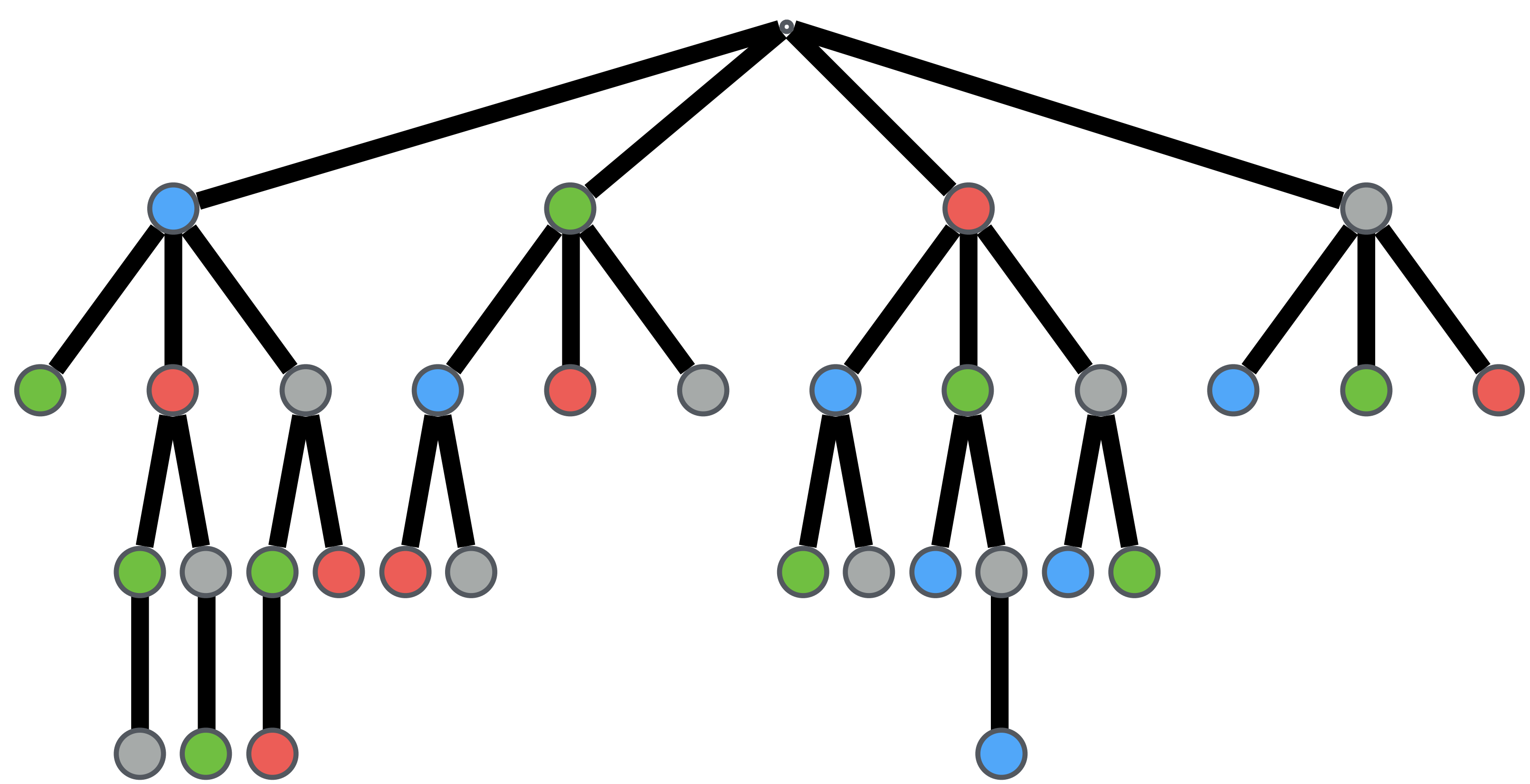
- cache
- queue
- map

cache

- a trie efficiently represents evaluated prefixes

- supports incremental computation

- nodes store computed lower bounds



queue

- can support different scheduling policies
- indexes the trie’s leaves (next computations)
- FIFO/LIFO for BFS/DFS
- priority queue (order by lower bound; “curiosity”)

garbage collection

- triggered when best known objective decreases
- traverse trie, call **delete_subtree**(node) when lower bound(node) < best known objective
- **delete_subtree**(node)
    - delete non-leaf (done) nodes
    - mark leaf nodes because they're in the queue
- lazily gc marked nodes when popped from queue

permutation map

- supports permutation-aware gc
- keys represent sets of prefixes equivalent up to a permutation
- values stores index of best known (only) such prefix that has been evaluated, plus its lower bound

$$\{\ (1, 2, 3) : (\ (3, 1, 2), 0.05\ )\ \}$$

- supports our permutation lower bound
- note that (3, 1, 2) could be deleted from the cache

implementation details

- Python useful for prototyping but ultimately slow, not memory-efficient, limited options for parallelism and custom data structures
- initial rewrite in C++ with custom trie about 10x faster, 100x more cache elements
- templates enable efficient modular exploration of many variants depending on different data structures
- leverages library for fast bit vector operations based on GMP

ProPublica COMPAS recidivism dataset

Northpointe’s **C**orrectional **O**ffender **M**anagement **P**rofiling for **A**lternative **S**anctions (COMPAS) is a controversial, proprietary black box algorithm

"In 2009, Brennan [et al] published a validation study that found that Northpointe’s risk of recidivism score had an accuracy rate of [68%] in a sample of 2,328 people. Their study also found that the score was slightly less predictive for black men than white men — [67% vs. 69%].”

open problem: can human-interpretable machine learning transparently achieve competitive results?

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

ProPublica COMPAS recidivism dataset

- 7214 individuals (52% recidivate within 2 years)

  - **sex** (male, female)
  - **age** (18-20, 21-25, 26-30, 31-40, 41-50, >50)
  - **race** (African American, Caucasian, Asian, Hispanic, Native American, Other)
  - **# juvenile felonies** (0, >0)
  - **# juvenile misdemeanors** (0, >=1)
  - **# juvenile crimes** (0, >=1)
  - **# priors** (0, 1, 2-3, >3)
  - **current charge** (misdemeanor, felony)

- rule mining yields ~157 rules
- 1-2 clauses with min (max) support = 0.01 (0.99)

https://github.com/propublica/compas-analysis

example rule lists (with certificates of optimality)

predict whether individual recidivates within 2 years

if (male **and** juvenile crimes > 0) **then** (yes)
else if (juvenile felonies = 0 **and** priors > 3) **then** (yes)
else (no)

if (age = 18-20) **then** (yes)
else if (male **and** age = 21-25) **then** (yes)
else if (age = 26-30 **and** priors = 2-3) **then** (yes)
else if (priors > 3) **then** (yes)
else (no)

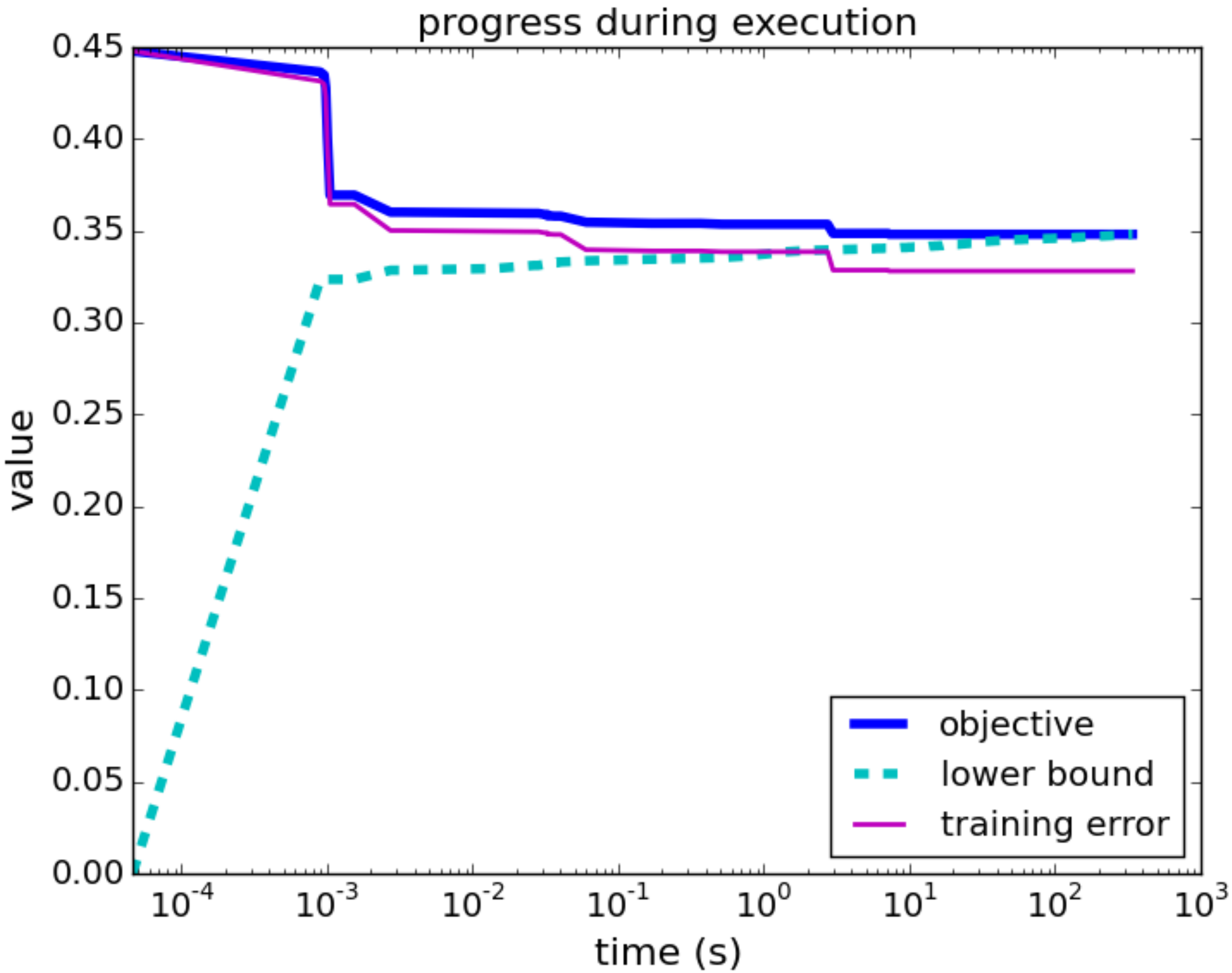
regularization (c) = 0.005
test accuracy = 0.675 +/- 0.020 (10-fold cross-validation)

human-interpretable, transparent, competitive

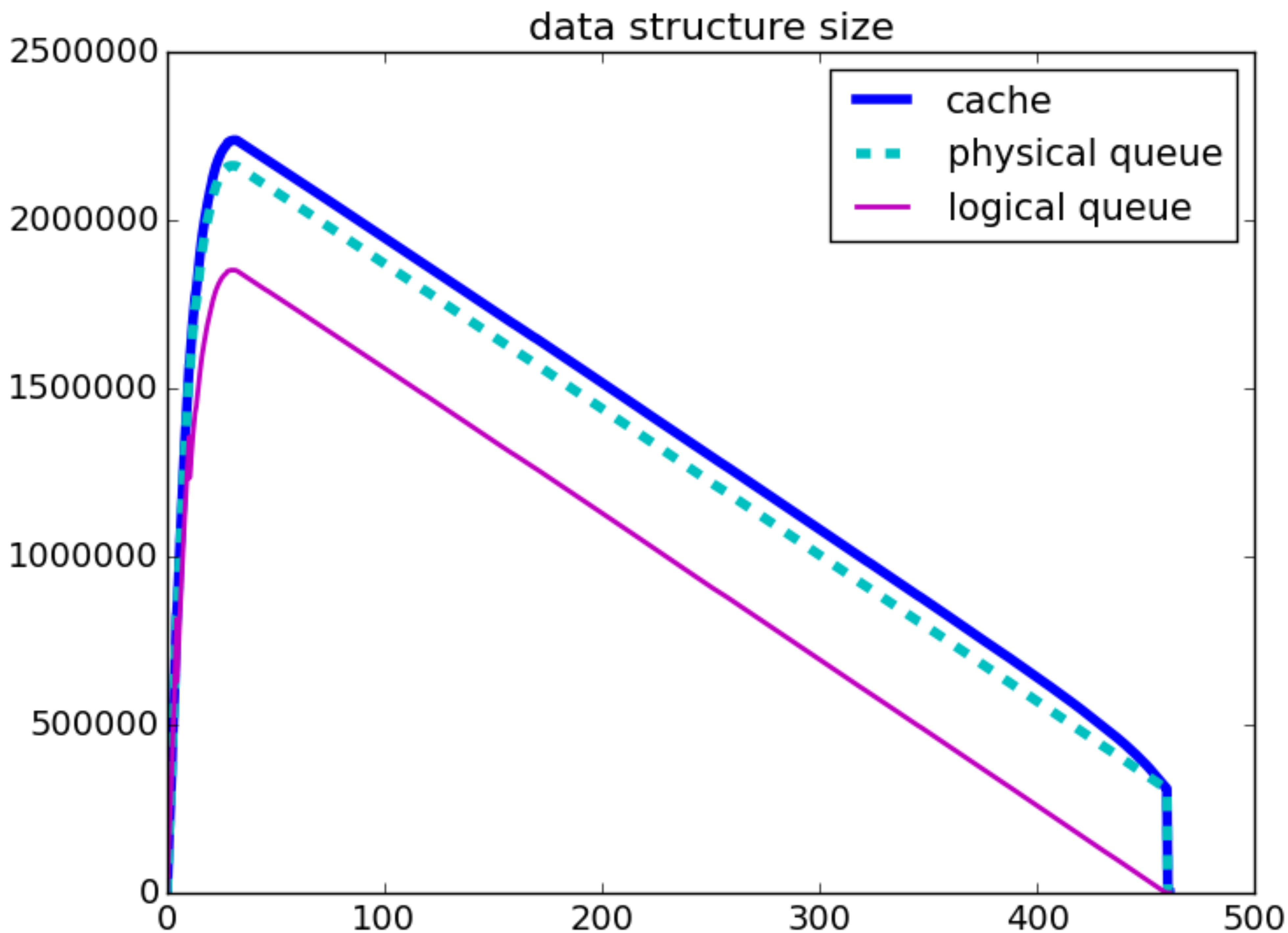
if (age = 18-20) **then** (yes)
else if (male **and** age = 21-25) **then** (yes)
else if (age = 26-30 **and** priors = 2-3) **then** (yes)
else if (priors > 3) **then** (yes)
else (no)

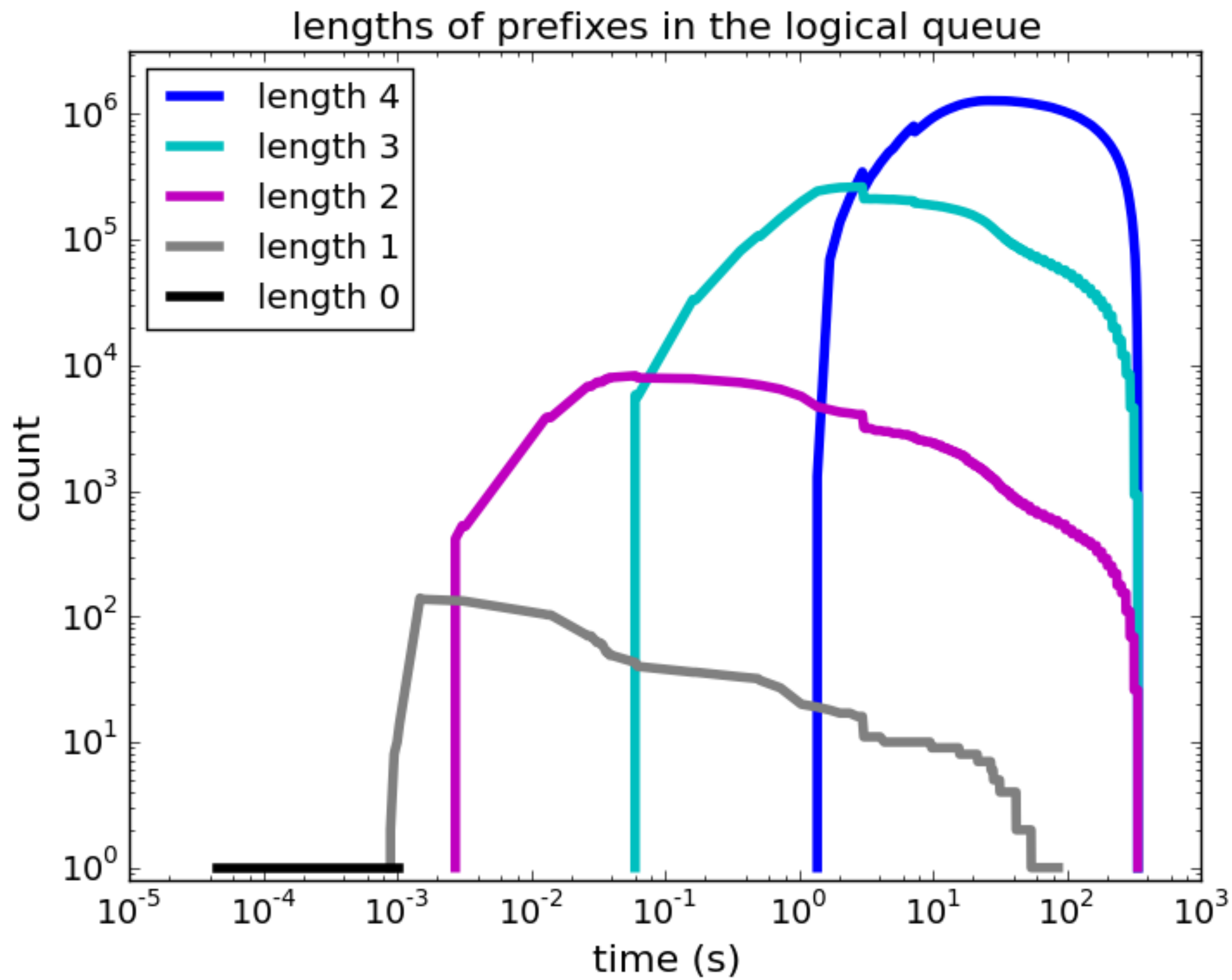
"Brennan said it is difficult to construct a score that doesn't include items that can be correlated with race — such as poverty, joblessness and social marginalization. 'If those are omitted from your risk assessment, accuracy goes down,' he said."

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

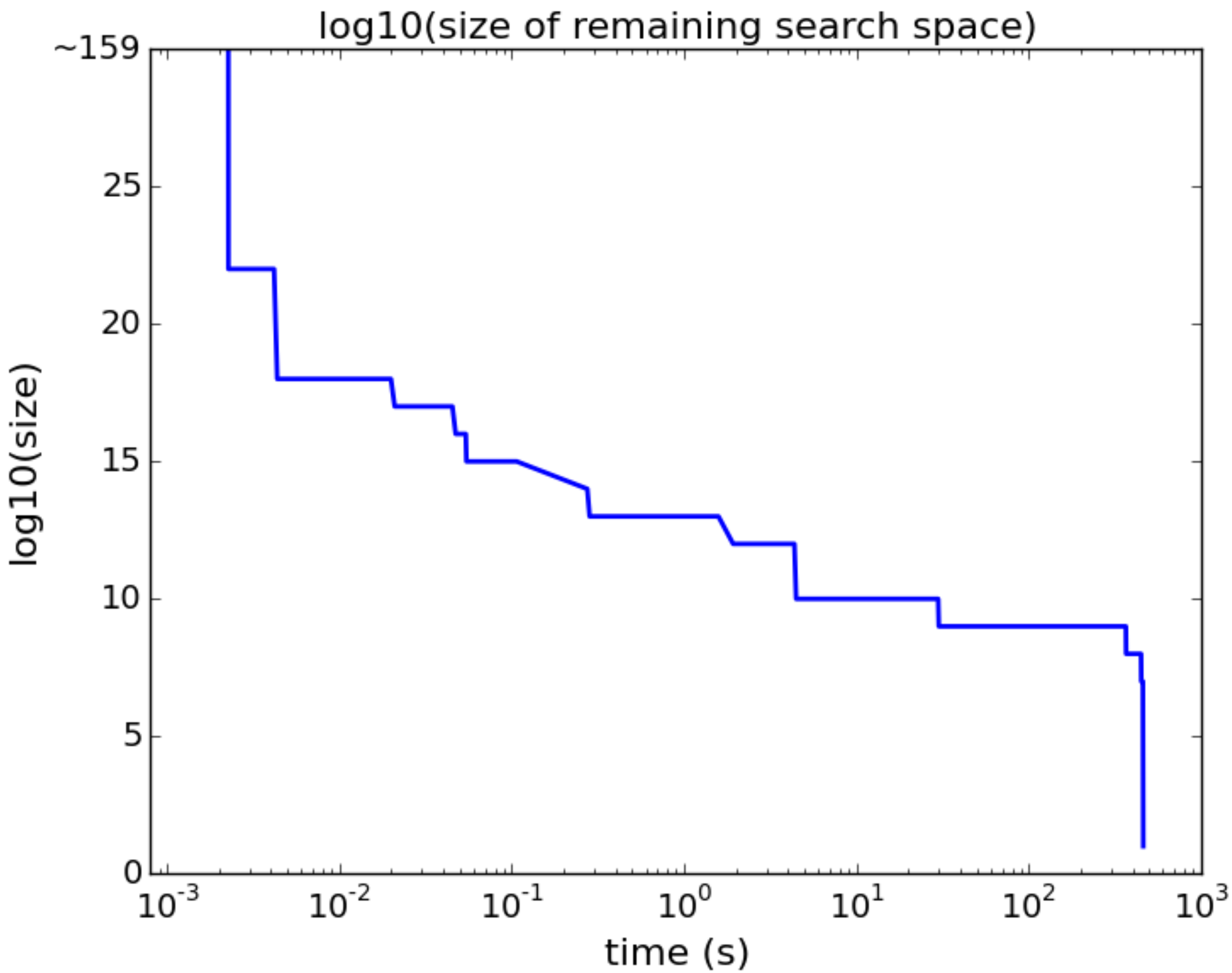


progress during execution
value
time (s)
objective
lower bound
training error

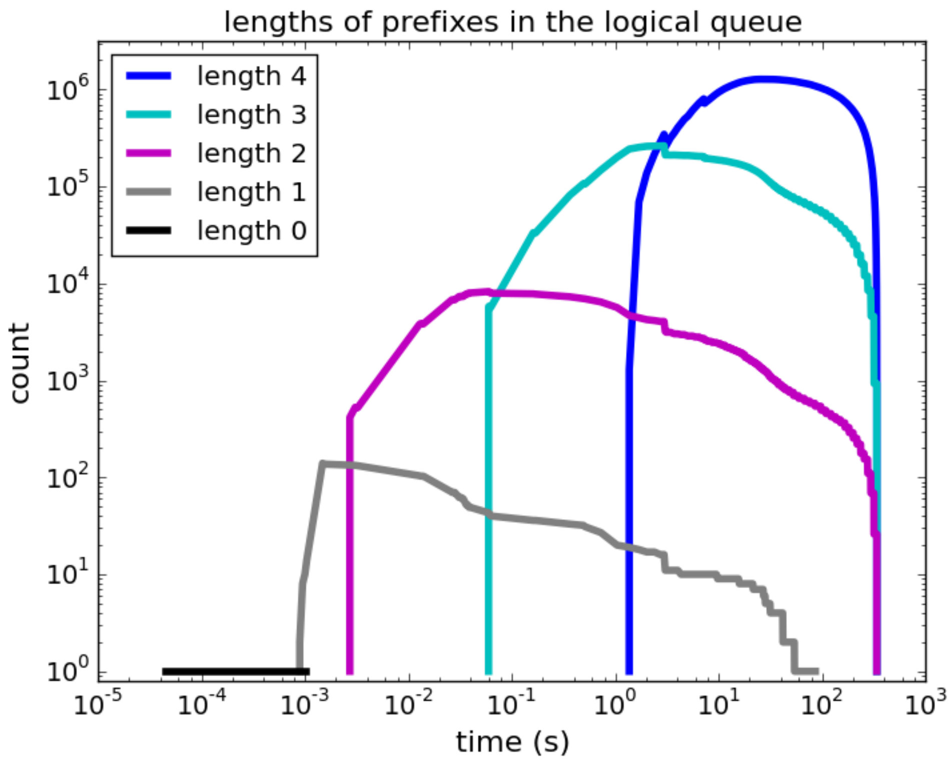

data structure size
cache
physical queue
logical queue
2500000
2000000
1500000
1000000
500000
0
0
100
200
300
400
500

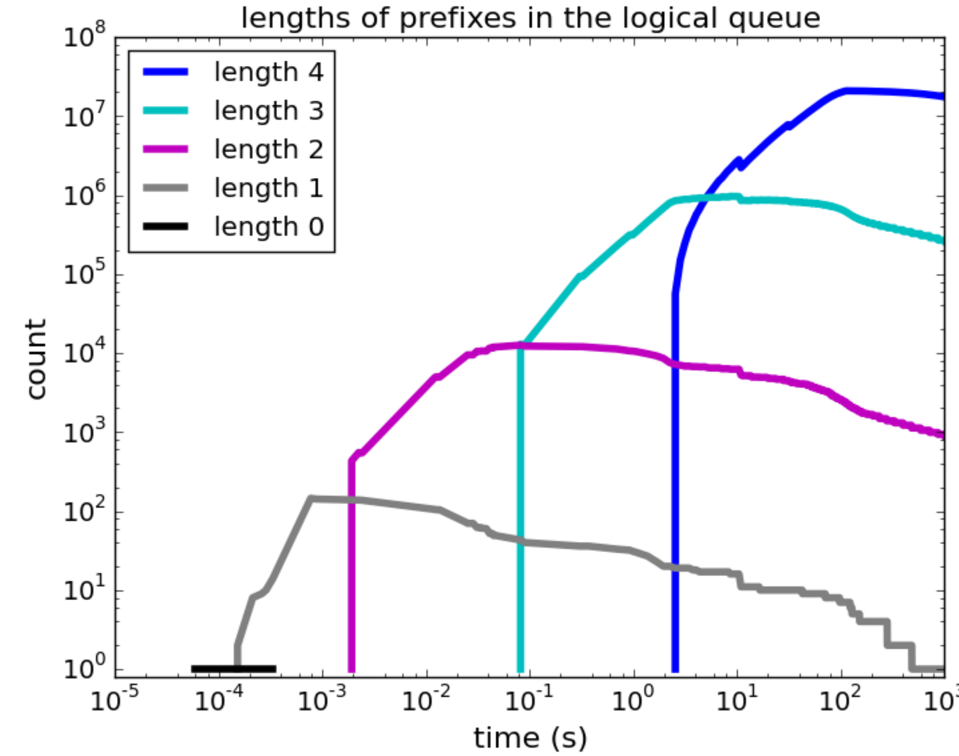


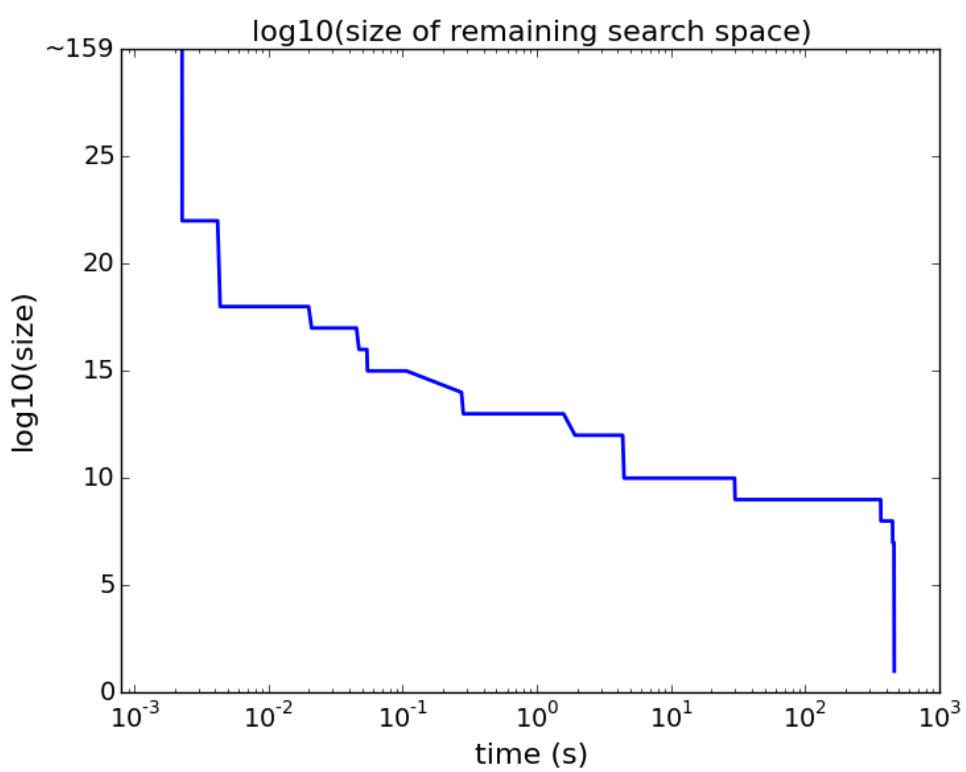
lengths of prefixes in the logical queue
length 4
length 3
length 2
length 1
length 0
count
time (s)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
10^1
10^2
10^3

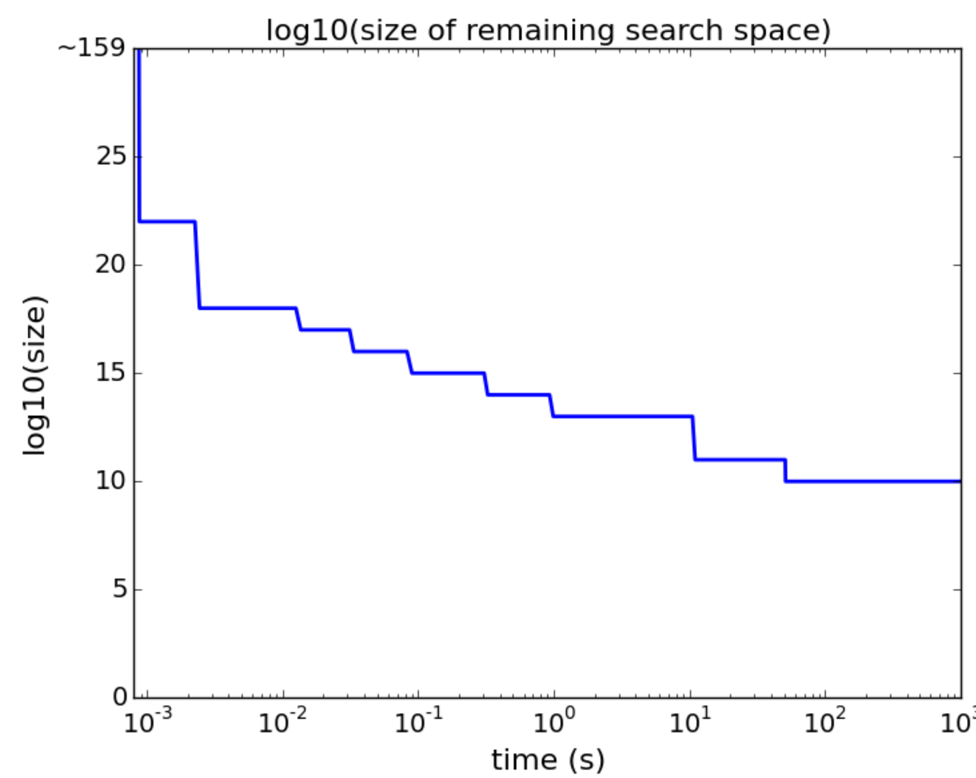

log10(size of remaining search space)
~159
25
20
15
10
5
0
log10(size)
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
time (s)

significantly slower without permutation map

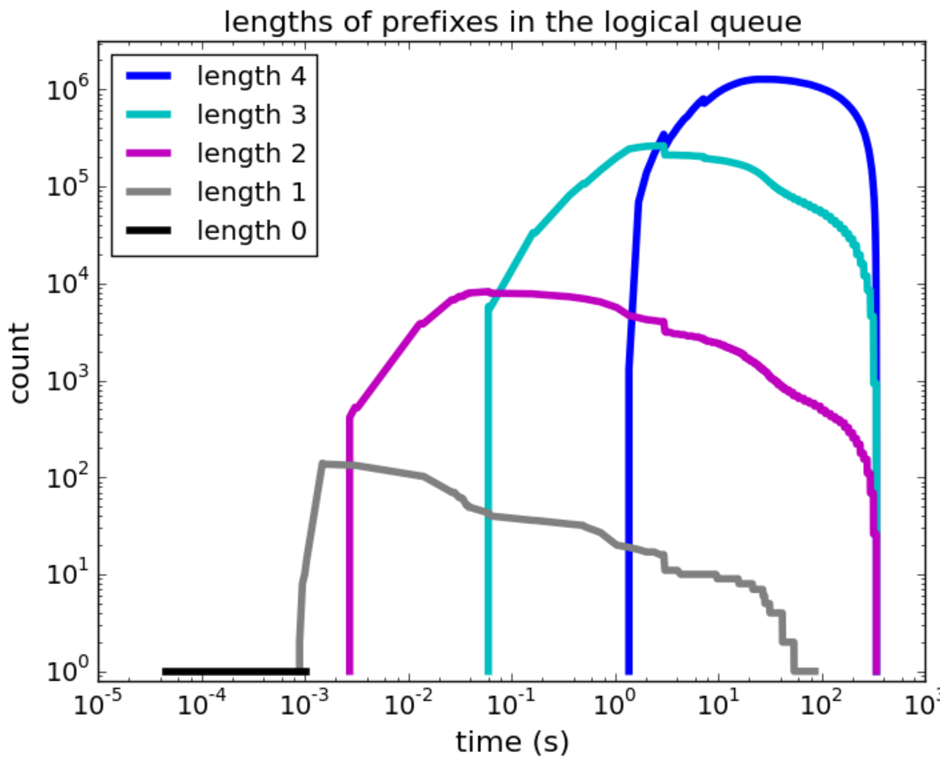


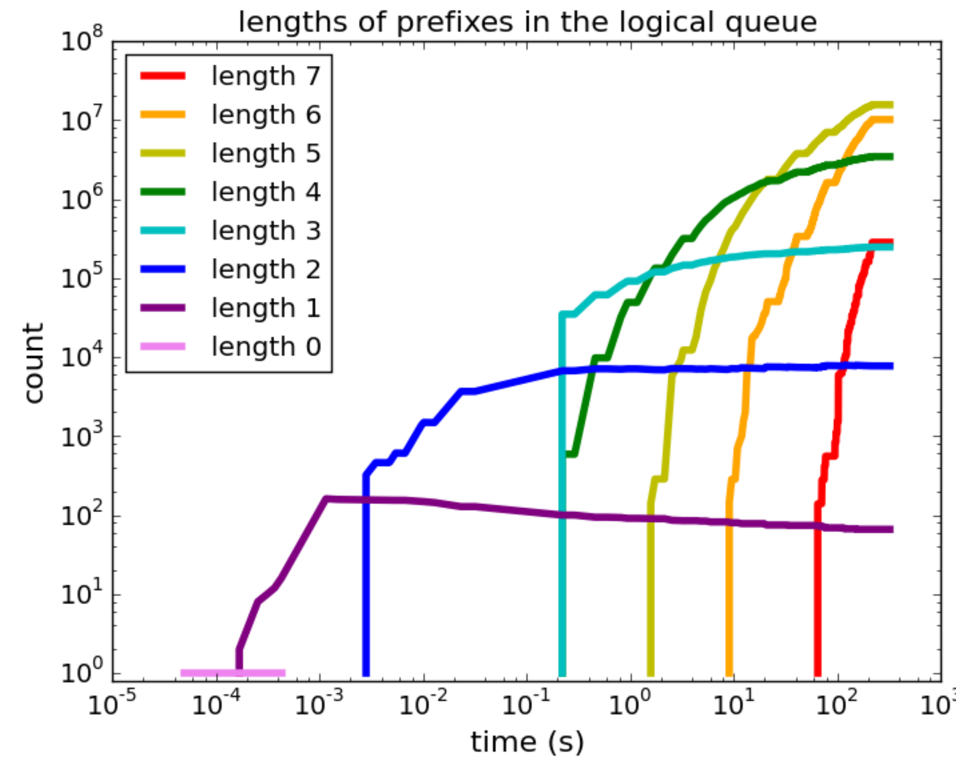


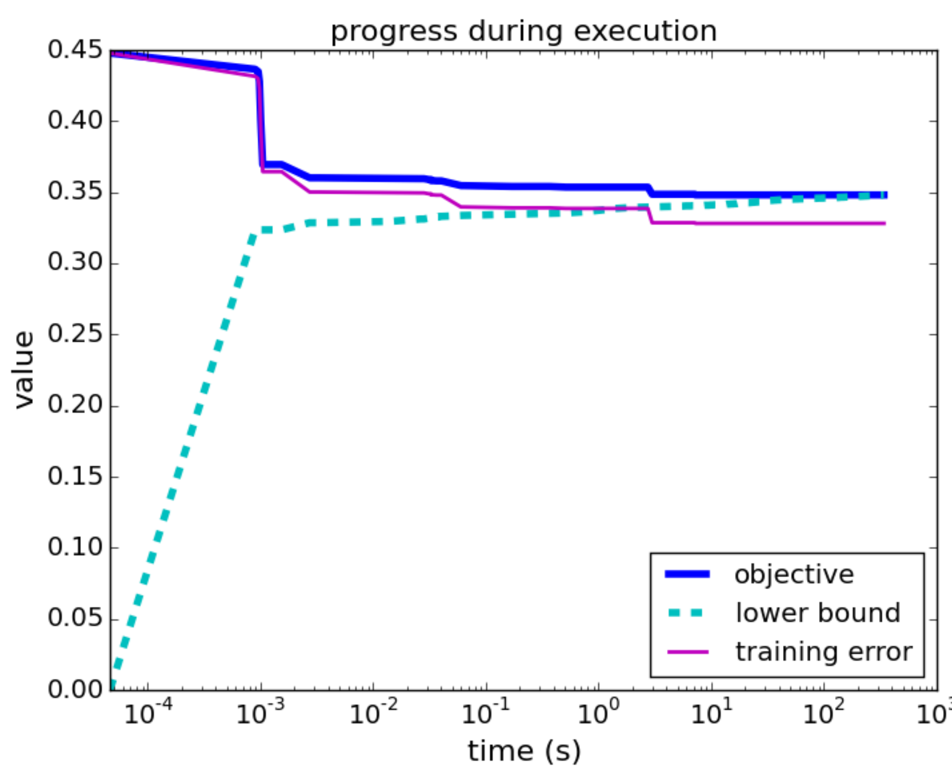


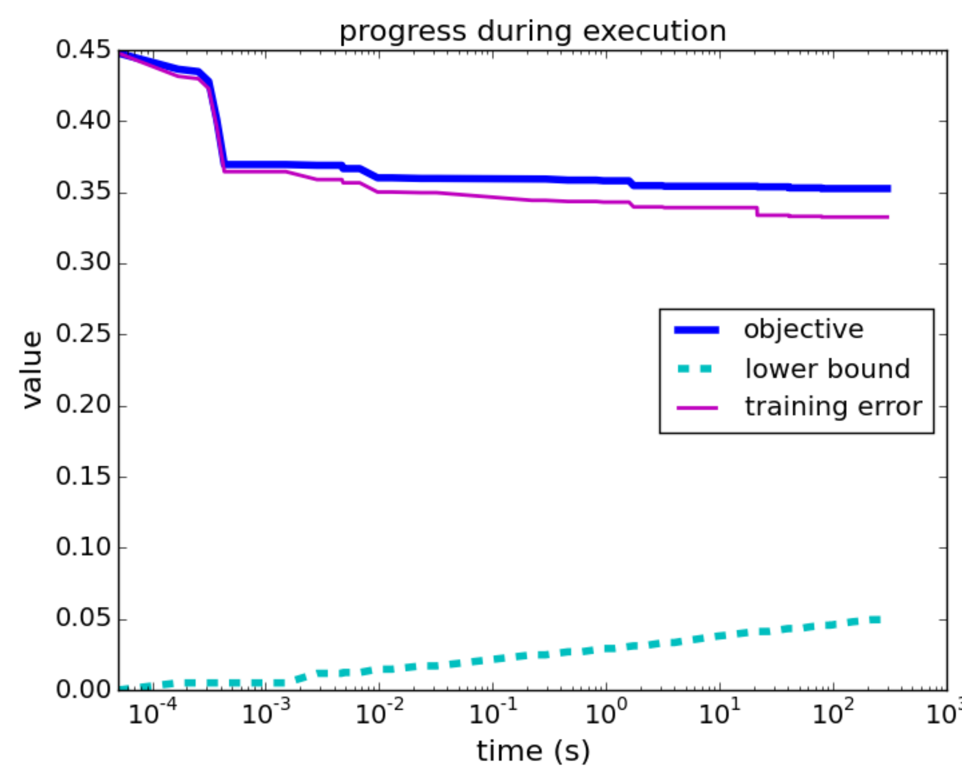


significantly slower without data-driven bound









current and future directions

- tighter data-driven lower bounds
- more memory-efficient data structures
- parallel implementation
- opportunities to use LatticeFlow?
- better scheduling policies?
- other rule mining strategies?
- approximate variants of the algorithm?
- problem can be stated as MAX-SAT

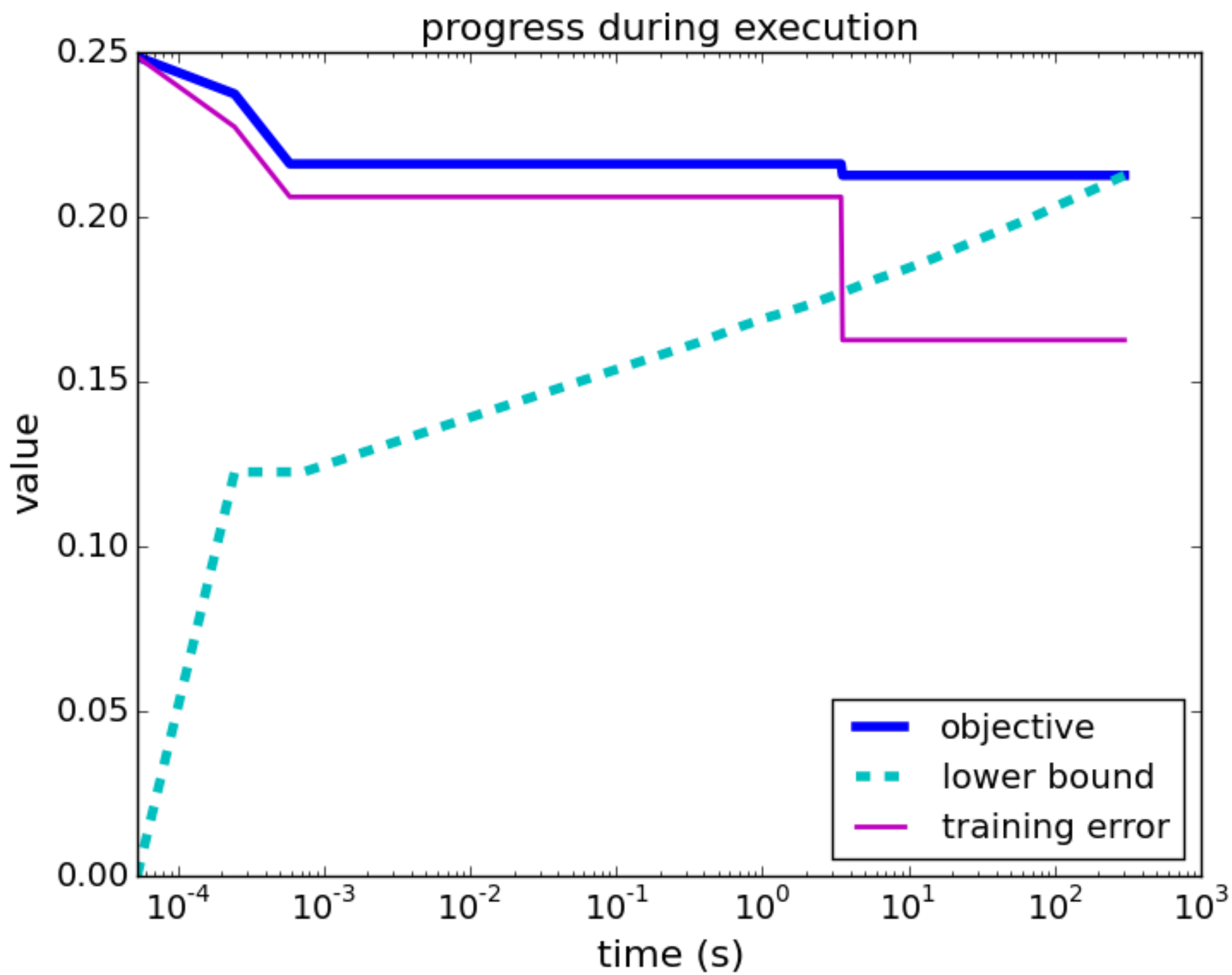
(observation by Johann Schleier-Smith)

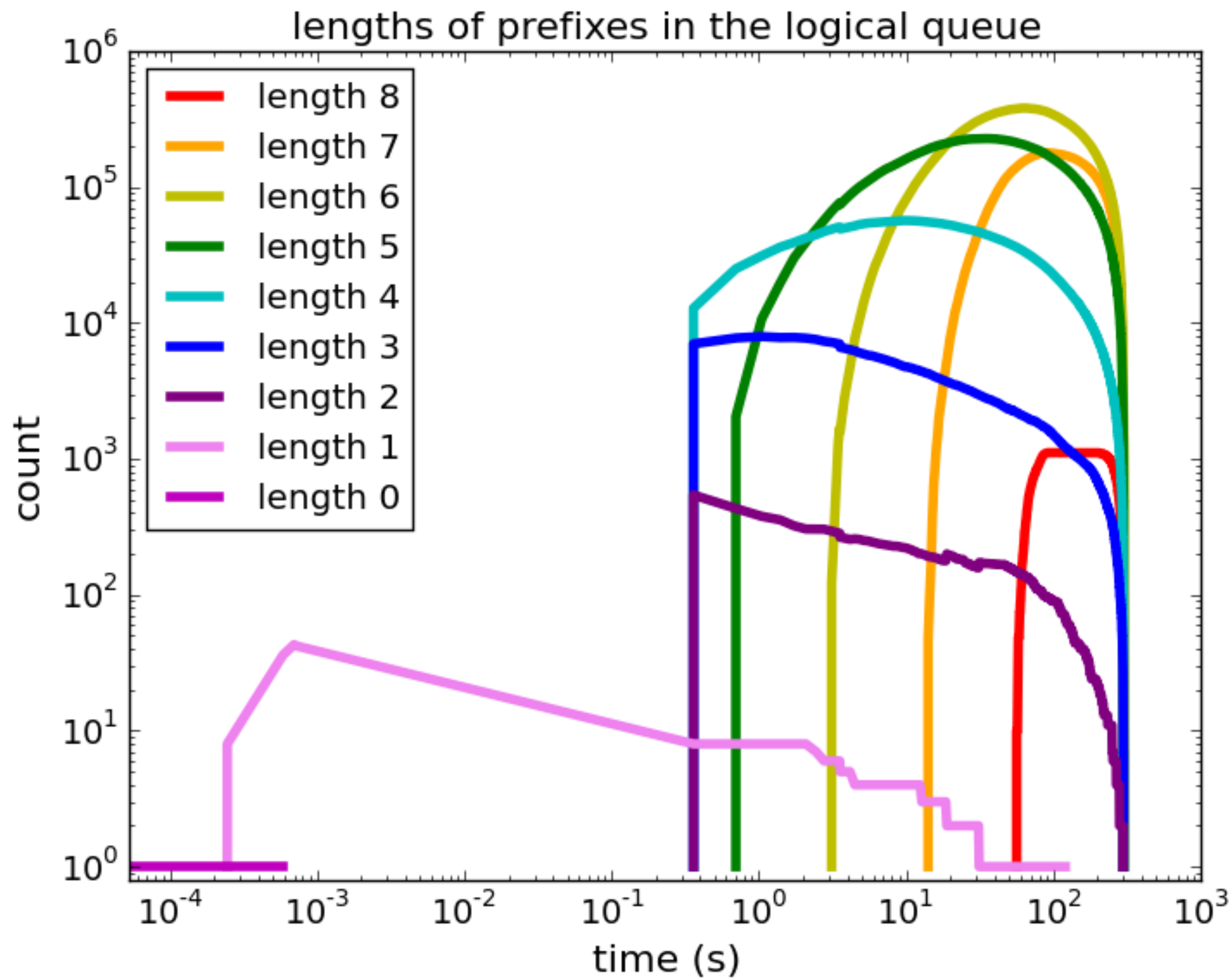
Adult with a single clause performs well

Predict whether income > 50K

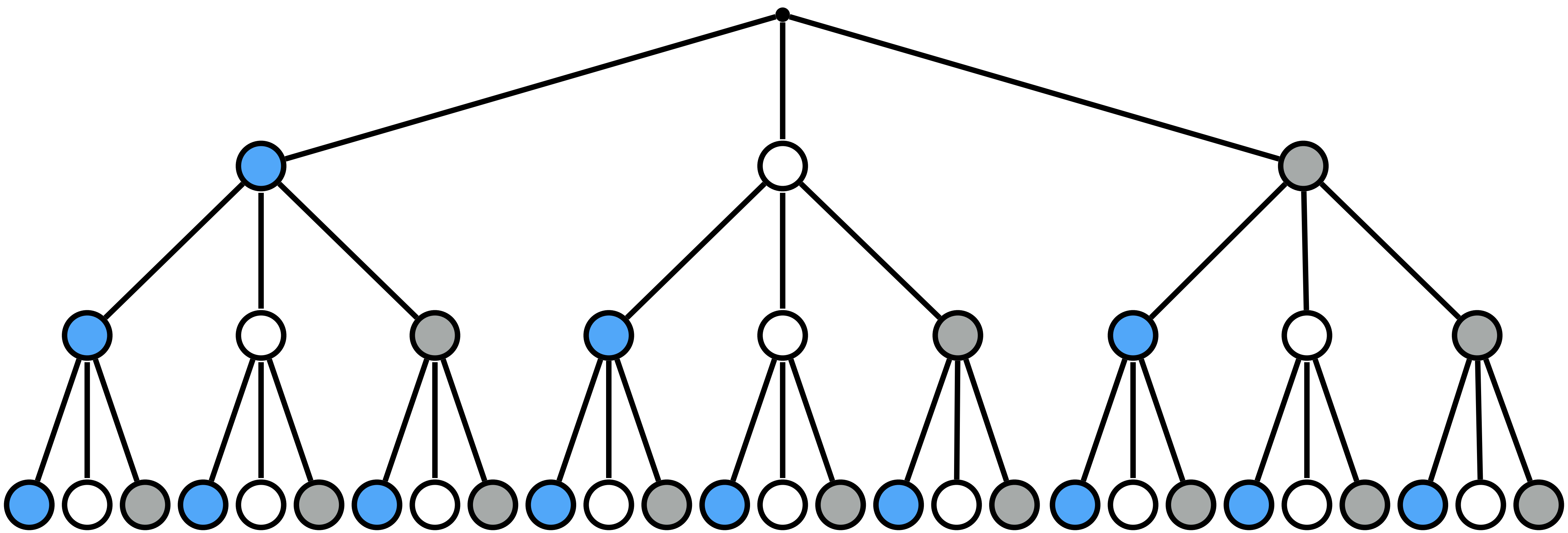
if (capital gains >= 7298) **then** (yes)
else if (never married) **then** (no)
else if (no longer with spouse) **then** (no)
else if (education = graduate school) **then** (yes)
else if (education = Bachelors) **then** (yes)
else (no)

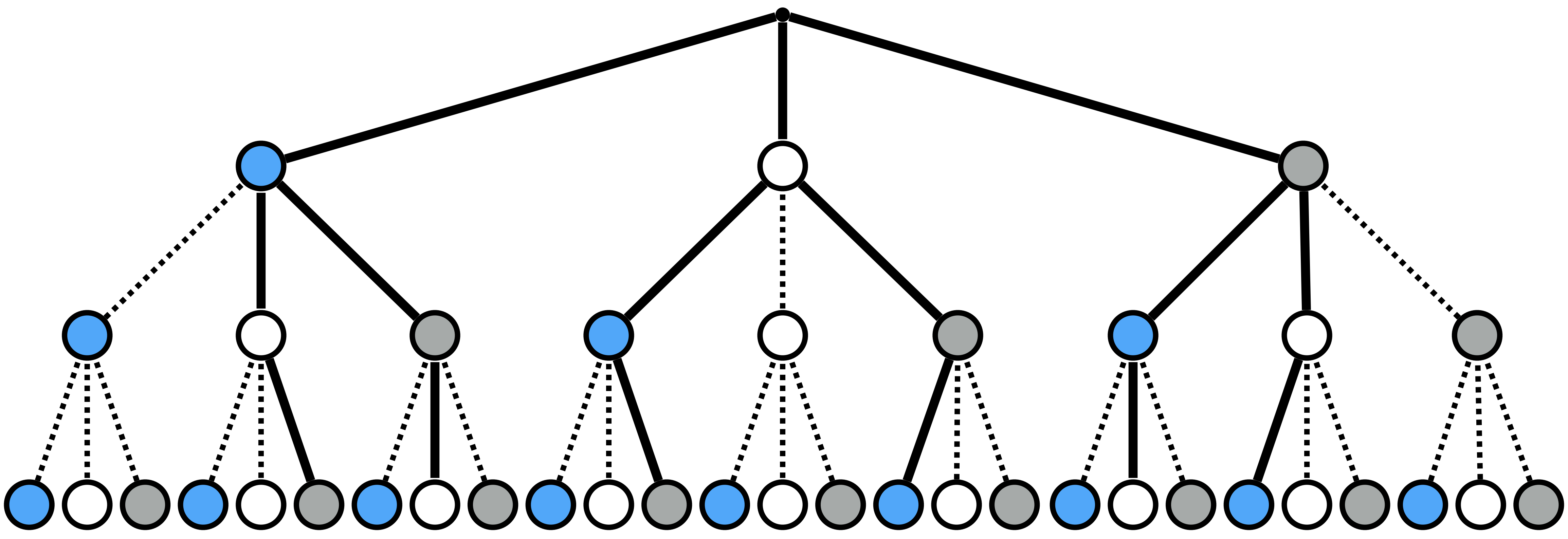
All 10 cross-validation folds find this rule list
Regularization (c) = 0.01
Test accuracy = 0.8376 +/- 0.0045

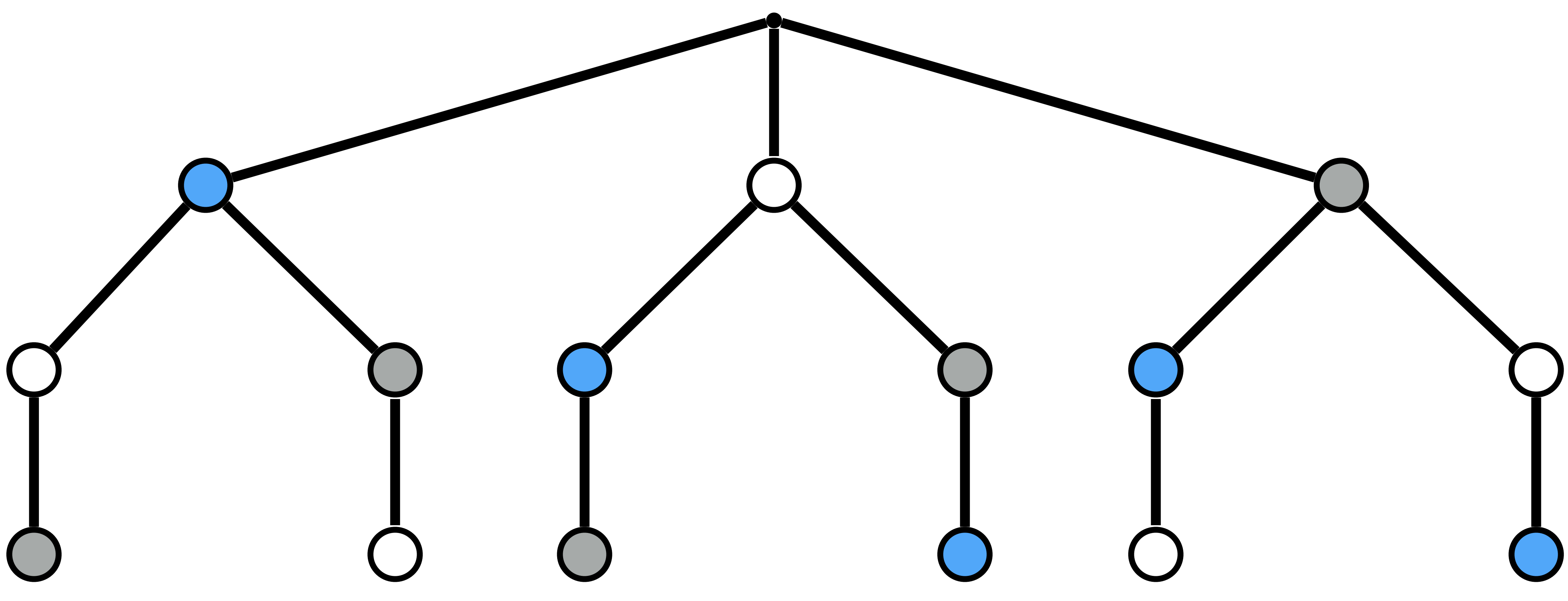

progress during execution
value
time (s)
0.25
0.20
0.15
0.10
0.05
0.00
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
objective
lower bound
training error

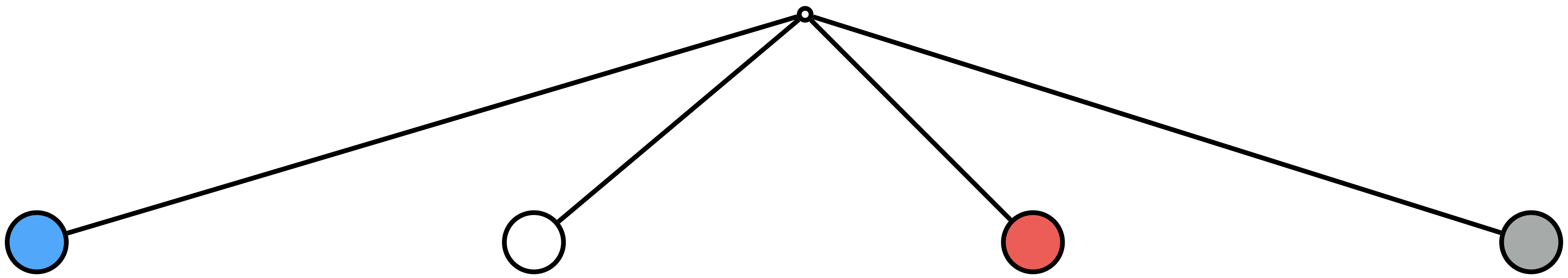


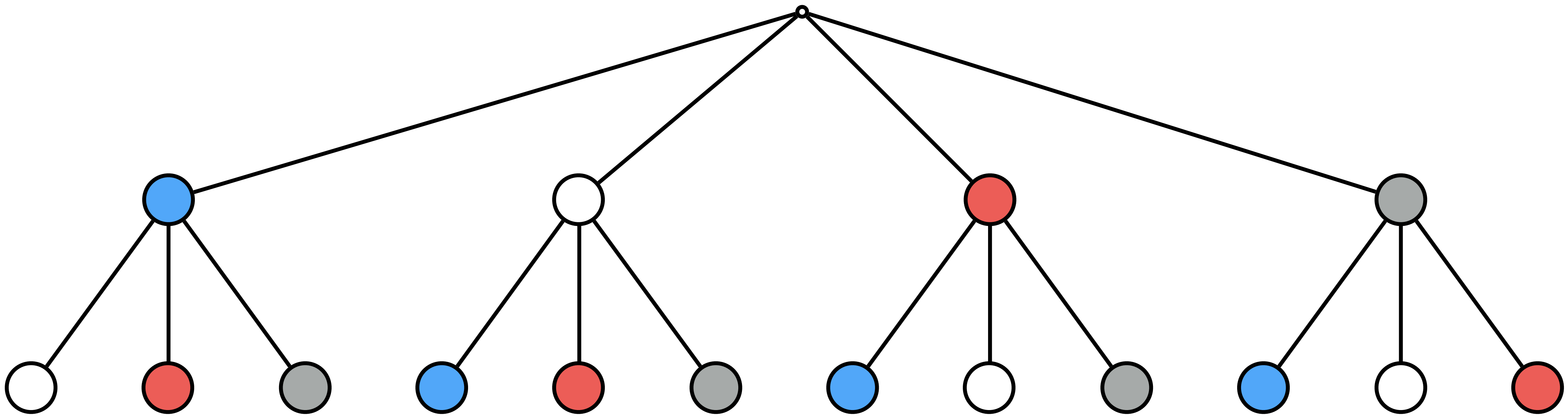
lengths of prefixes in the logical queue
length 8
length 7
length 6
length 5
length 4
length 3
length 2
length 1
length 0
count
time (s)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^0
10^1
10^2
10^3

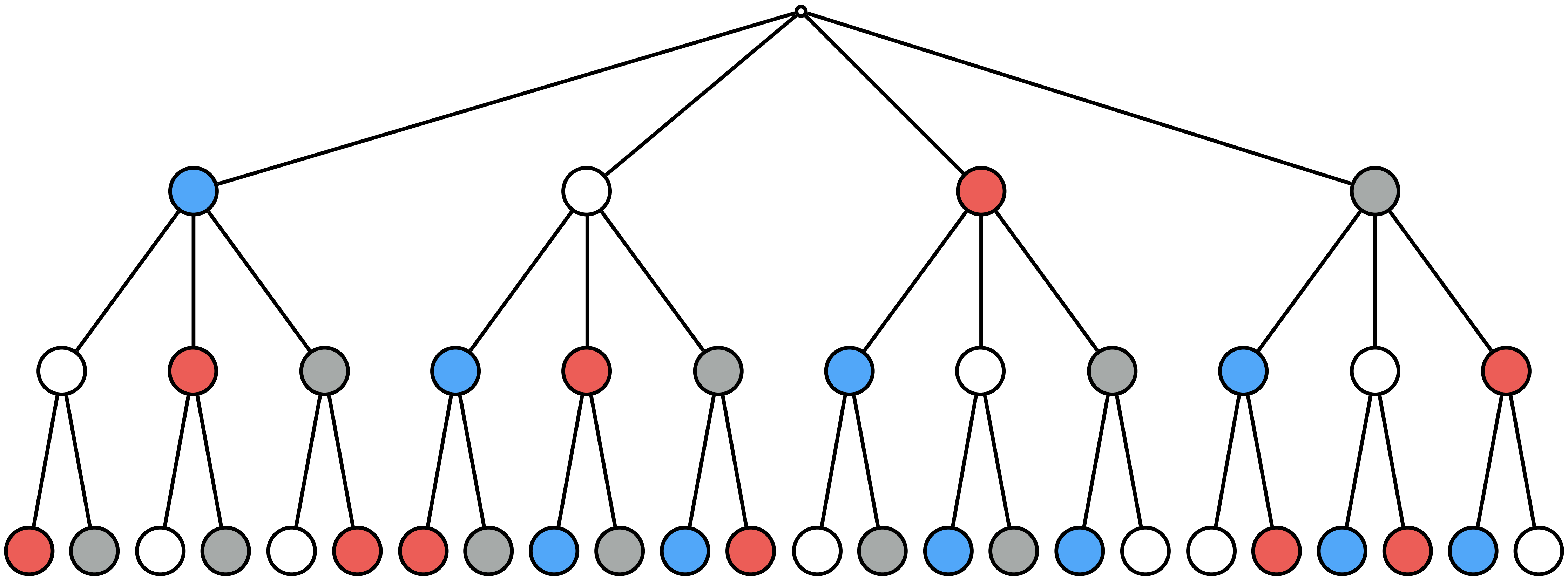


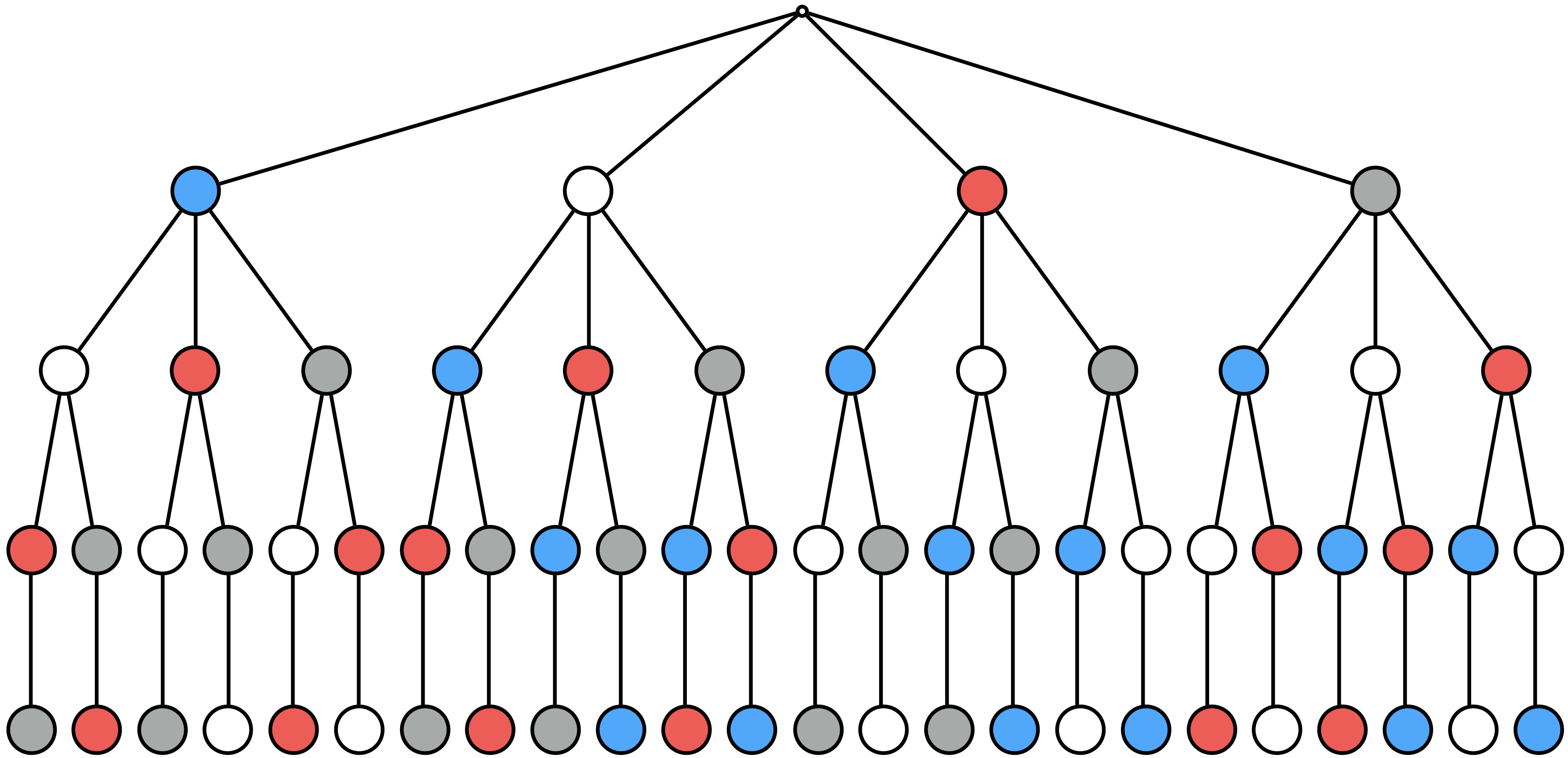


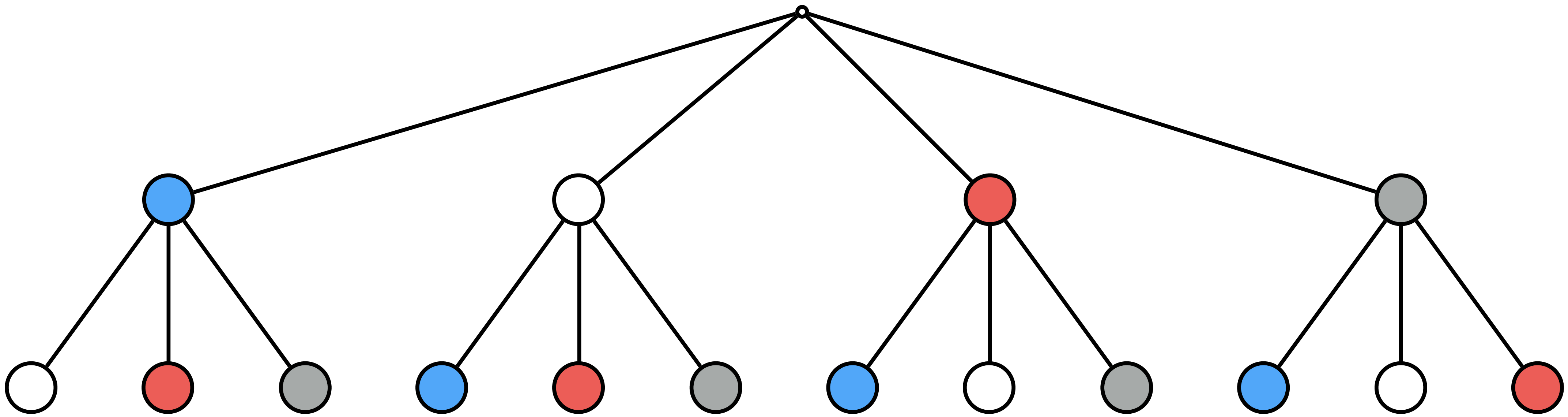


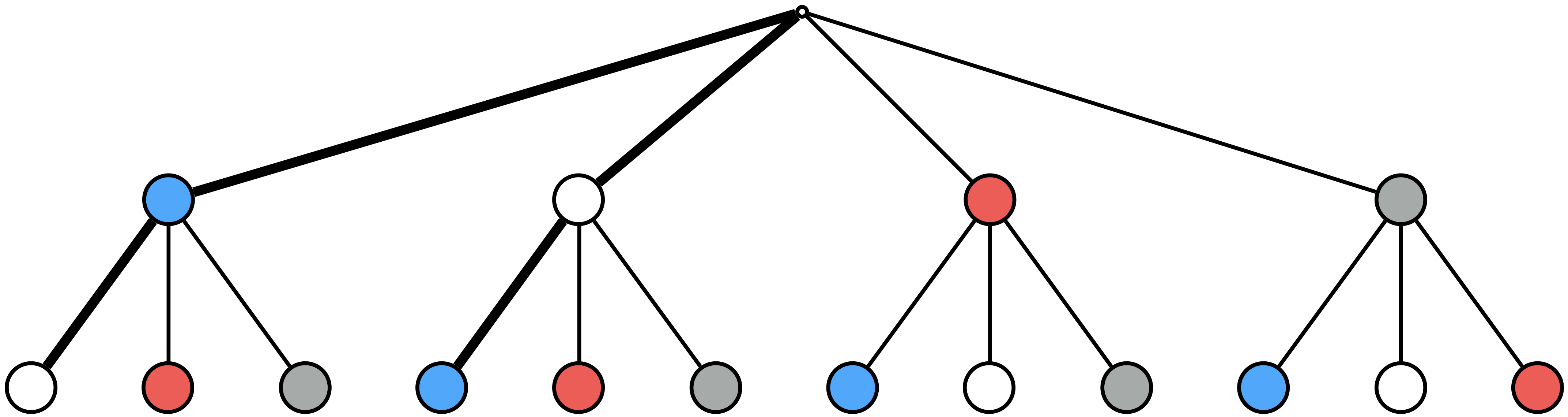


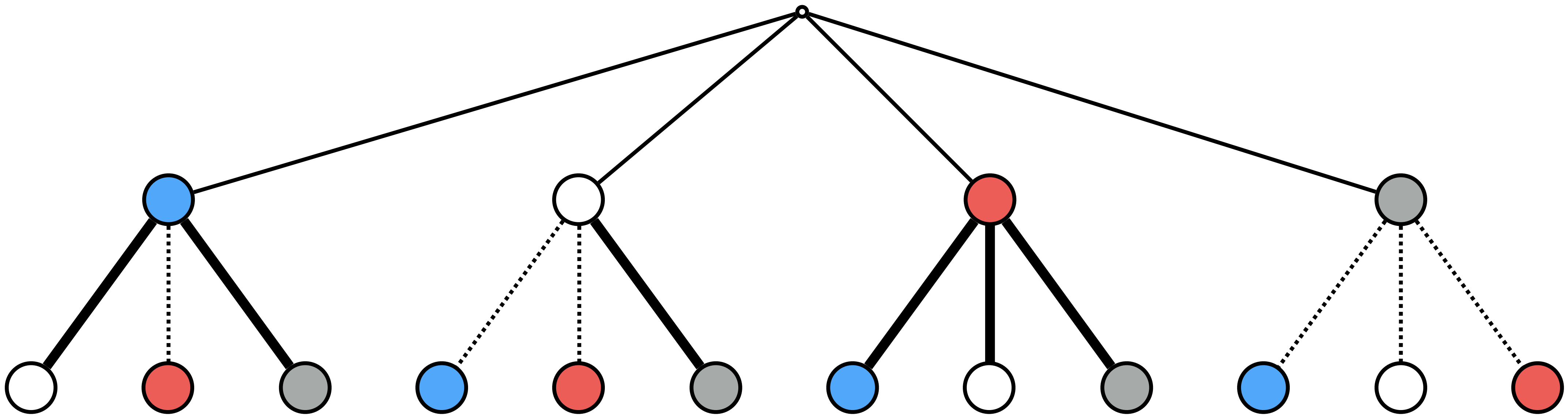


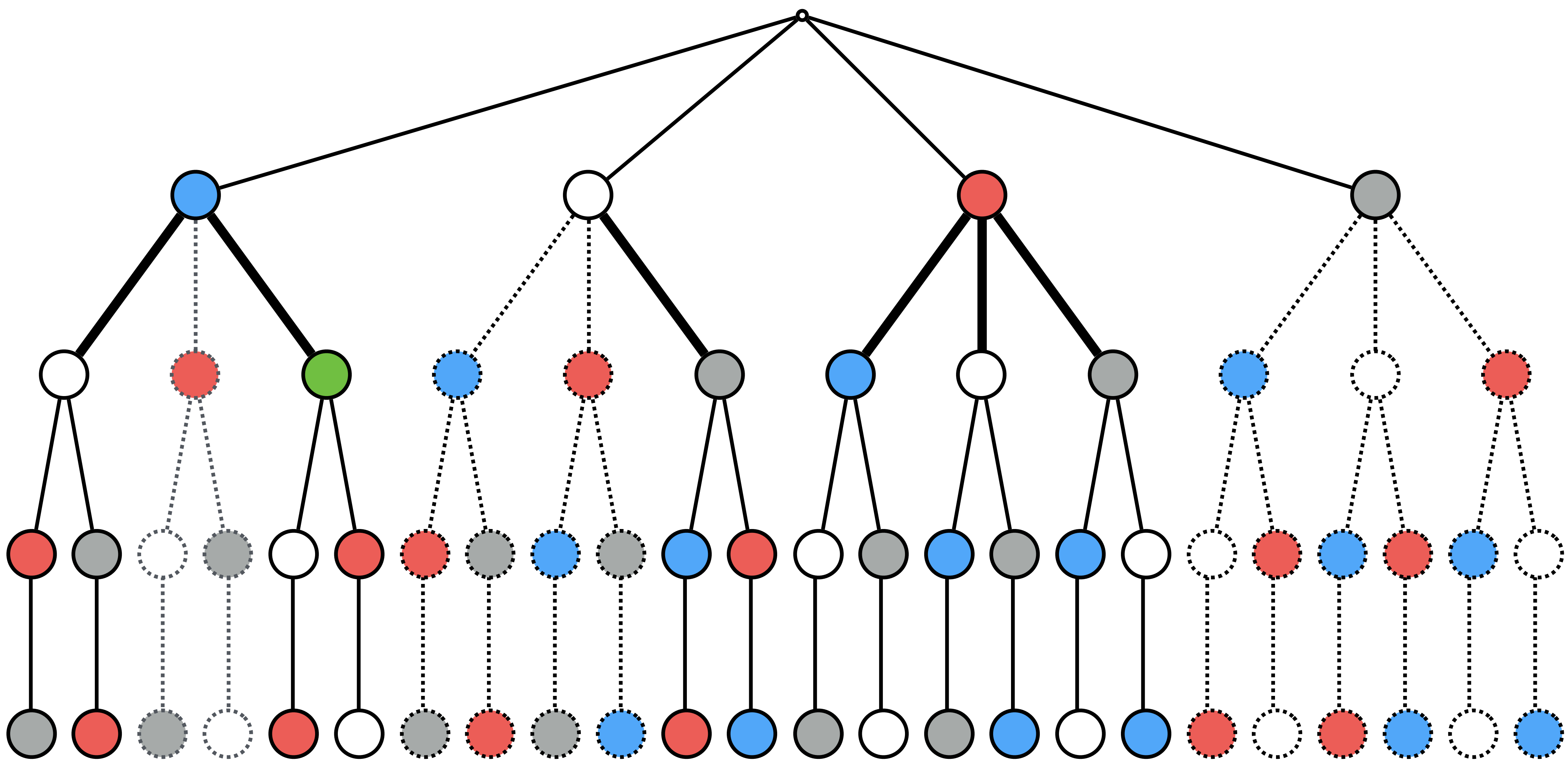


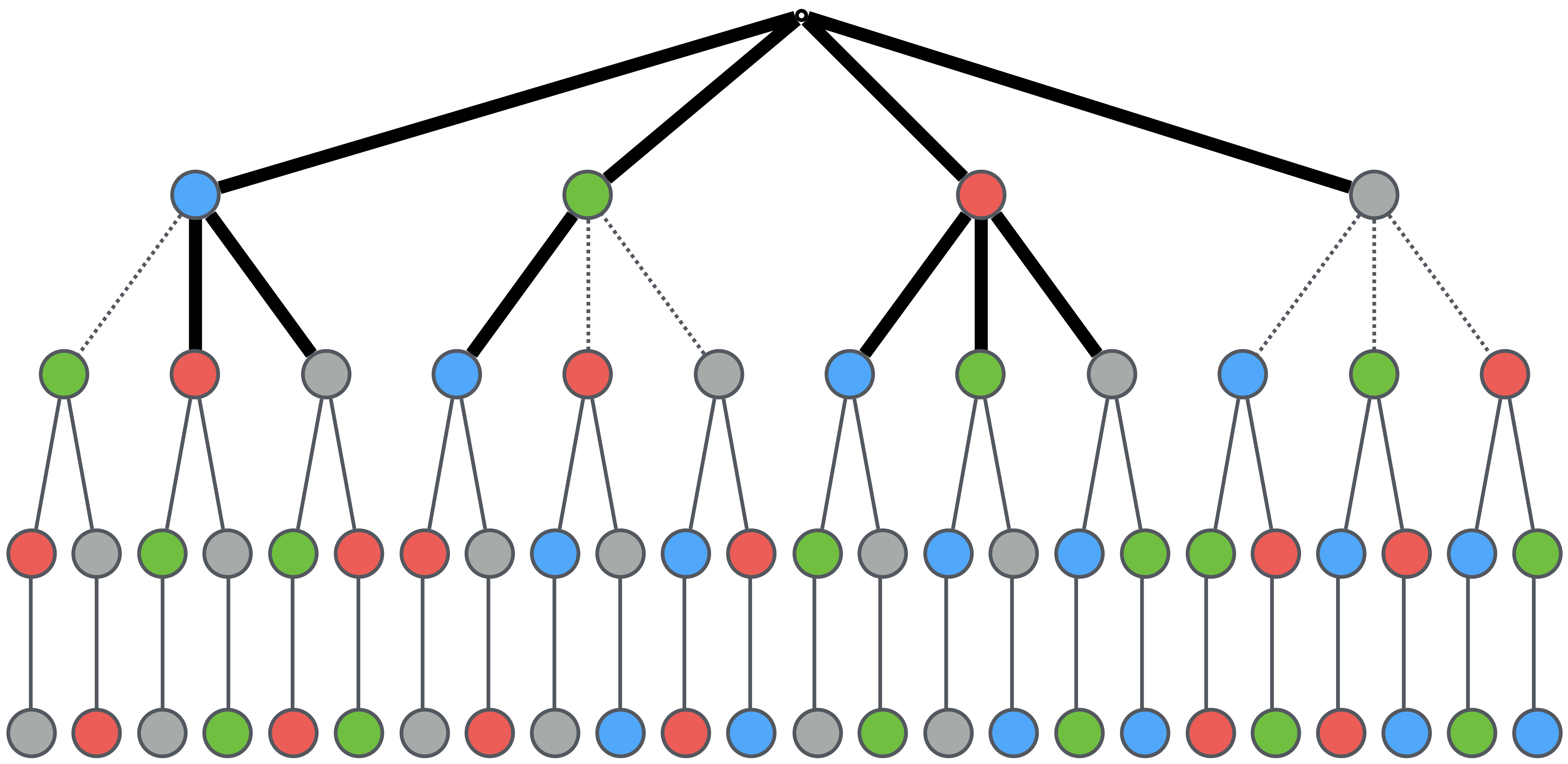




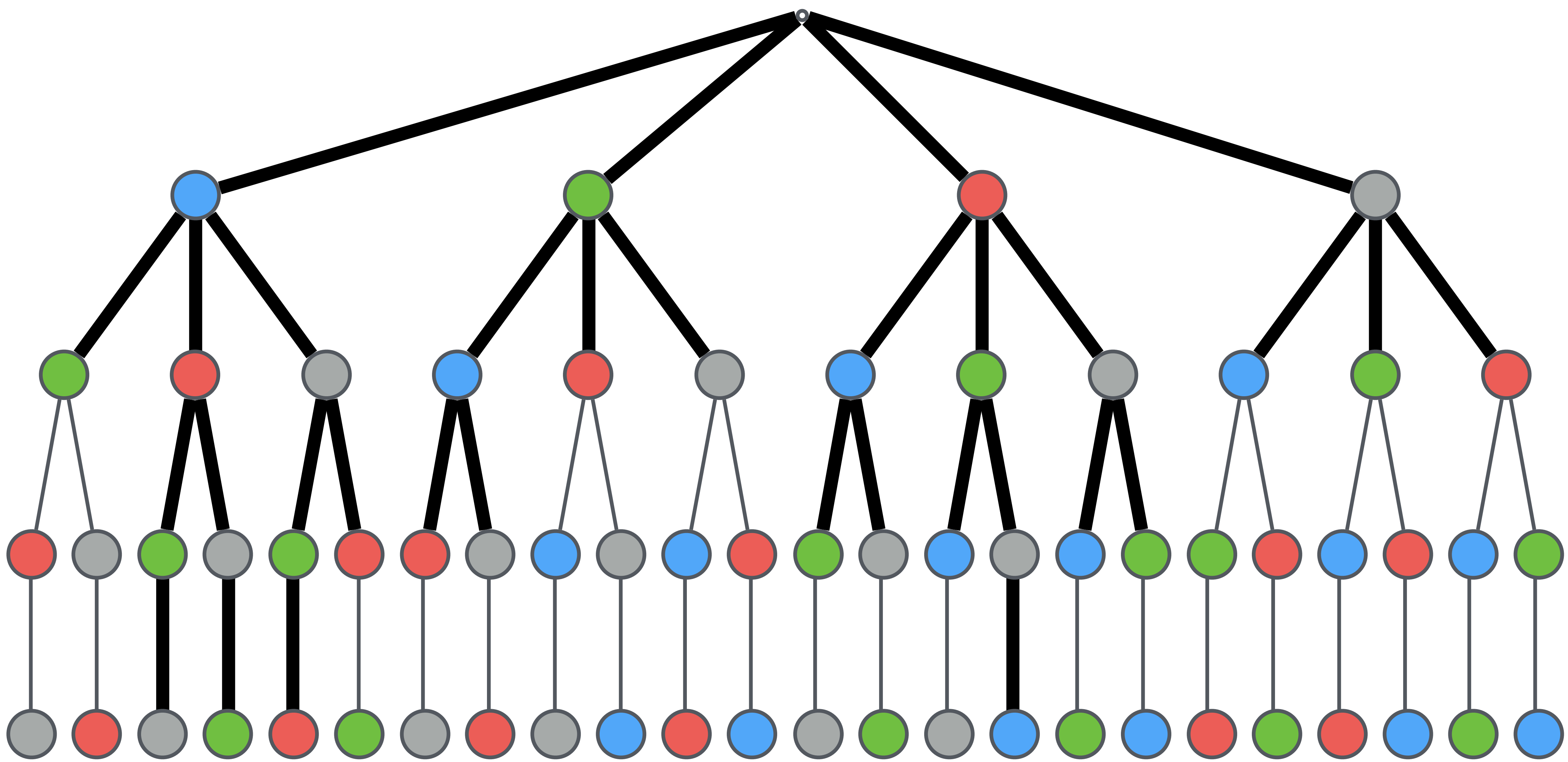








summary of computations



\# states = 4 + (4x3) + (4x3x2) + (4x3x2x1) = 64

\# evaluated states = 4 + (4x3) + (4x3) + 4 = 32

Will the person stay inside or go outside?

if (using computer) **then predict** (stay inside)

else if (time is between 11PM-7AM) **then predict** (stay inside)

else if (sunny **and** not raining/snowing) **then predict** (go outside)

else if (temperature > 55 F) **then predict** (go outside)

else predict (stay inside)

fast branch-and-bound for rule lists

incremental
parallel (?)
asynchronous (?)

global combinatorial optimization algorithm
(vs. greedy heuristics or random search)

human-interpretable
machine learning

with a symmetry-aware cache

exploit problem structure
(permutations)

data structures to store &
reuse computation

Joint work with Nicholas Larus-Stone, Margo Seltzer, Cynthia Rudin