CS 181AG
Lecture 10

Prefix Lookup Cont. + Intro to Packet Classification

Arthi Padmanabhan

Oct 3, 2022

Recap

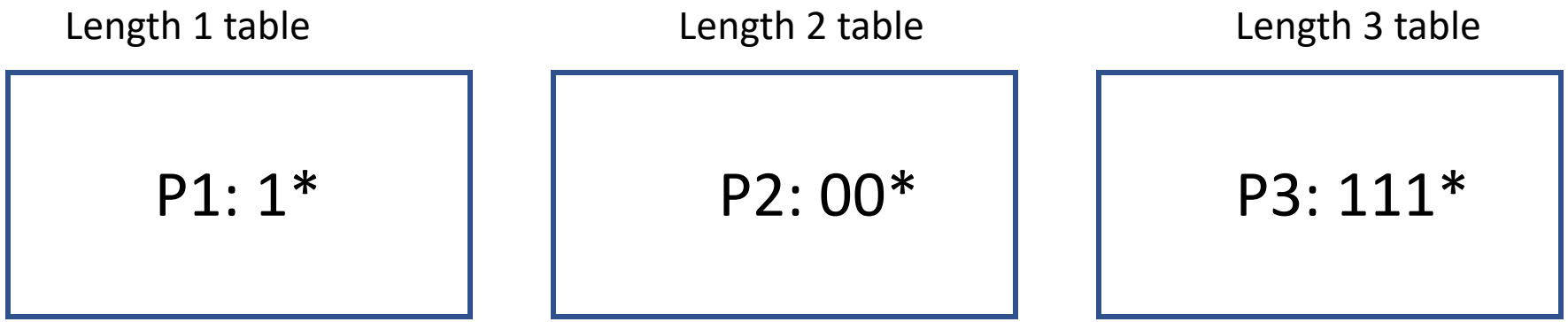
- Unibit tries have a suboptimal worst-case lookup (32)
- We can use multibit tries to look at more than one bit at a time, but then we have to expand prefixes, and that uses a lot of memory
- We can address the memory problem using compression (store only a bit array and a value array per node)

Non-Trie Options

- If this were an exact match problem instead of longest match, what approach could we use?

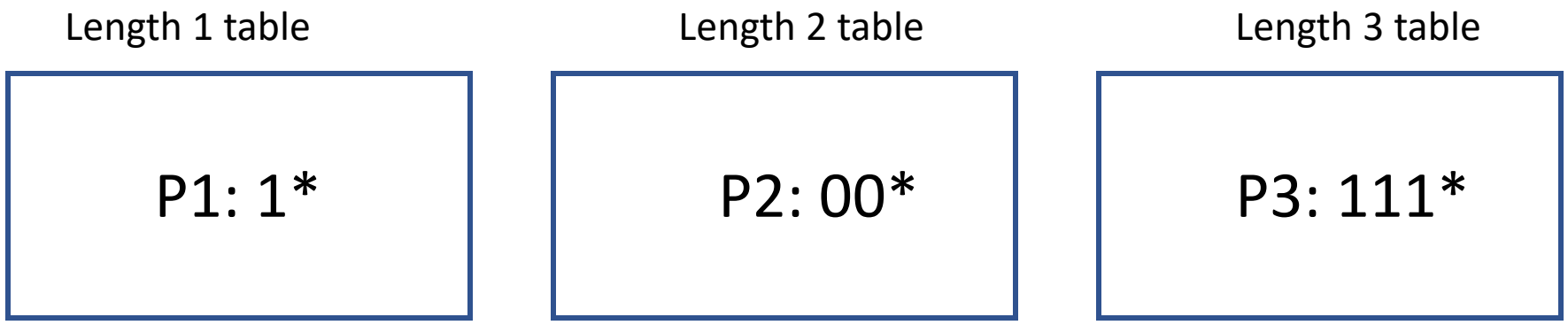
Non-Trie Options

- Split by prefix length
- Makes sense to start from longest, but then we have worst-case lookup of 32



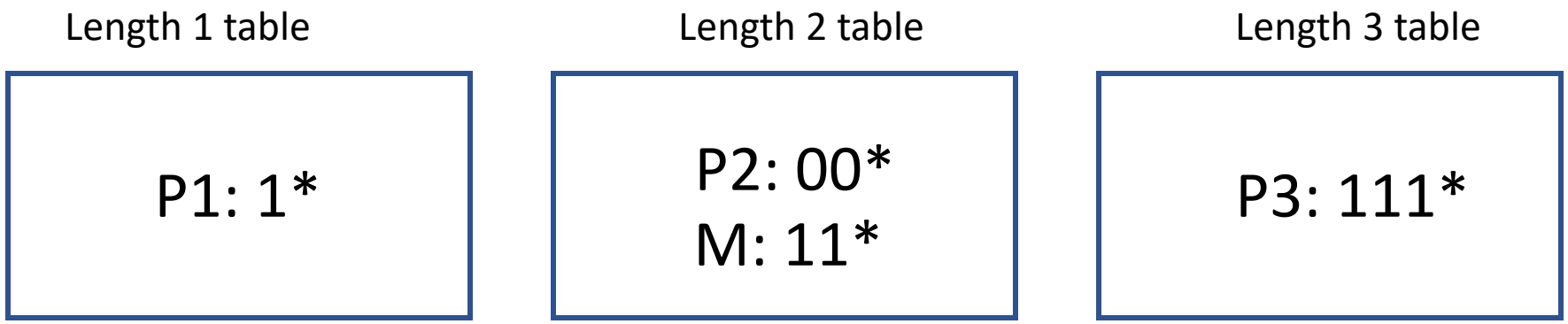
Non-Trie Options

- Binary search on prefix lengths
- Problem?



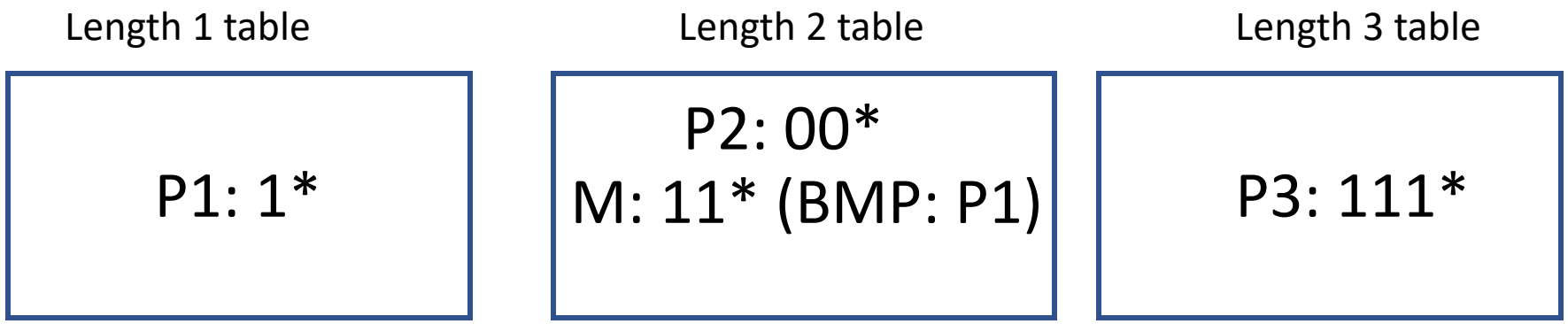
Non-Trie Options

- Binary search on prefix lengths



Non-Trie Options

- Binary search on prefix lengths



Add markers and BMPs to database

- Separate by prefix lengths and add marker + BMP for each element where needed
- If prefix already exists in shorter-length table, you don't need to add a marker

Interface	Prefix
P1	1*
P2	0*
P3	10*
P4	010*
P5	111*
P6	1000*
P7	11001*
P8	101000*
P9	110100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10*	P4: 010* P5: 111*	P6: 1000*	P7: 11001*	P8: 101000*	P9: 1100100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10*	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1)	P7: 11001*	P8: 101000*	P9: 1100100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10*	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000*	P9: 1100100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000*	P9: 1100100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

11000000…

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

11000000…
Matches marker, remember
P1, move right

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

11000000…
No match, move left
Still remember P1

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

11000000…
No match, search ends
Return P1

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

01100000…
110010000…
10110000…

Add markers and BMPs to database

Length 1	Length 2	Length 3	Length 4	Length 5	Length 6	Length 7
P1: 1* P2: 0*	P3: 10* M: 01* (BMP: P2) M: 11* (BMP: P1)	P4: 010* P5: 111*	P6: 1000* M: 1100* (BMP: P1) M: 1010* (BMP: P3)	P7: 11001*	P8: 101000* M: 110010* (BMP: P7)	P9: 1100100*

01100000… -> P2
110010000… -> P9
10110000… -> P3

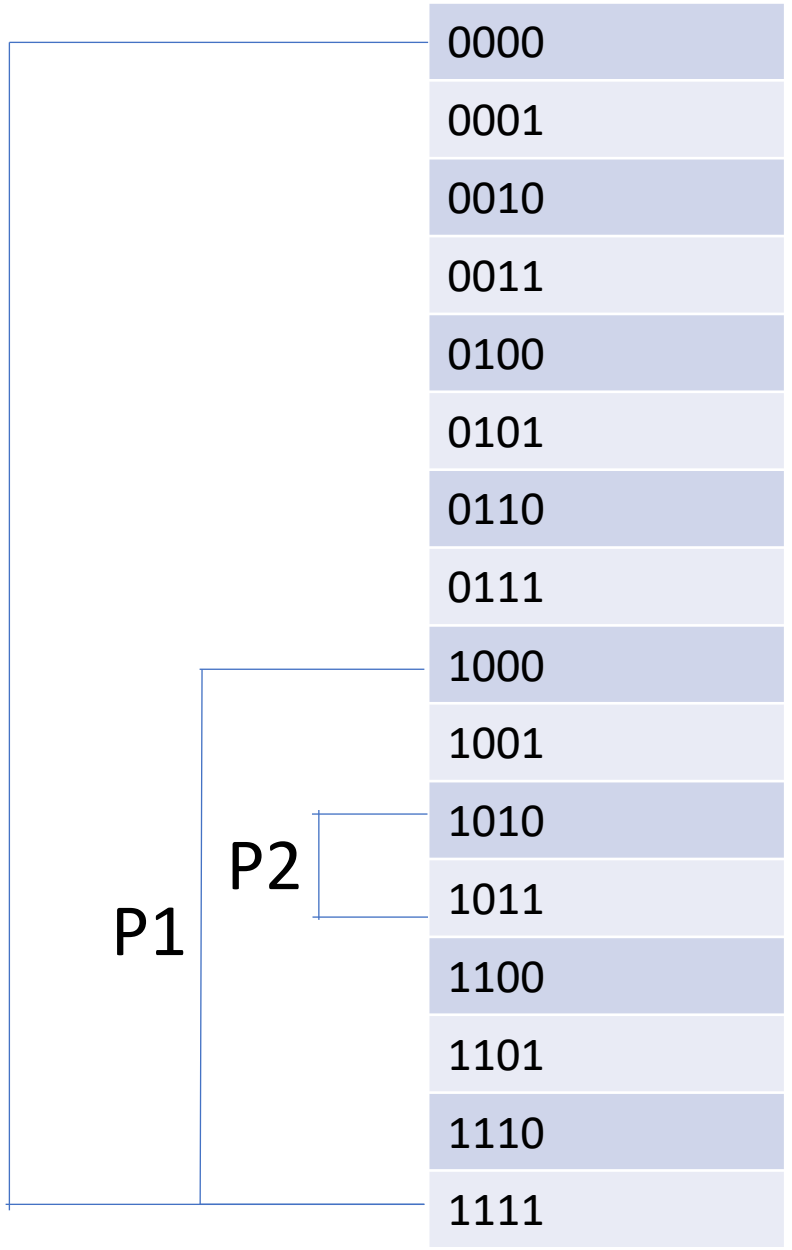
Prefixes as Ranges

- Assume we only have two prefixes P1 = 1*, and P2 = 101*
- Each of these could be seen as a range of IP addresses. For simplicity, let's say IPs are 4 bits long
  - P1: 1000 – 1111
  - P2: 1010 - 1011

Prefixes as Ranges

- P1: 1000 – 1111
- P2: 1010 - 1011

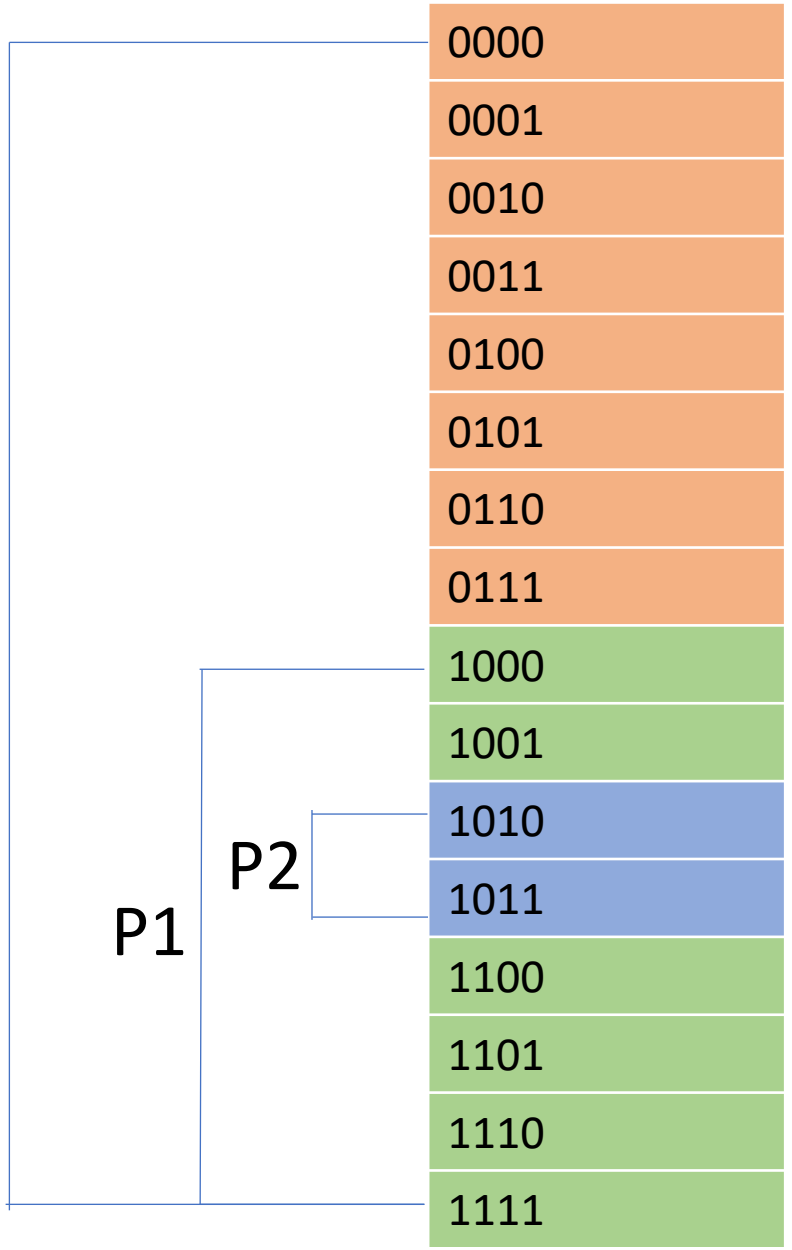
-



Prefixes as Ranges

- P1: 1000 – 1111
- P2: 1010 - 1011

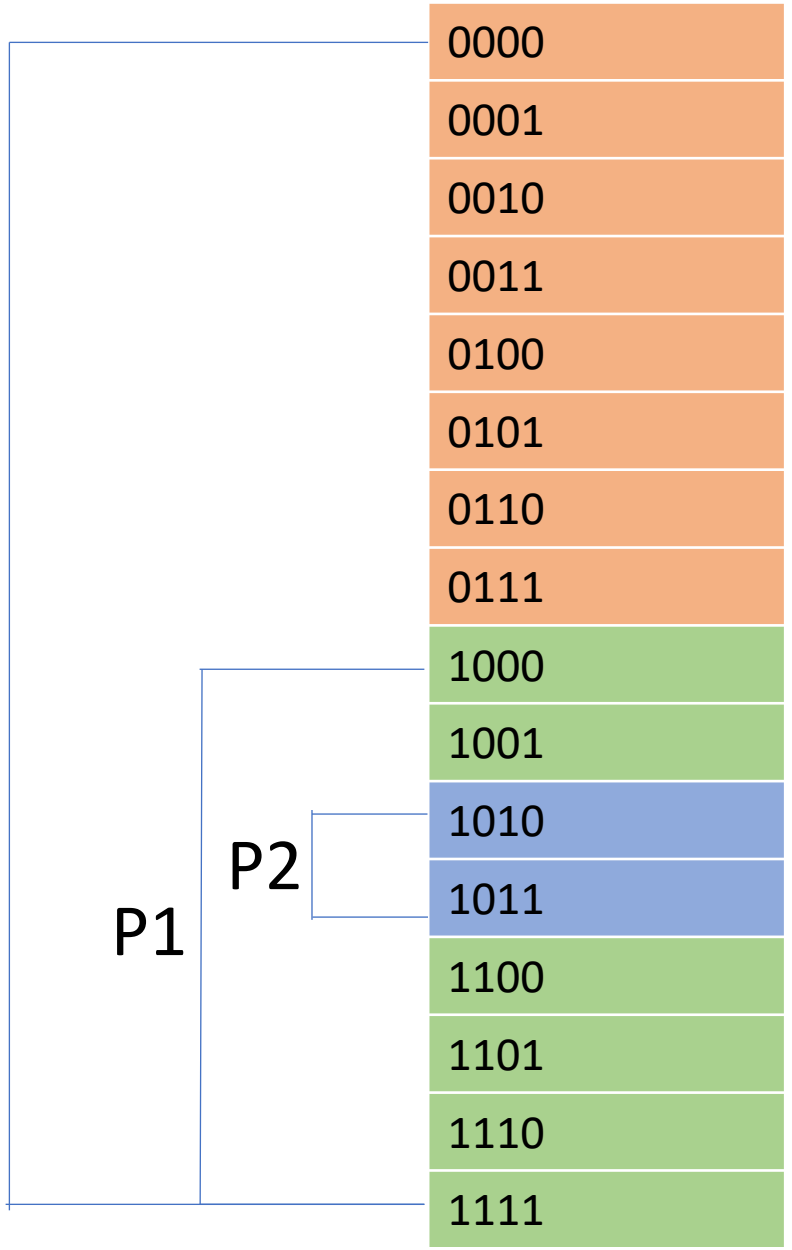
-



Prefixes as Ranges

- Consider the endpoints:
  - 0000
  - 1000
  - 1010
  - 1011
  - 1111

-

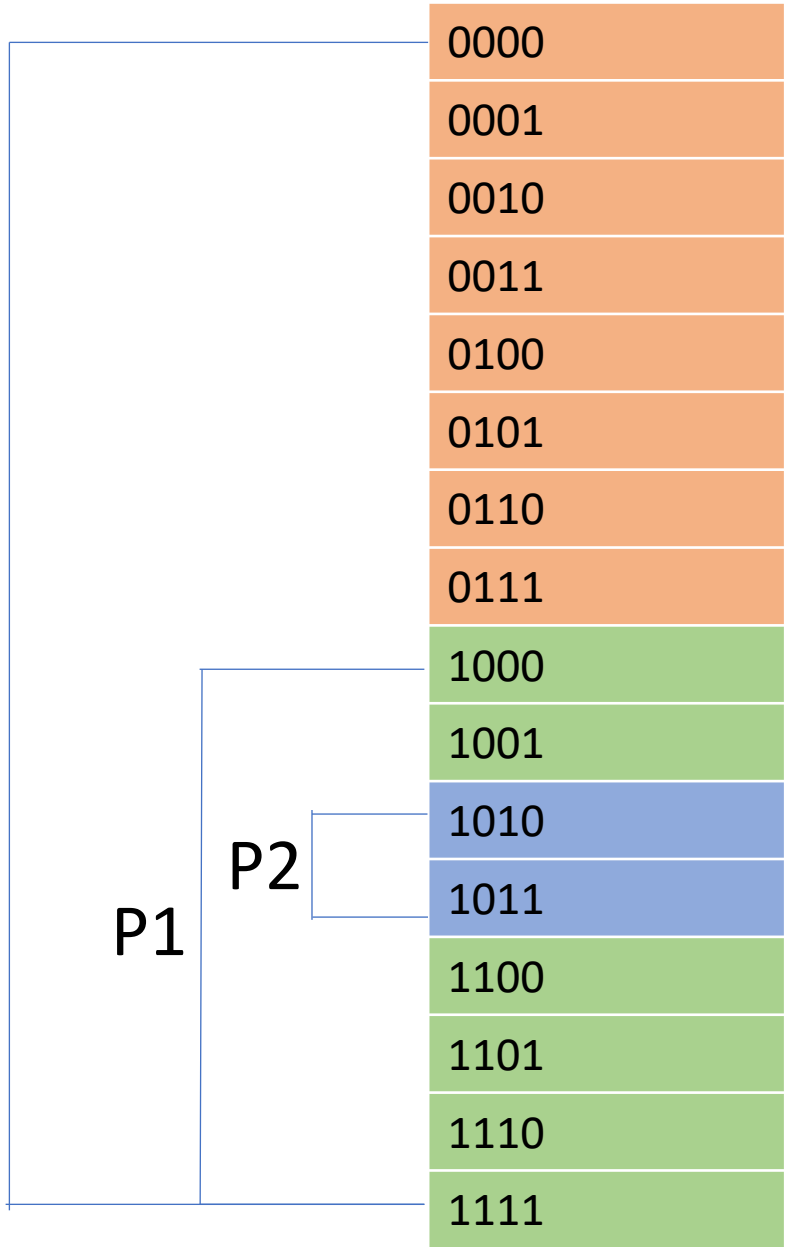


Prefixes as Ranges

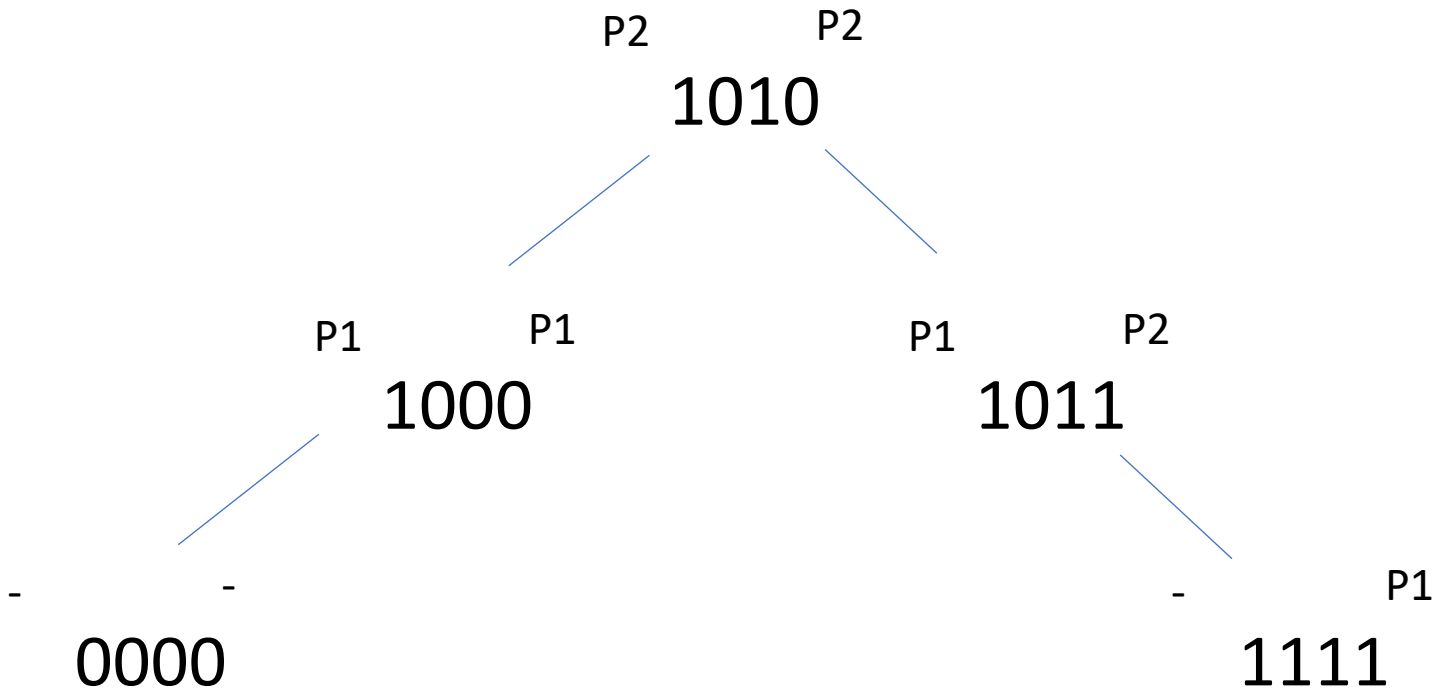
- Consider the endpoints:

IP	>	=
0000	-	-
1000	P1	P1
1010	P2	P2
1011	P1	P2
1111	-	P1

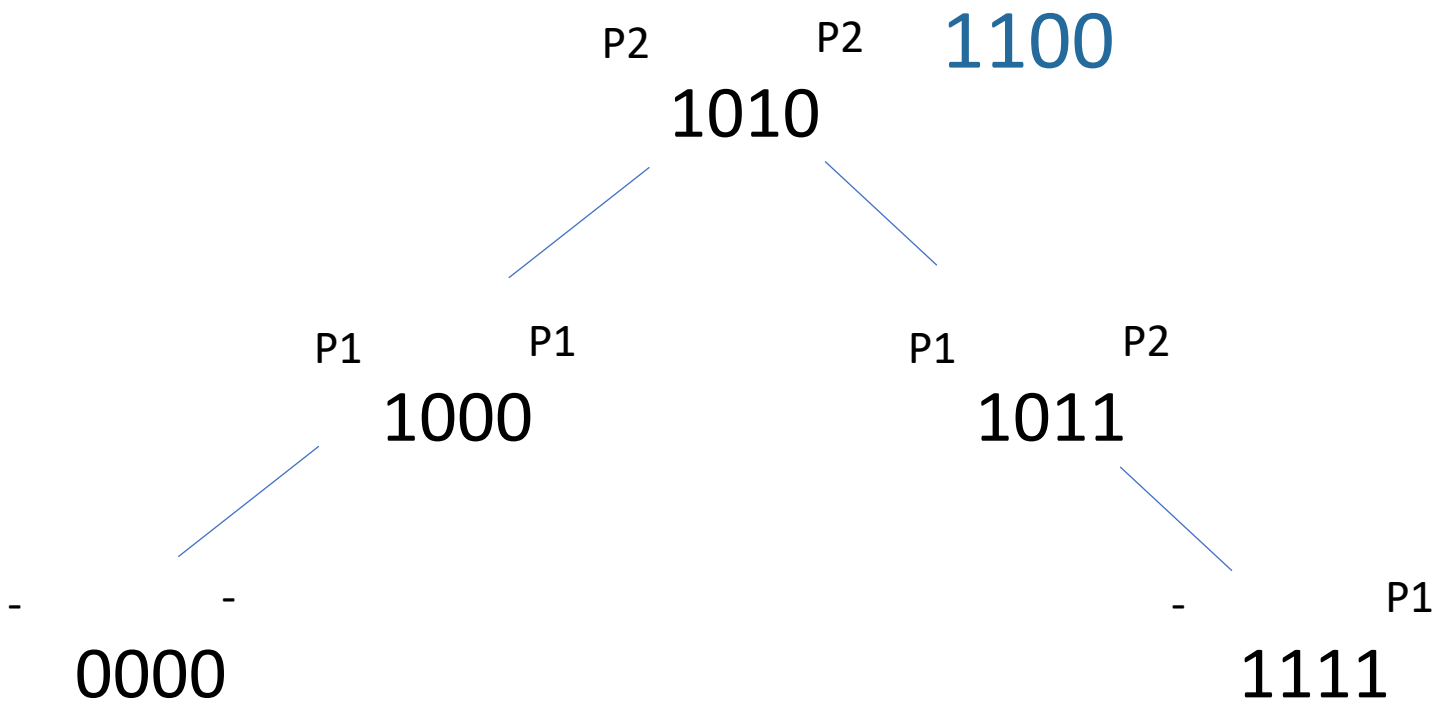
-



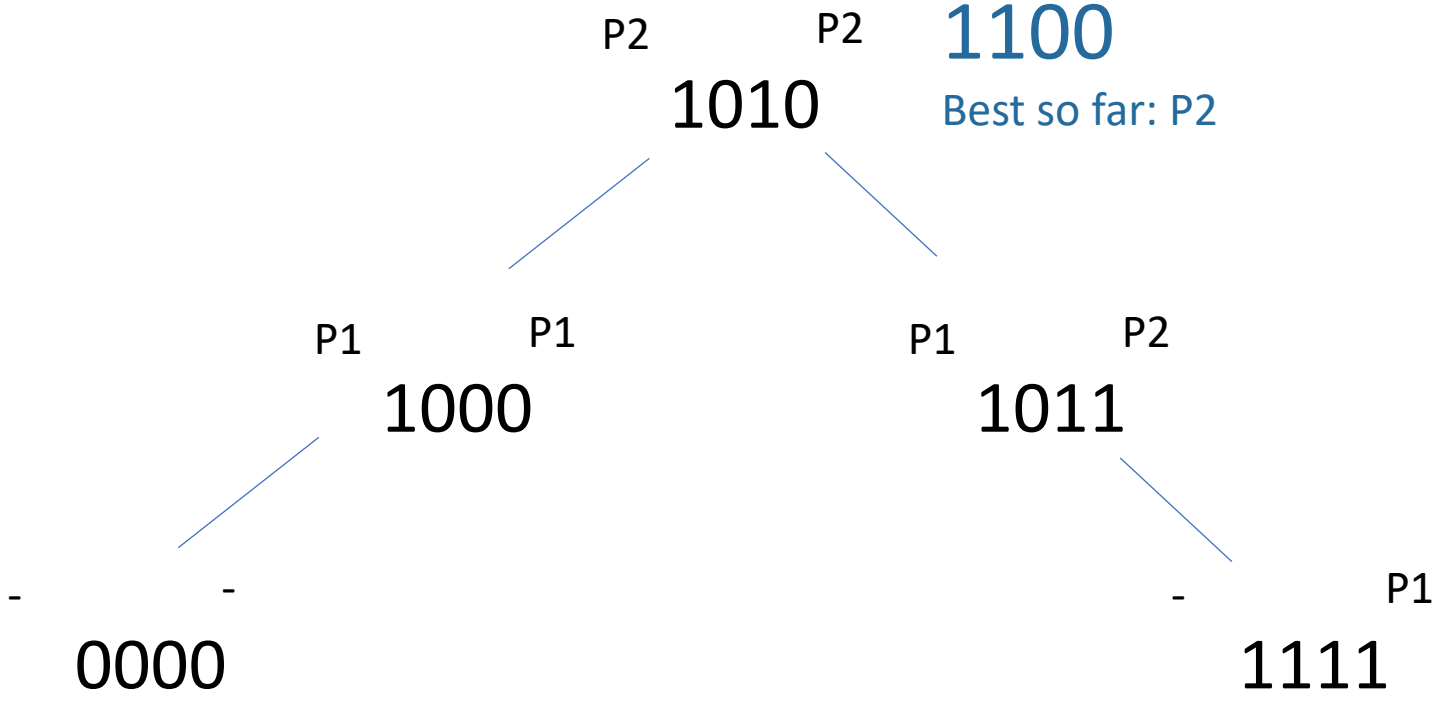
Binary Search on Ranges



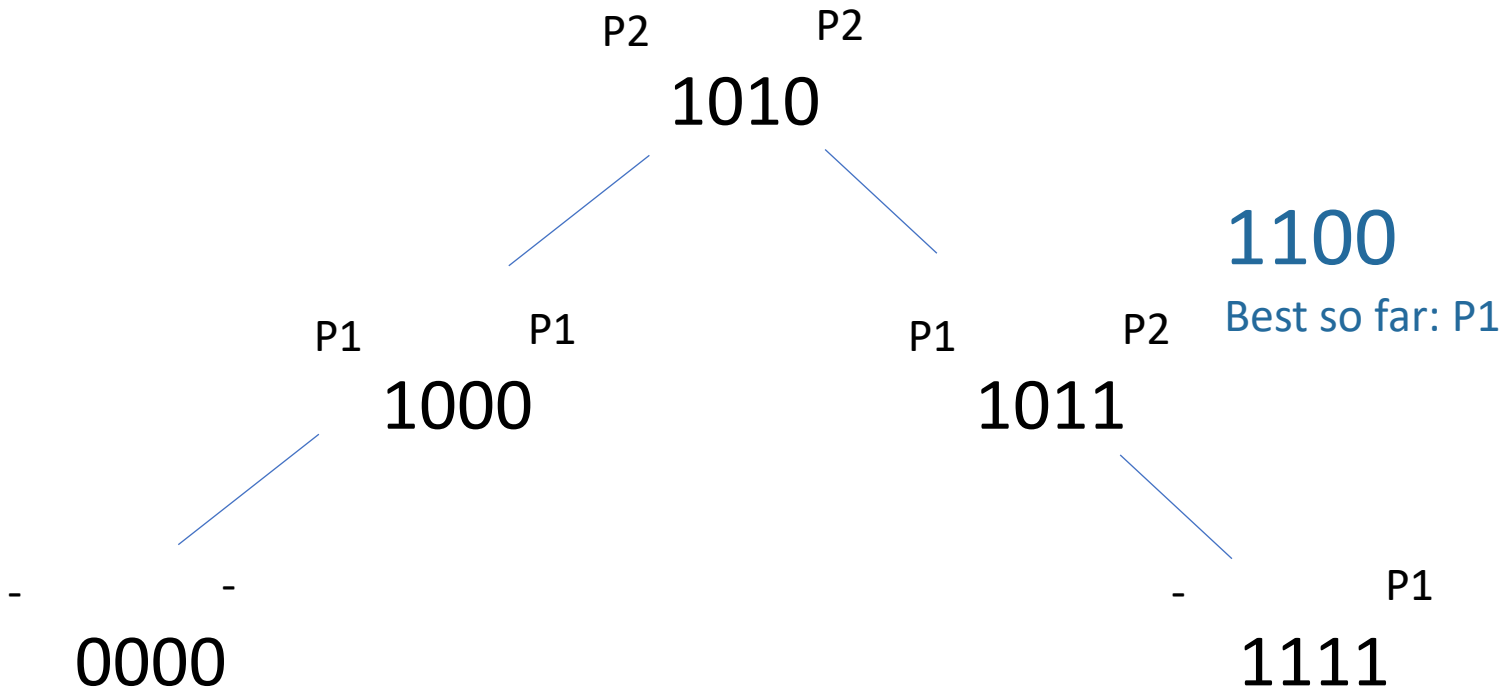
Binary Search on Ranges



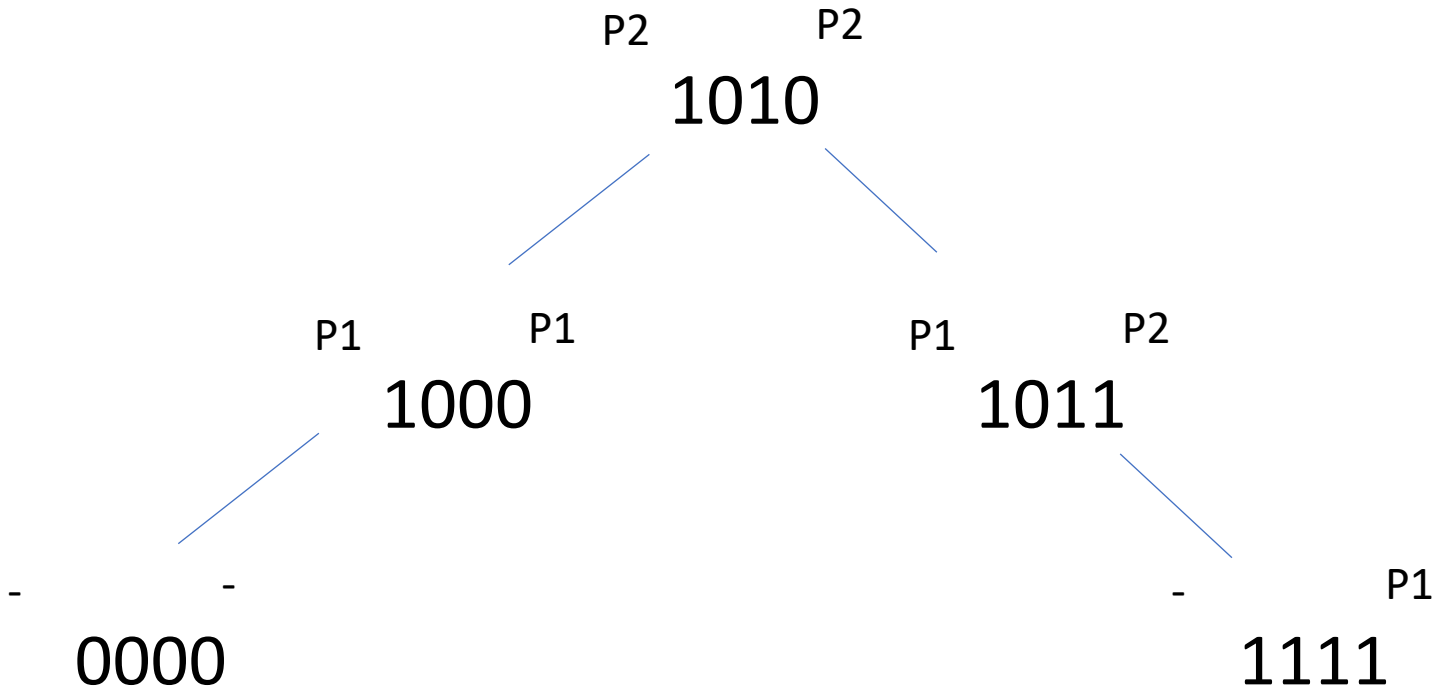
Binary Search on Ranges



Binary Search on Ranges



Binary Search on Ranges



Return P1

Other Thoughts

- These methods are both worse than tries in terms of lookups
- They are not patented though, so they did get some use