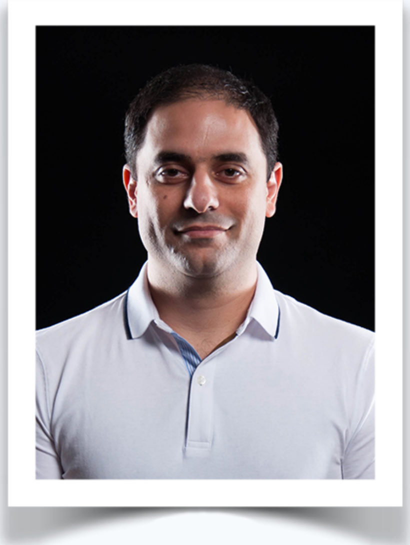
Storage & Indexing in Modern Databases

ECS 165A – Winter 2022



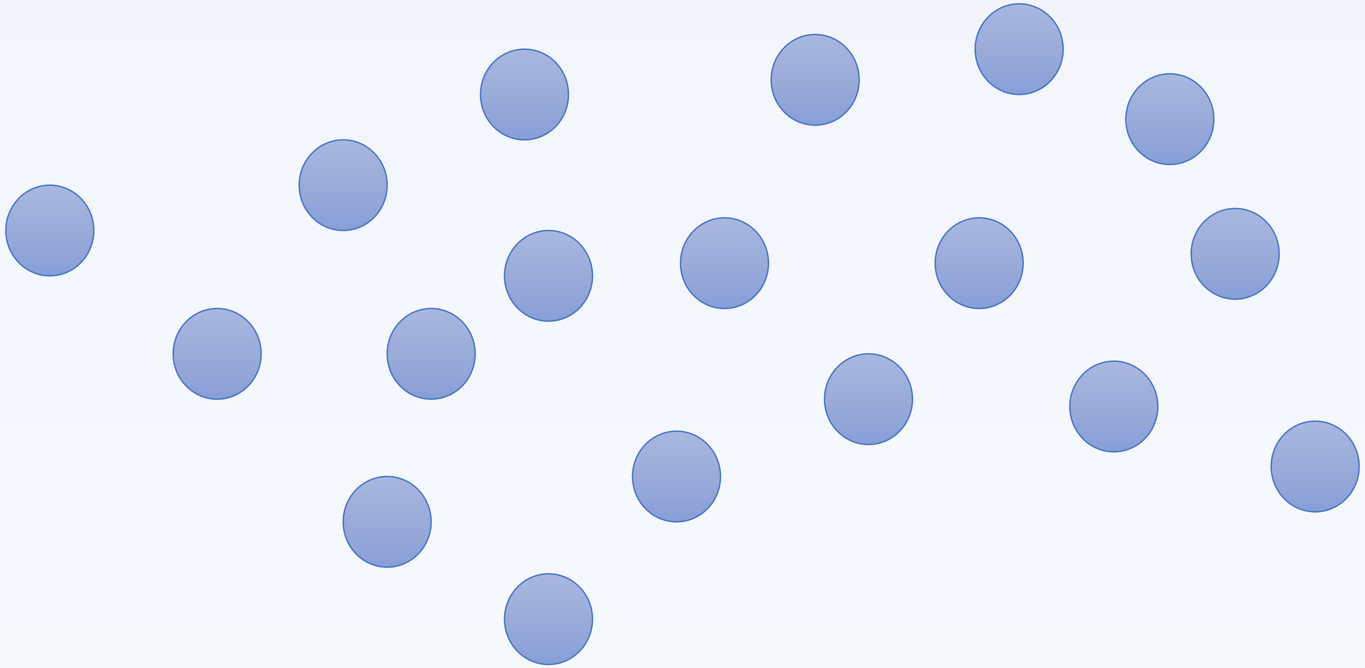
Mohammad Sadoghi
Exploratory Systems Lab
Department of Computer Science

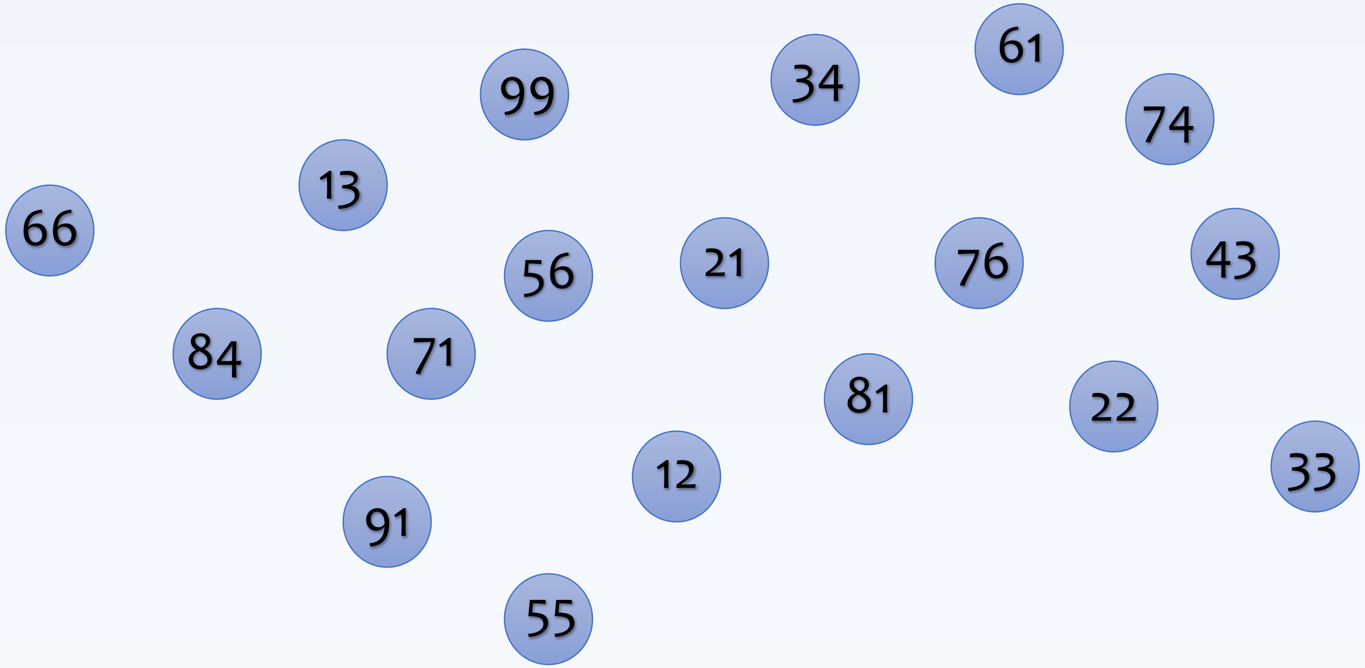
UCDAVIS
UNIVERSITY OF CALIFORNIA

ExpoLab
Creativity Unfolded

ResilientDB

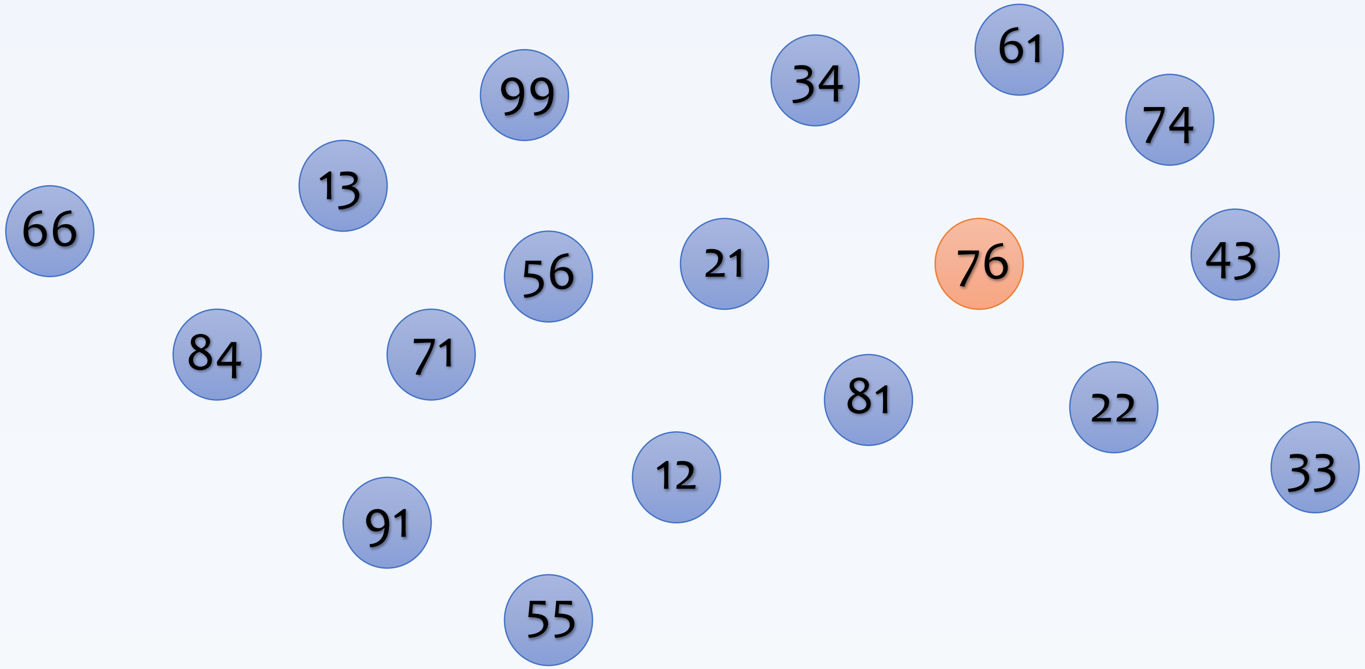
How to quickly search for the desired information?



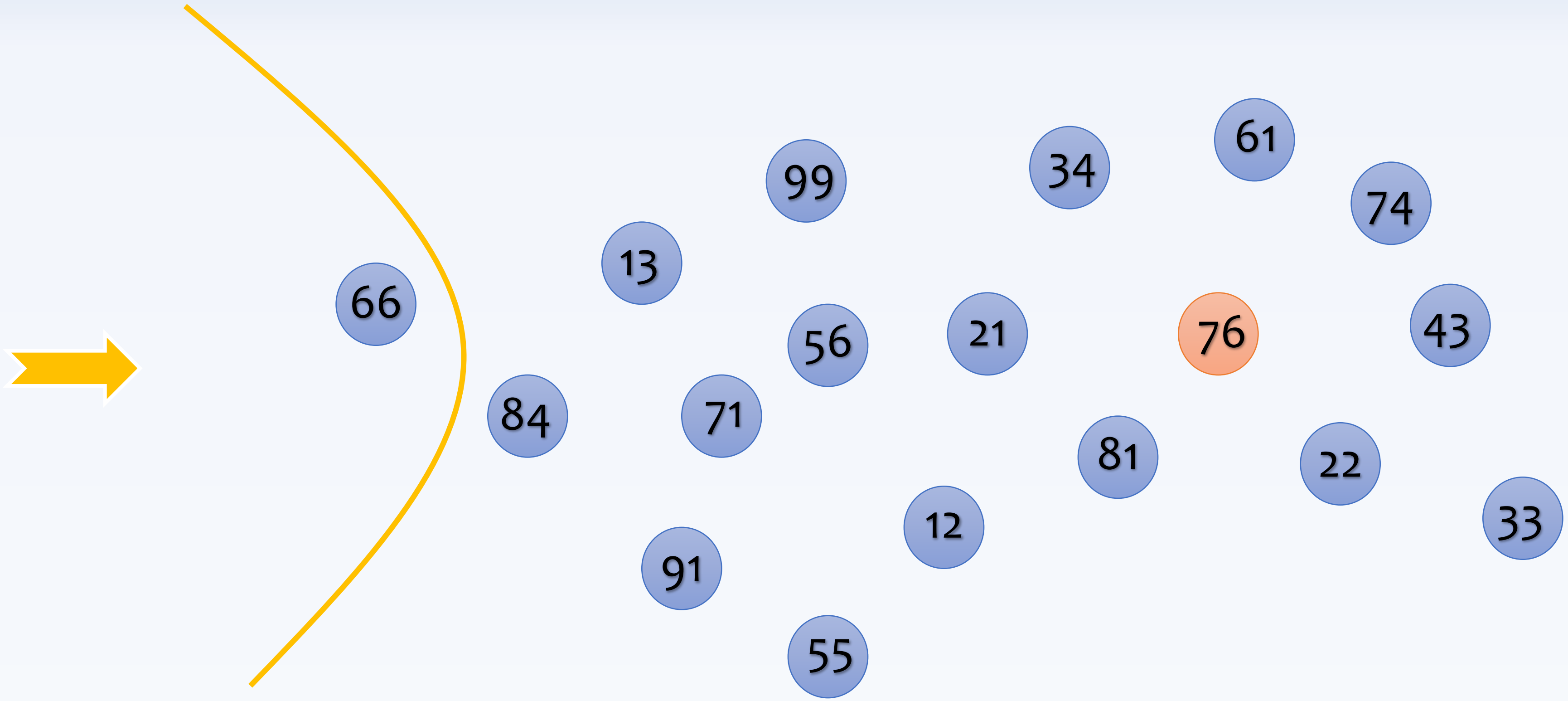


61
34
99
74
13
66
43
21
76
56
84
71
81
22
33
12
91
55

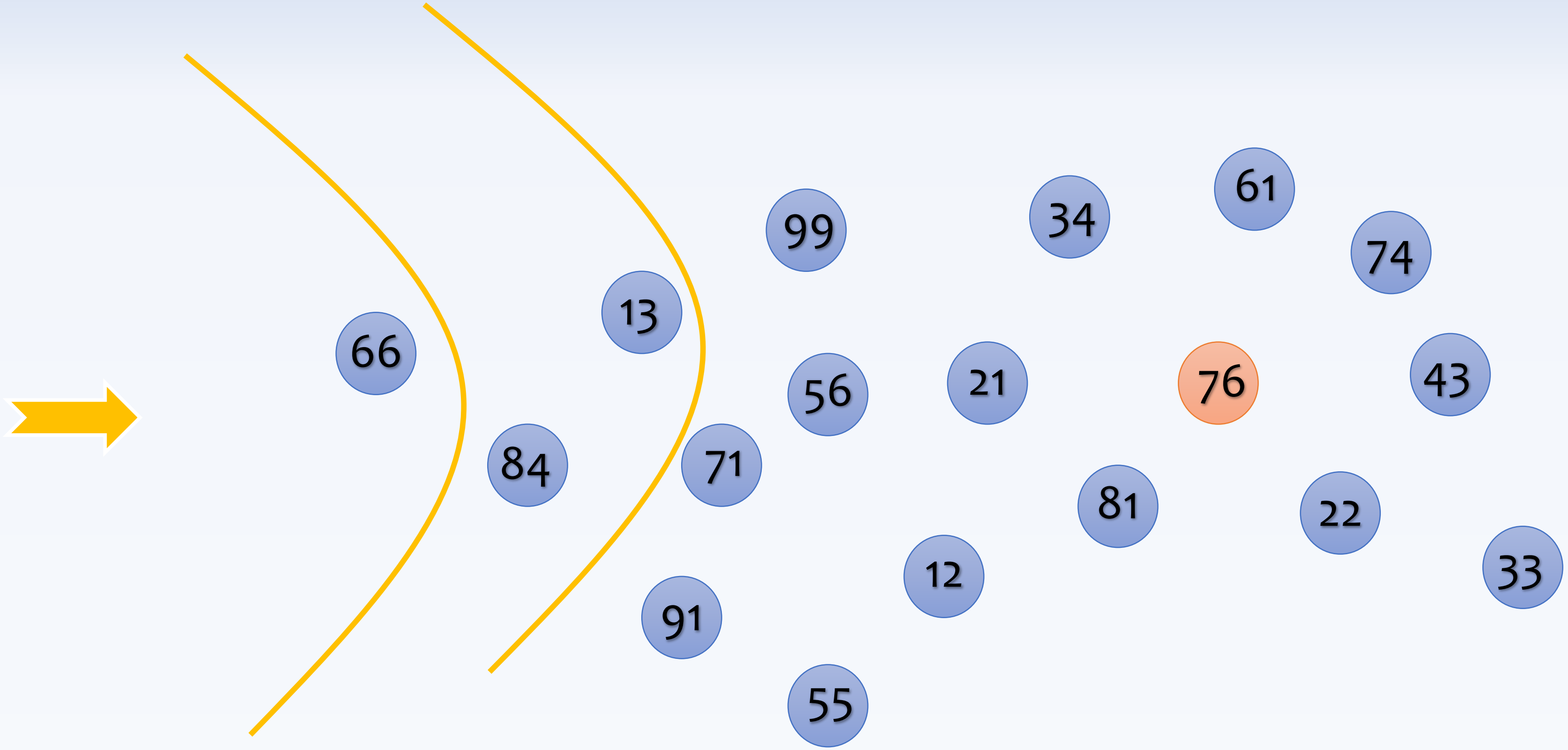
Searching for 76



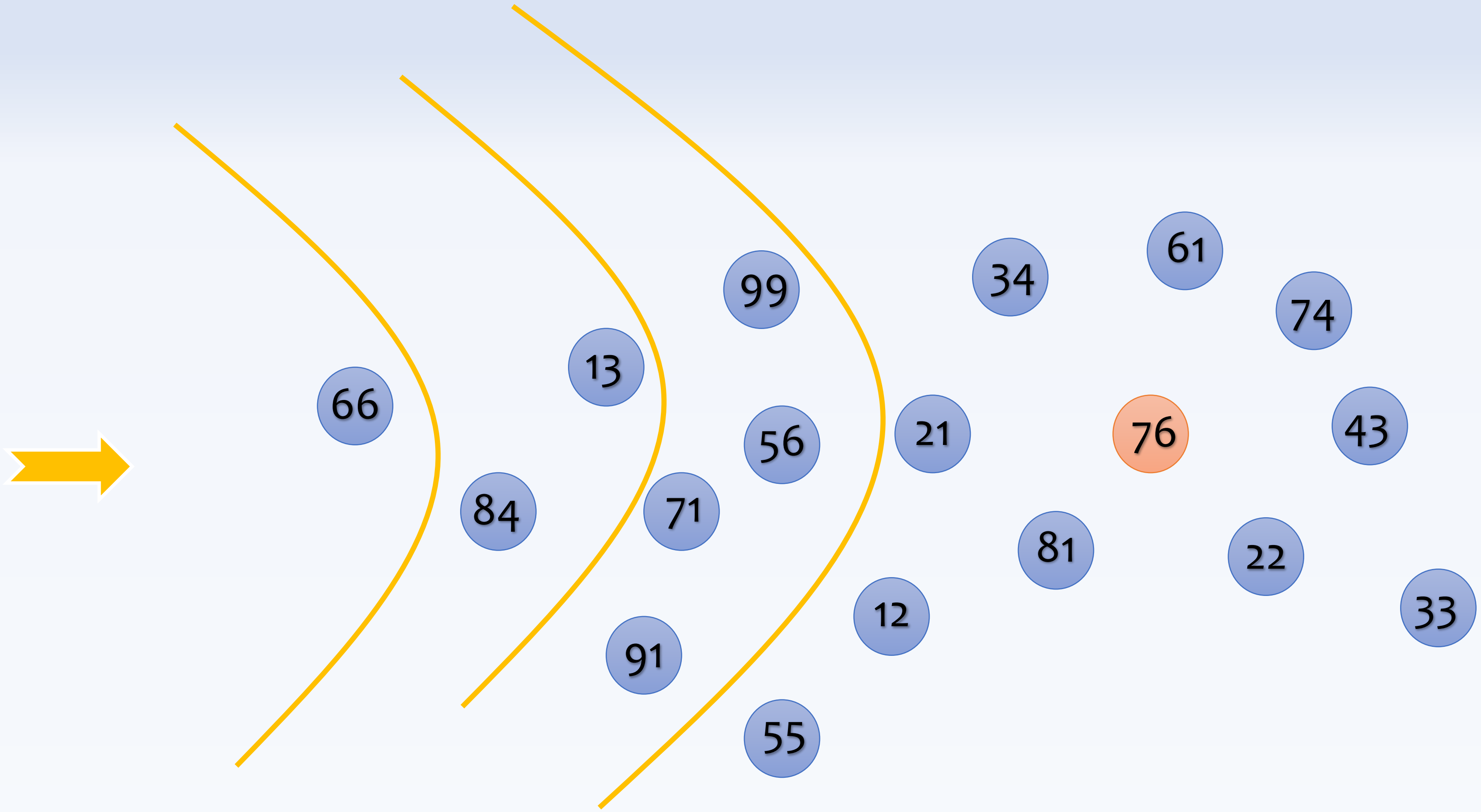
Searching for 76



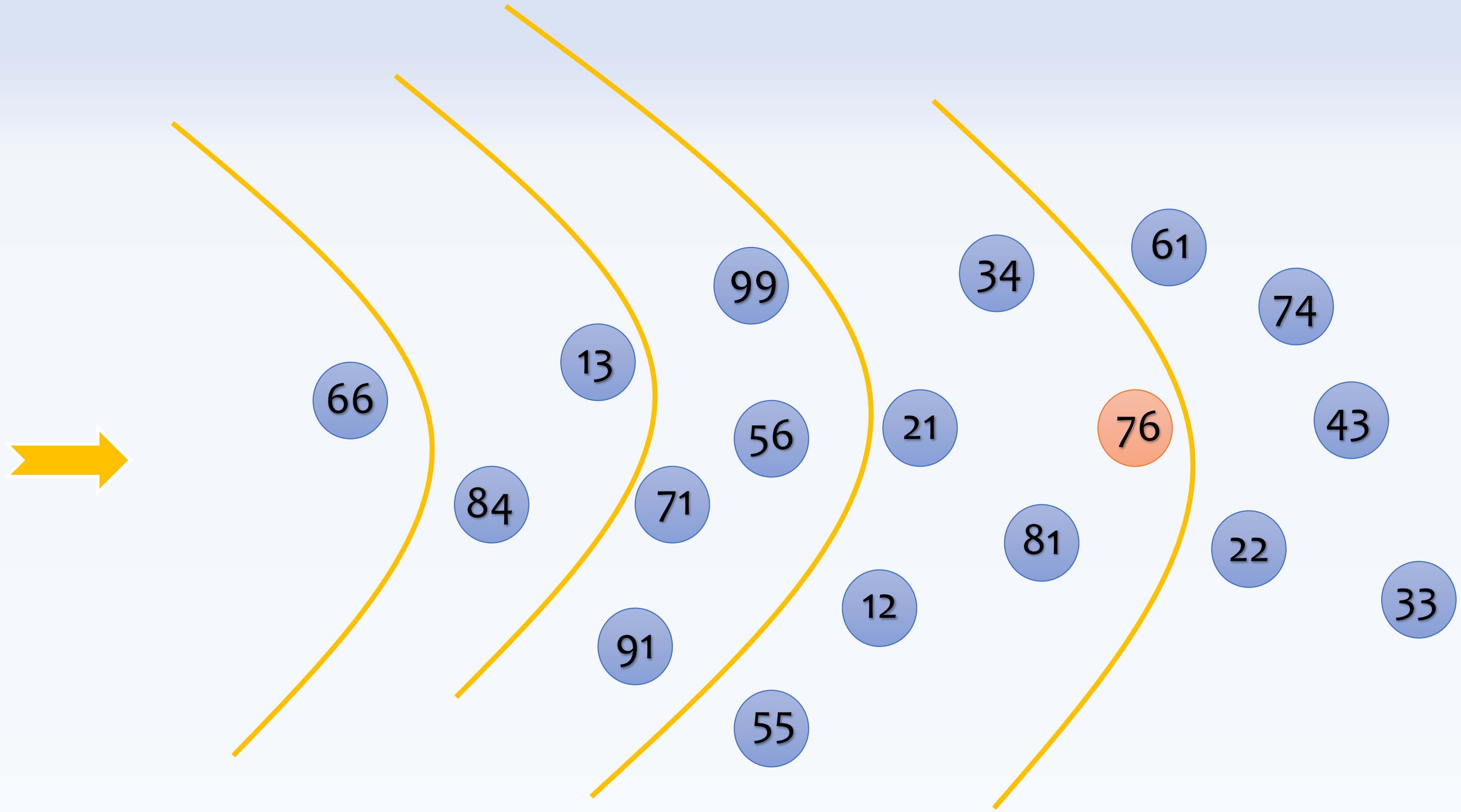
Searching for 76



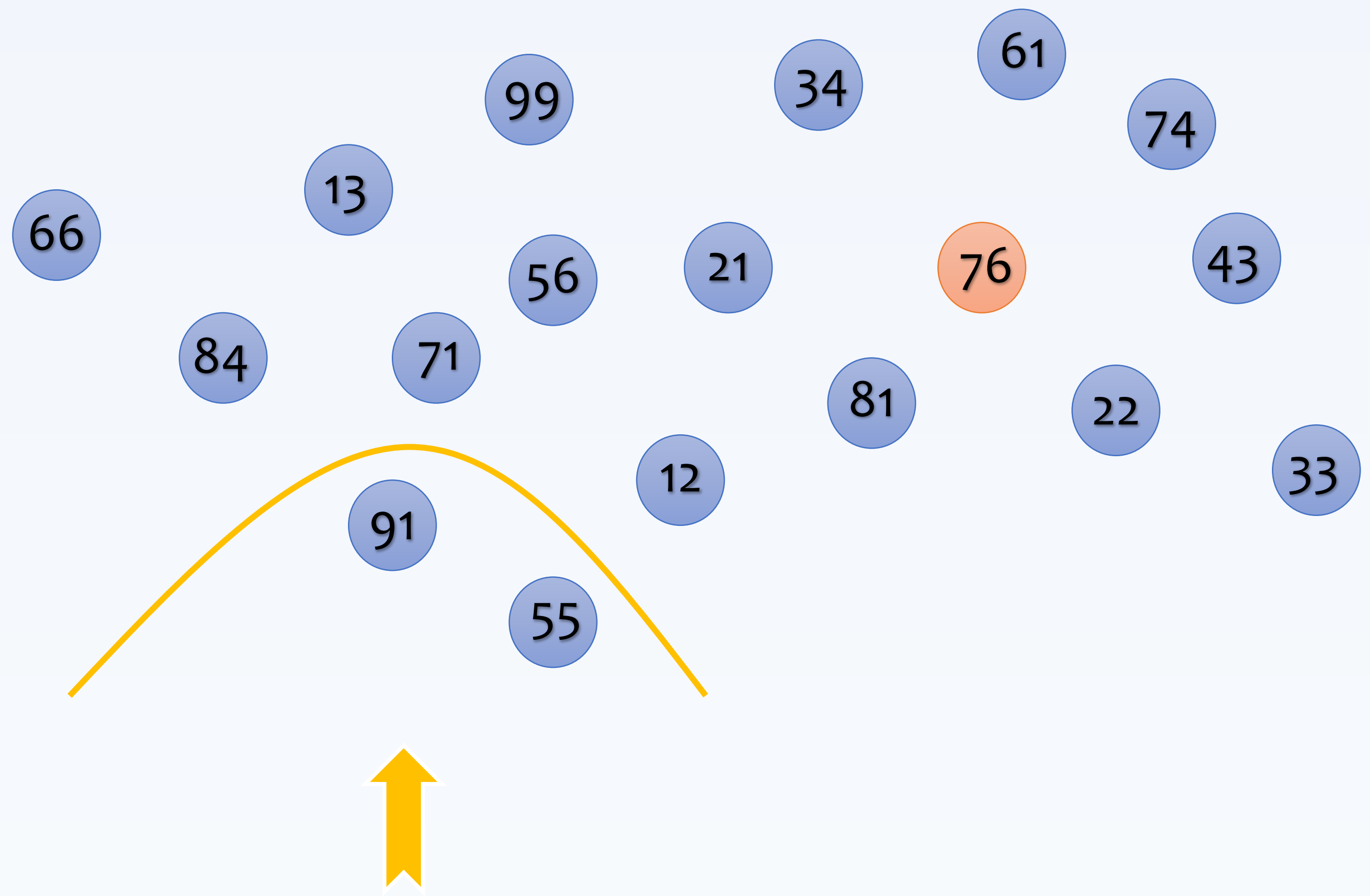
Searching for 76



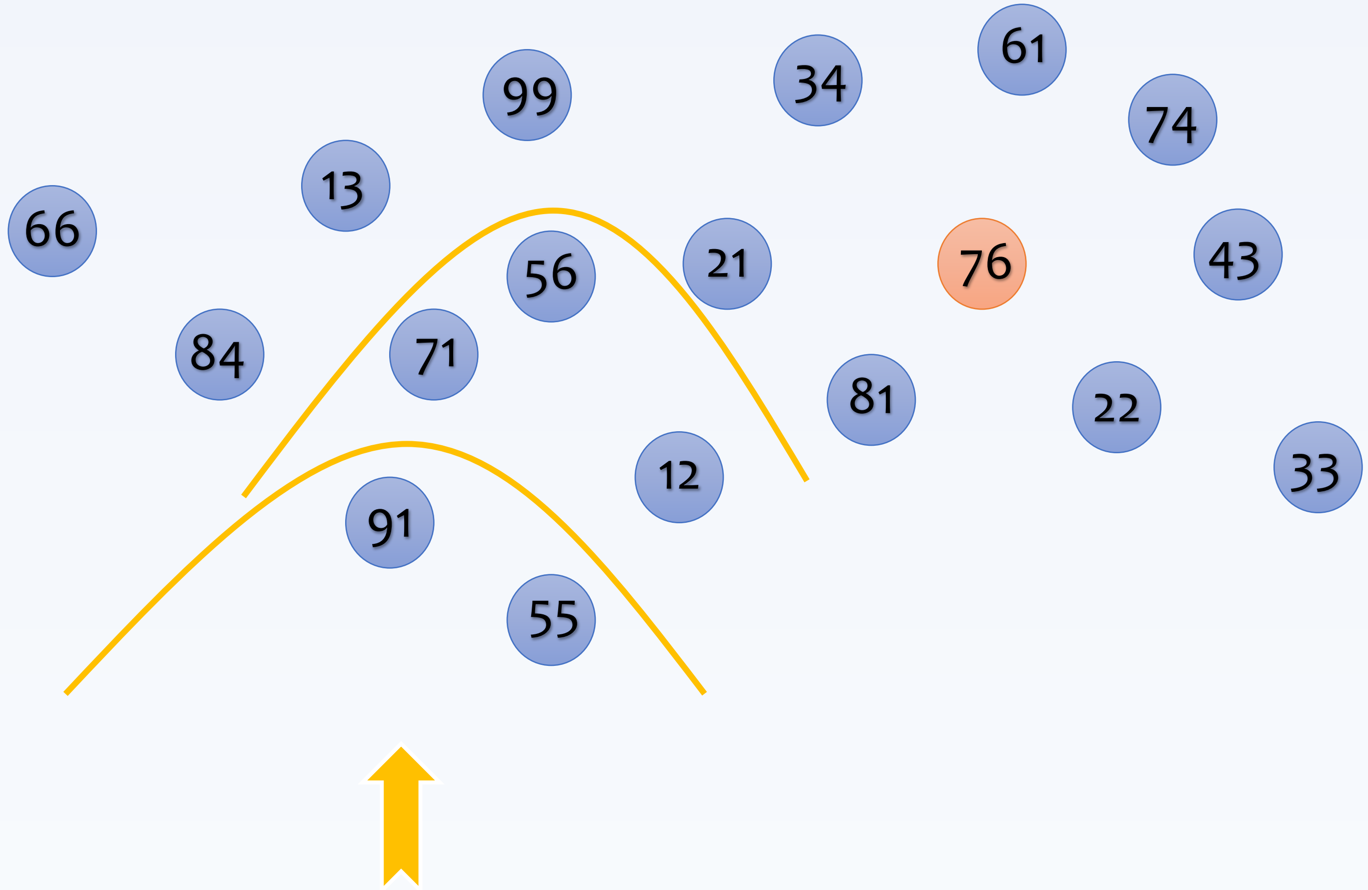
Searching for 76



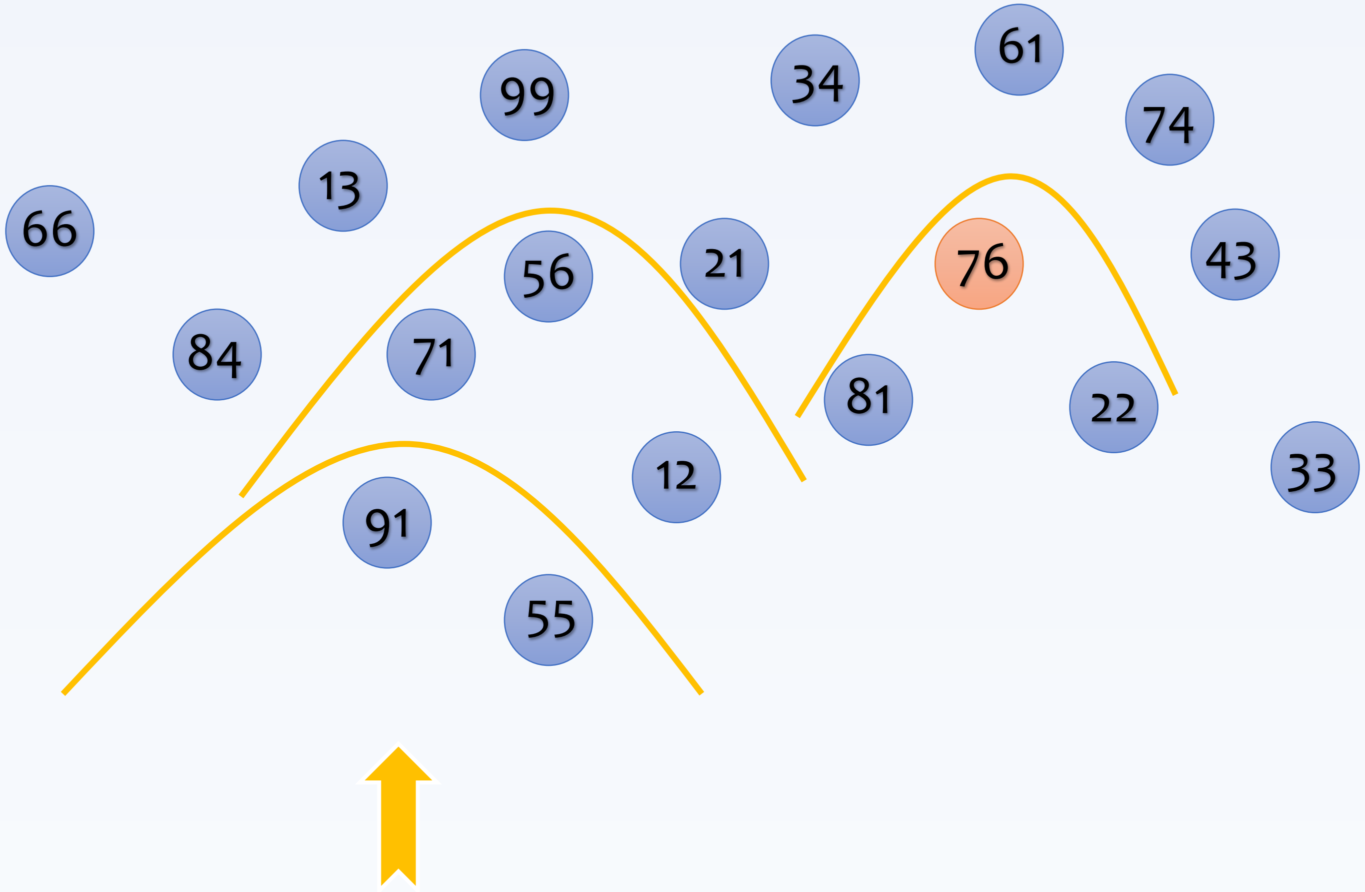
Searching for 76



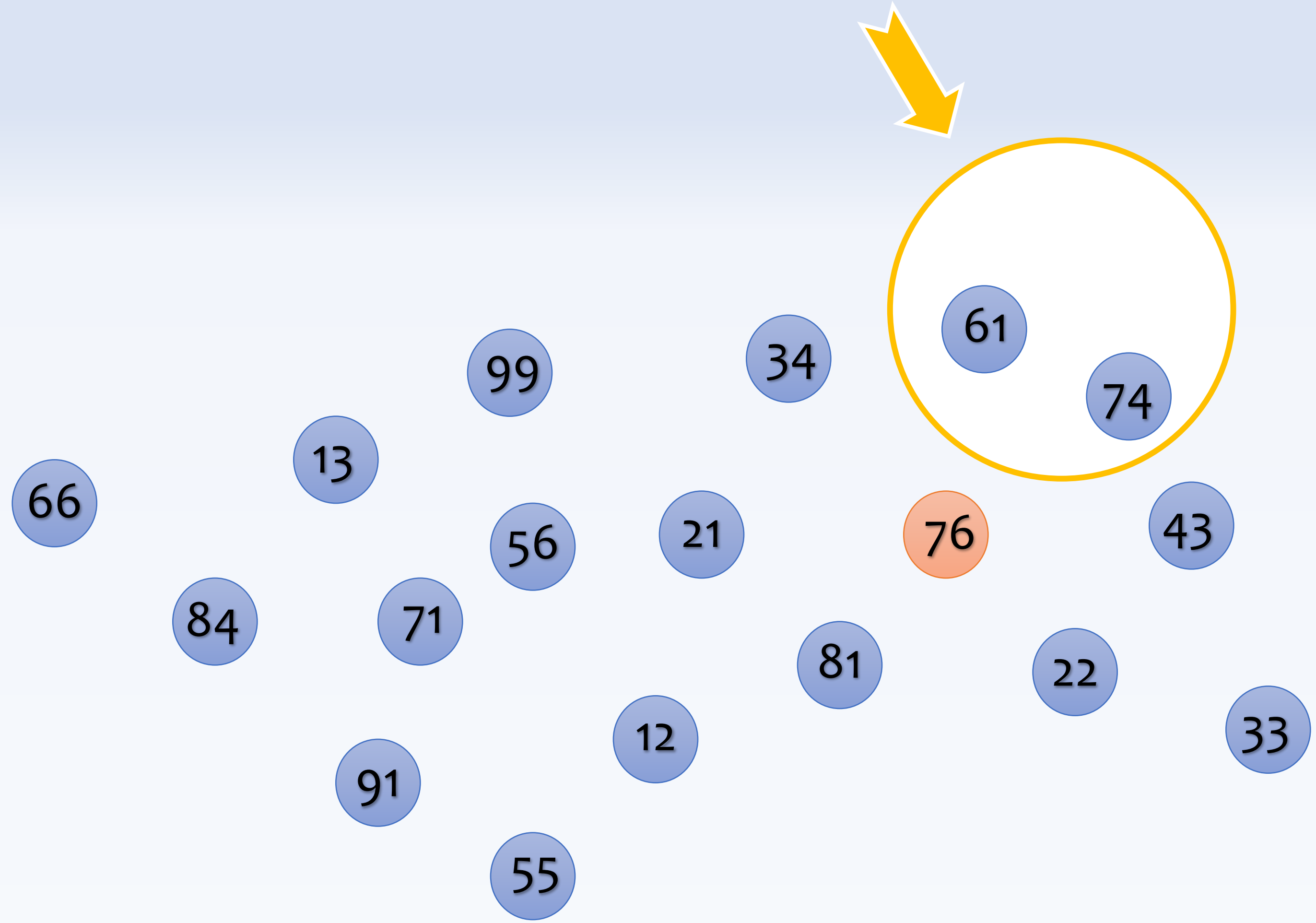
Searching for 76



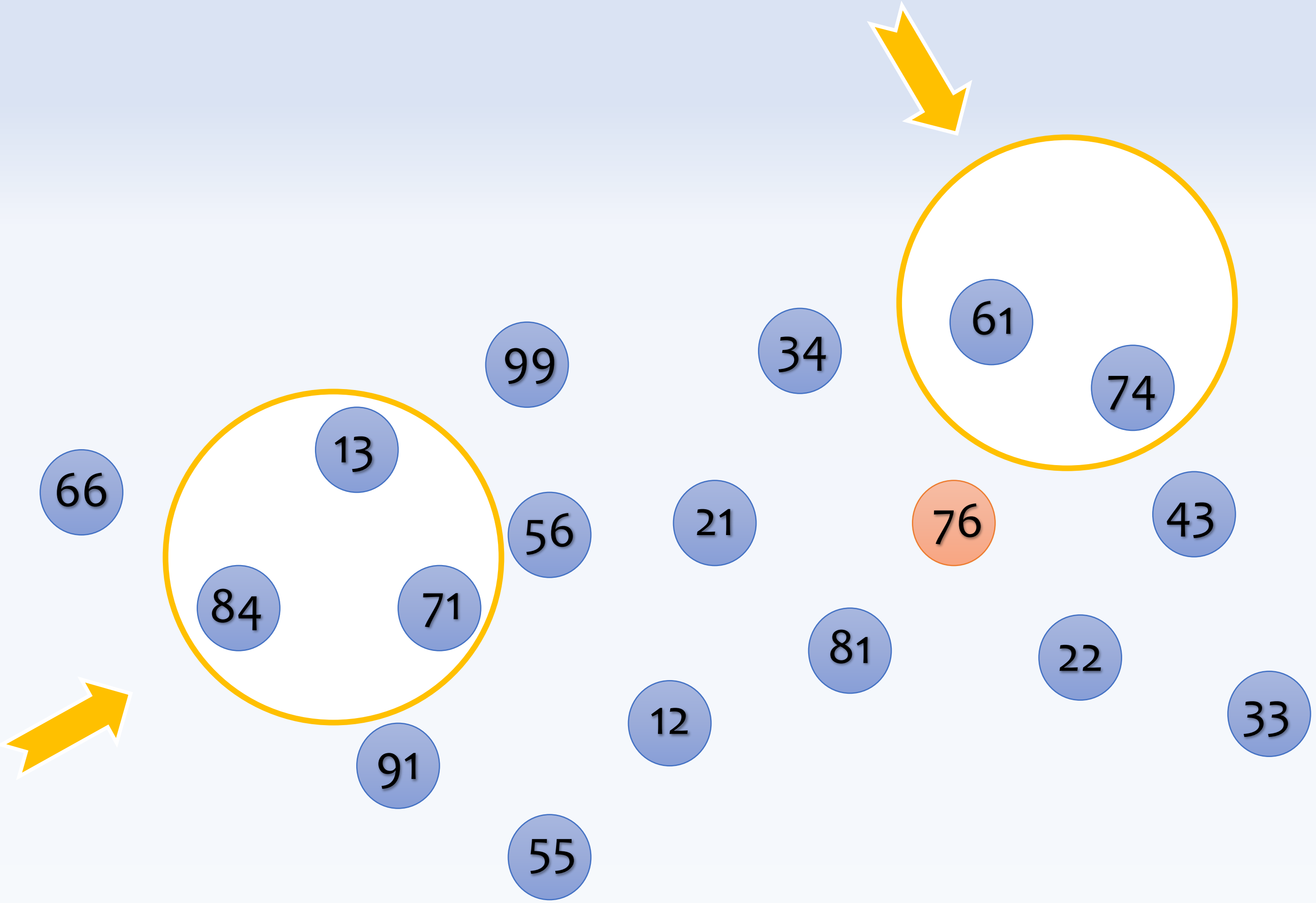
Searching for 76



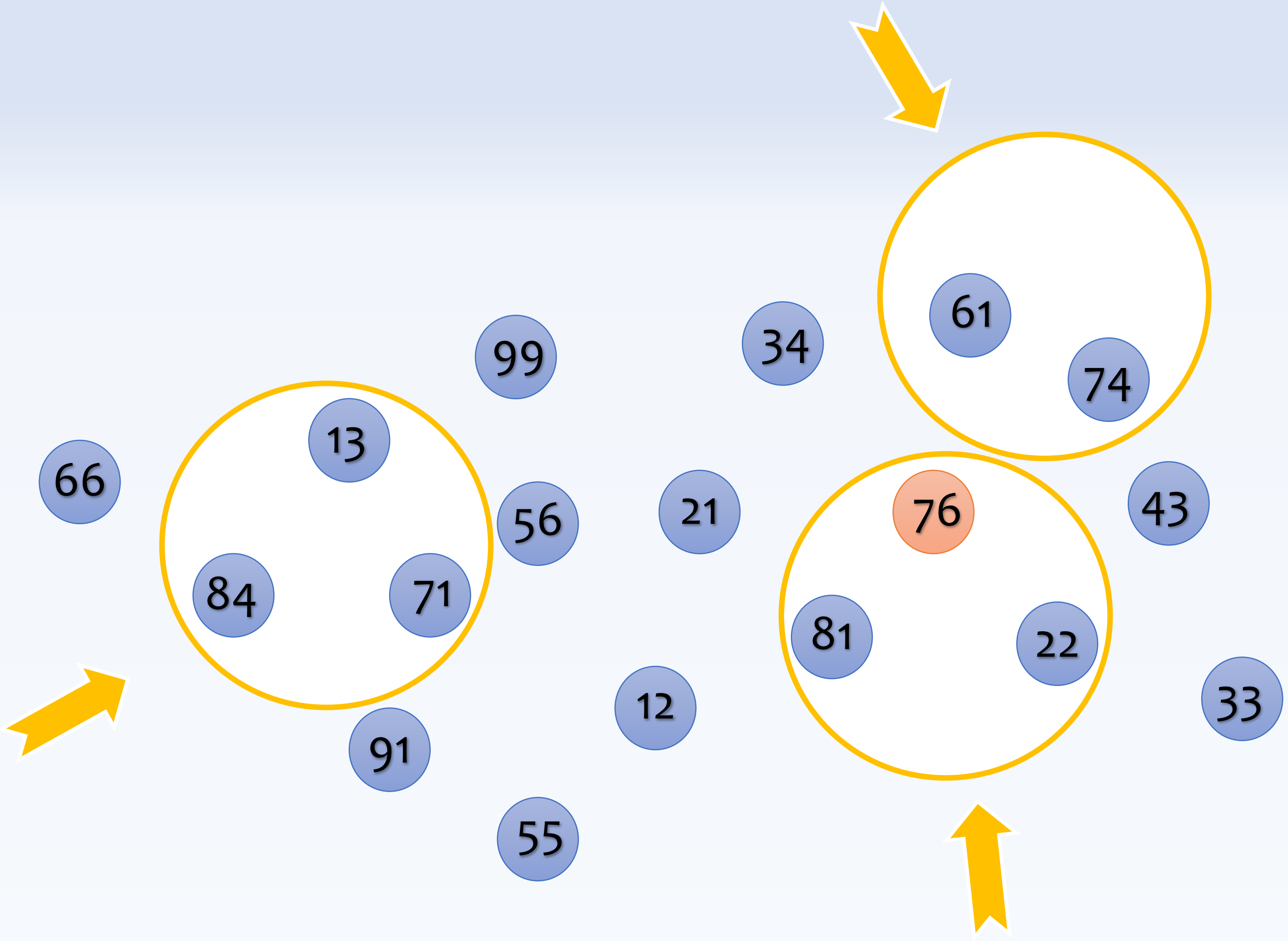
Searching for 76



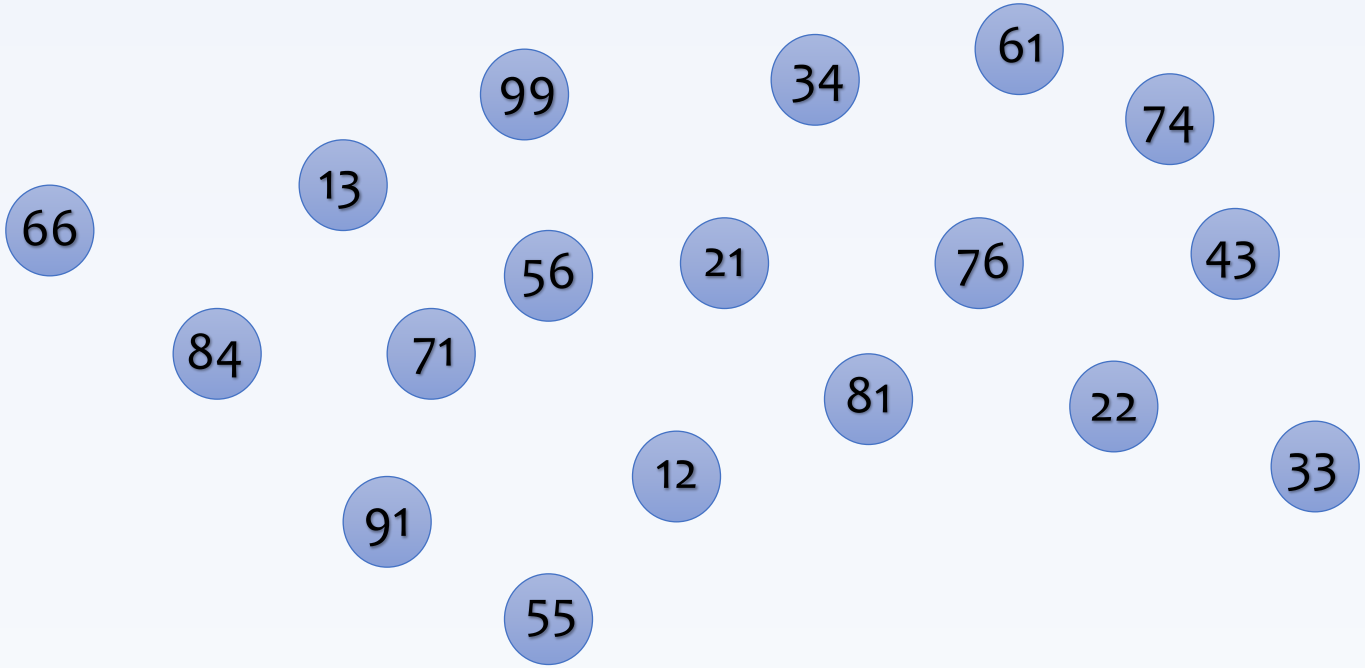
Searching for 76



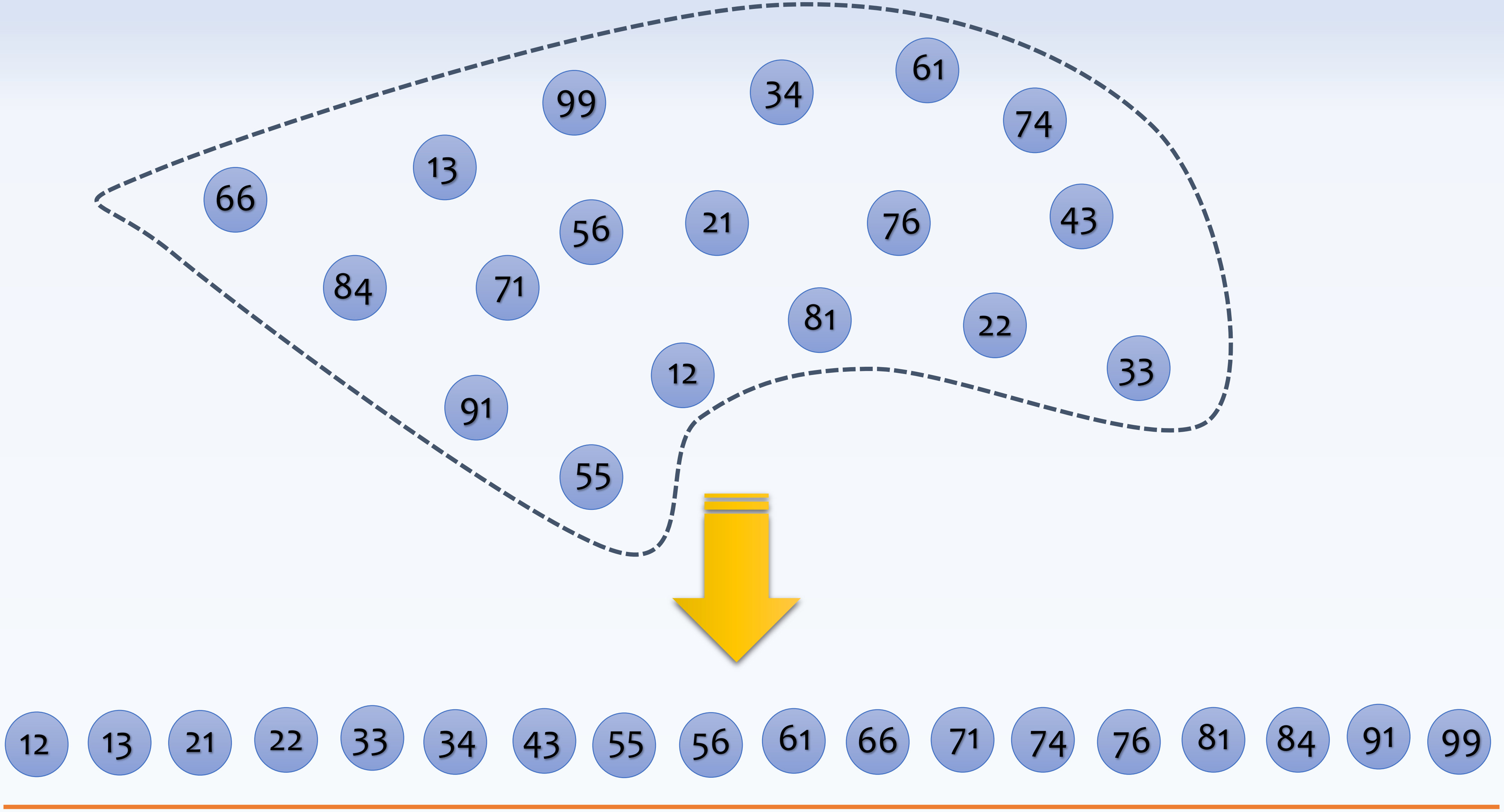
Searching for 76



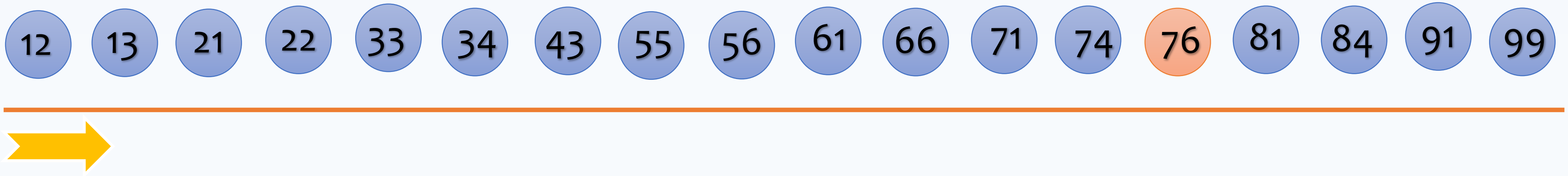
Searching for 44?
(what-if the value does not exist)
(could we have an early termination?)



Could we impose an order to improve the search?

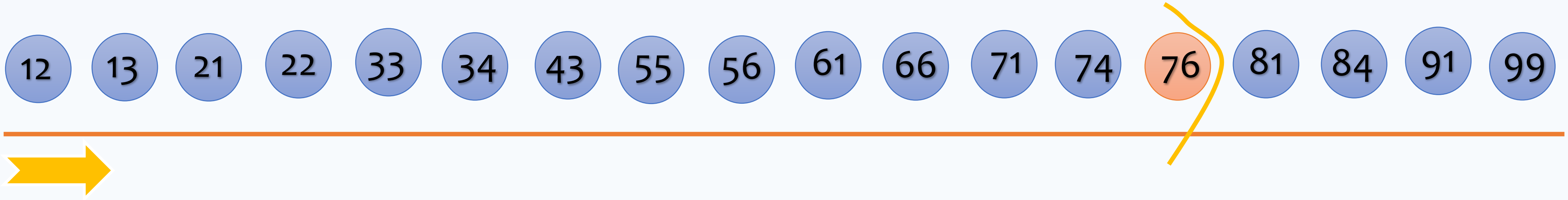

61
34
99
74
13
66
43
56
21
76
84
71
81
22
33
12
91
55
12
13
21
22
33
34
43
55
56
61
66
71
74
76
81
84
91
99

Searching for 76



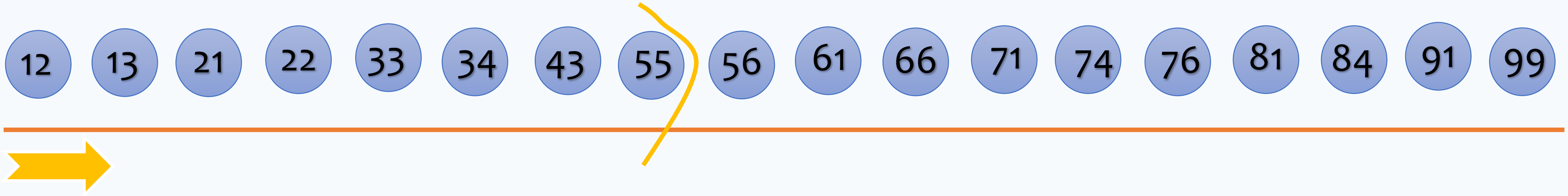
12 13 21 22 33 34 43 55 56 61 66 71 74 76 81 84 91 99

Searching for 76



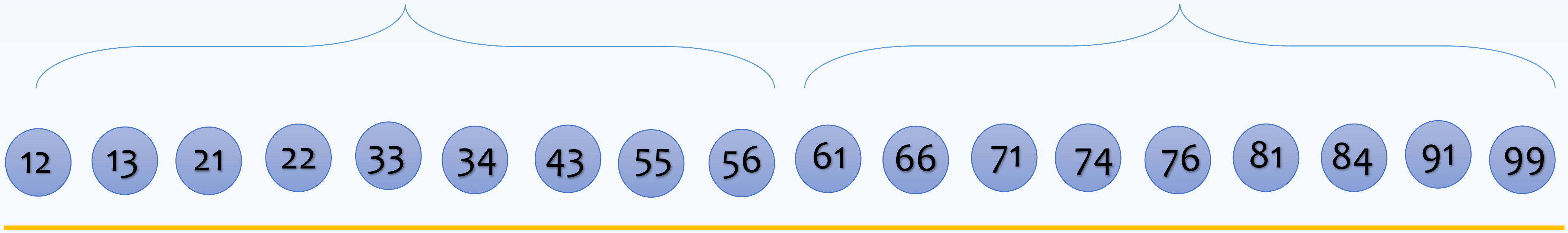
12 13 21 22 33 34 43 55 56 61 66 71 74 76 81 84 91 99

Searching for 44?
(could we have an early termination?)

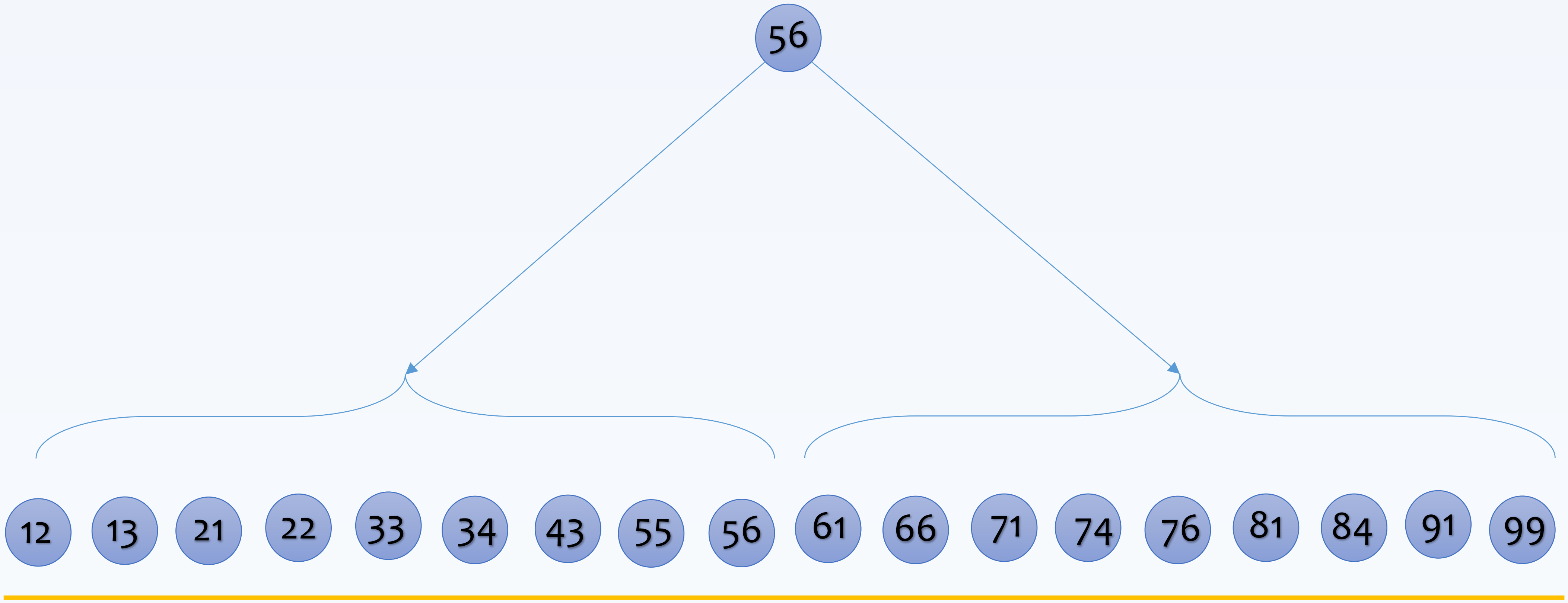


12 13 21 22 33 34 43 55 56 61 66 71 74 76 81 84 91 99

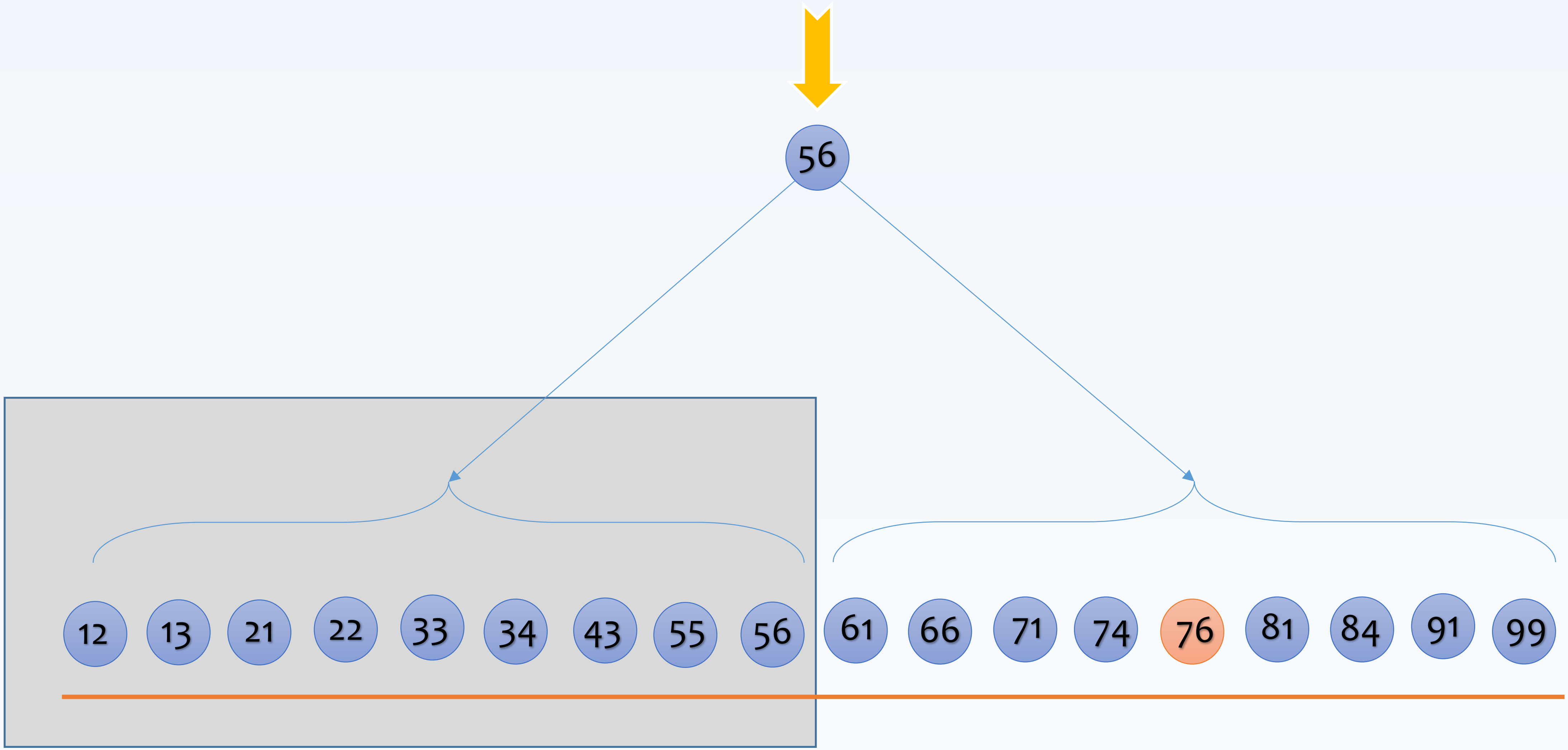
Could we impose a structure to further improve the search?



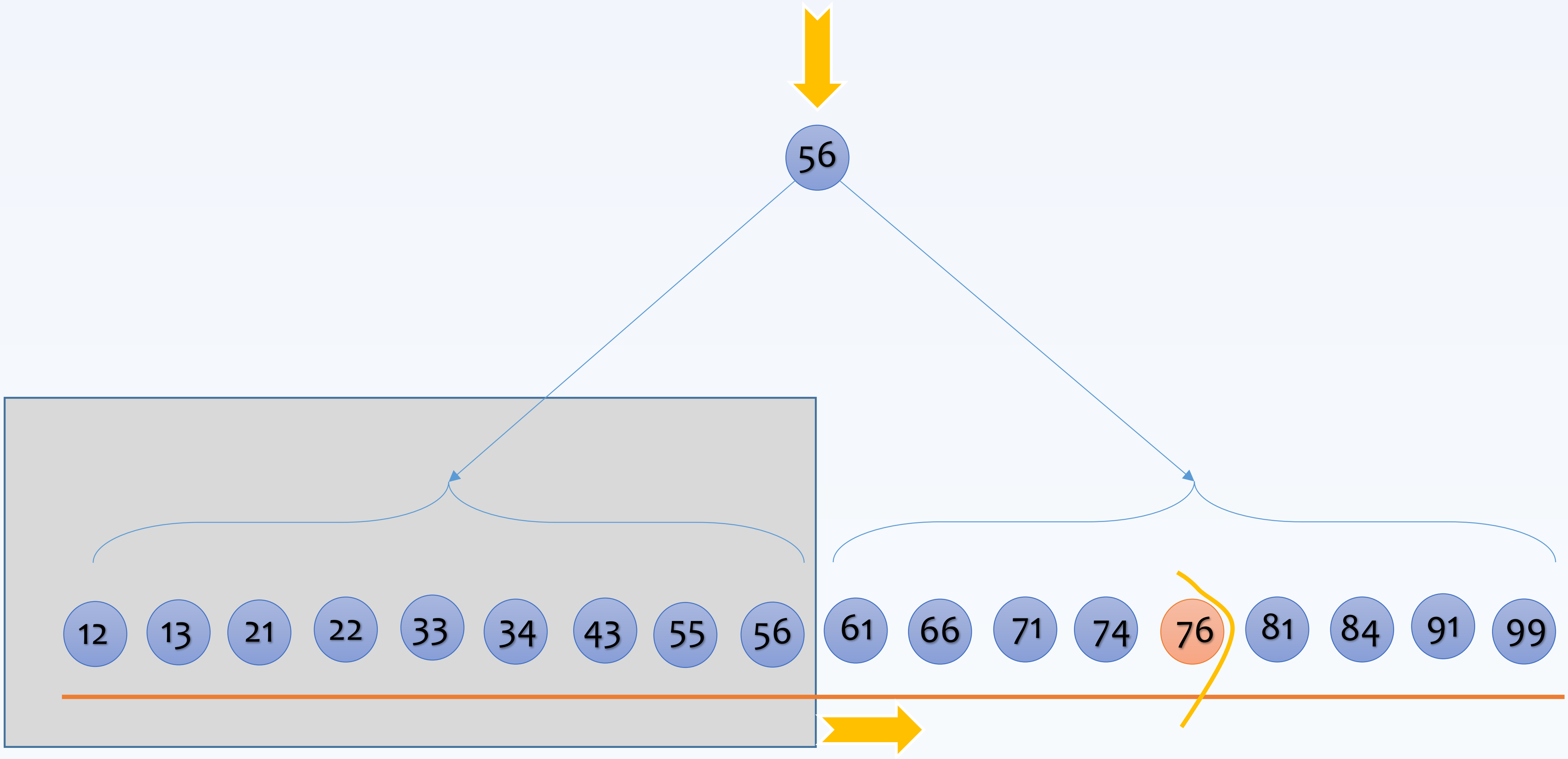
12 13 21 22 33 34 43 55 56 61 66 71 74 76 81 84 91 99

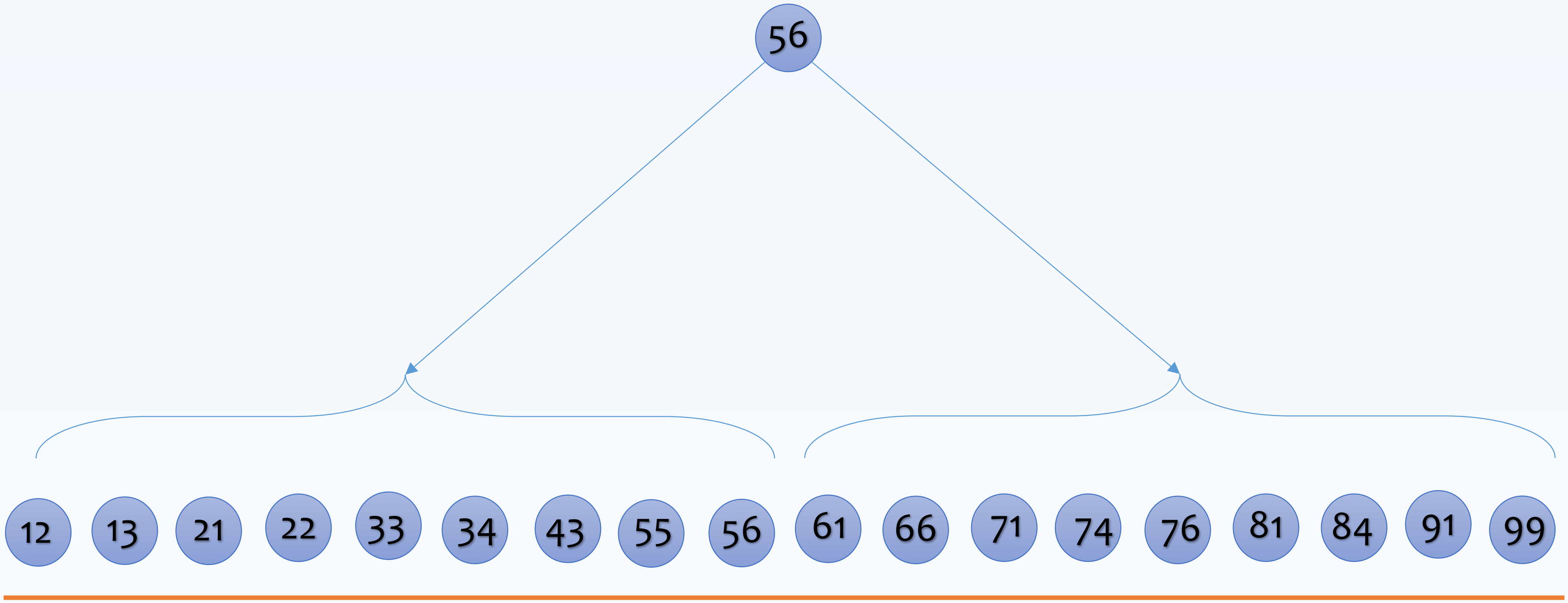

56
12
13
21
22
33
34
43
55
56
61
66
71
74
76
81
84
91
99

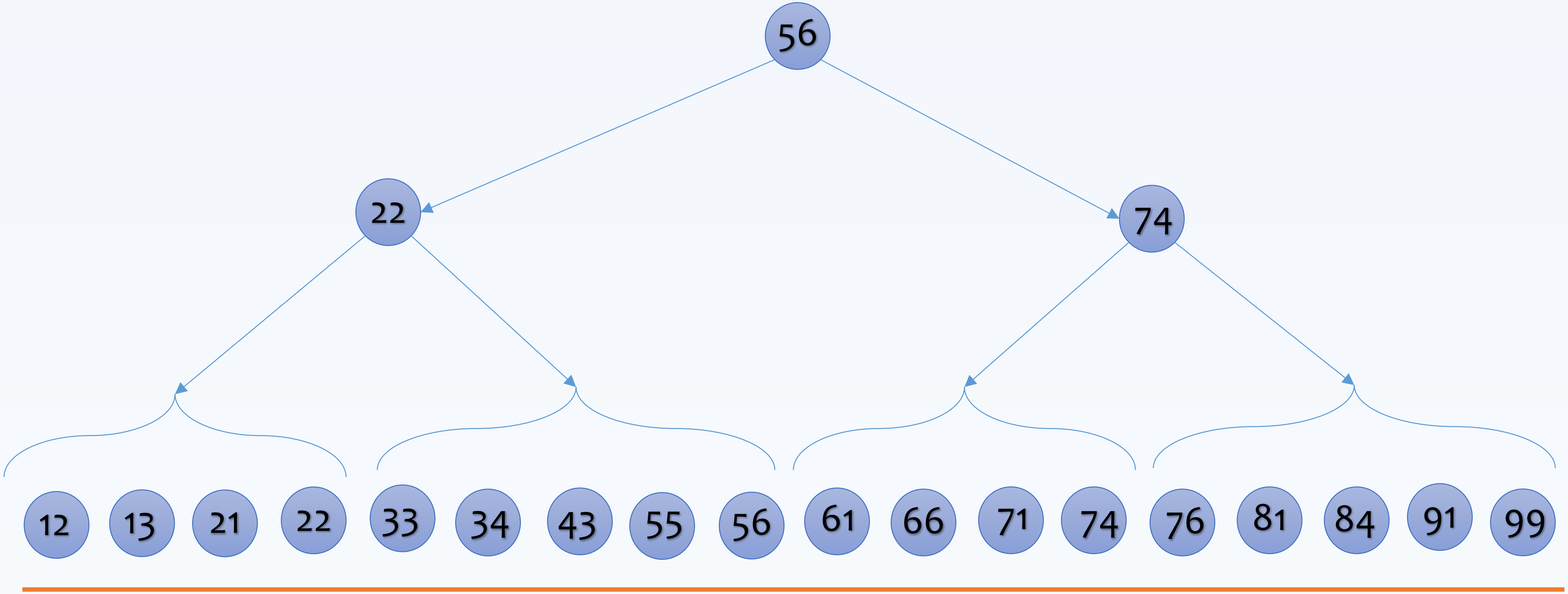
Searching for 76



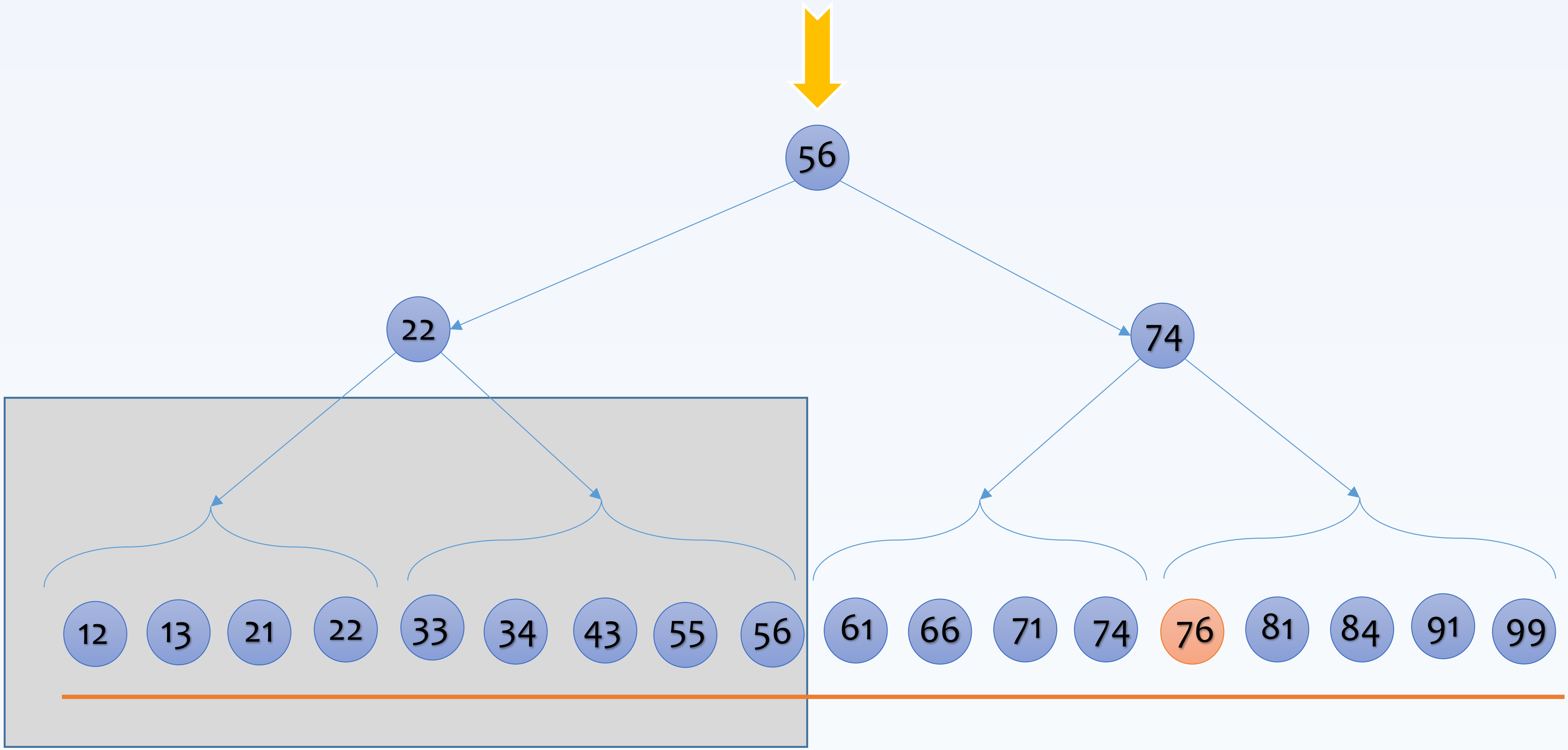
Searching for 76



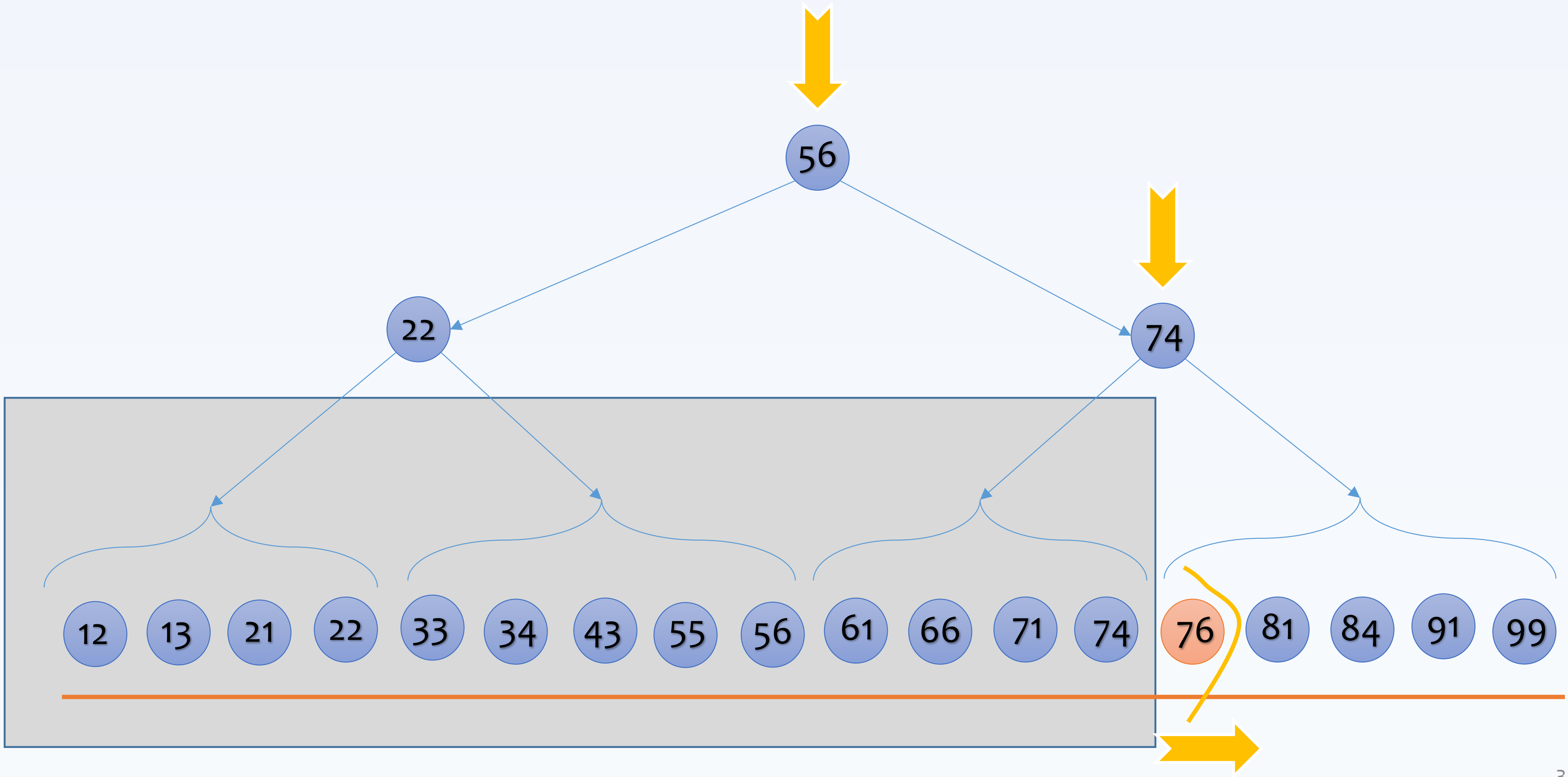

56
12
13
21
22
33
34
43
55
56
61
66
71
74
76
81
84
91
99


56
22
74
12
13
21
22
33
34
43
55
56
61
66
71
74
76
81
84
91
99

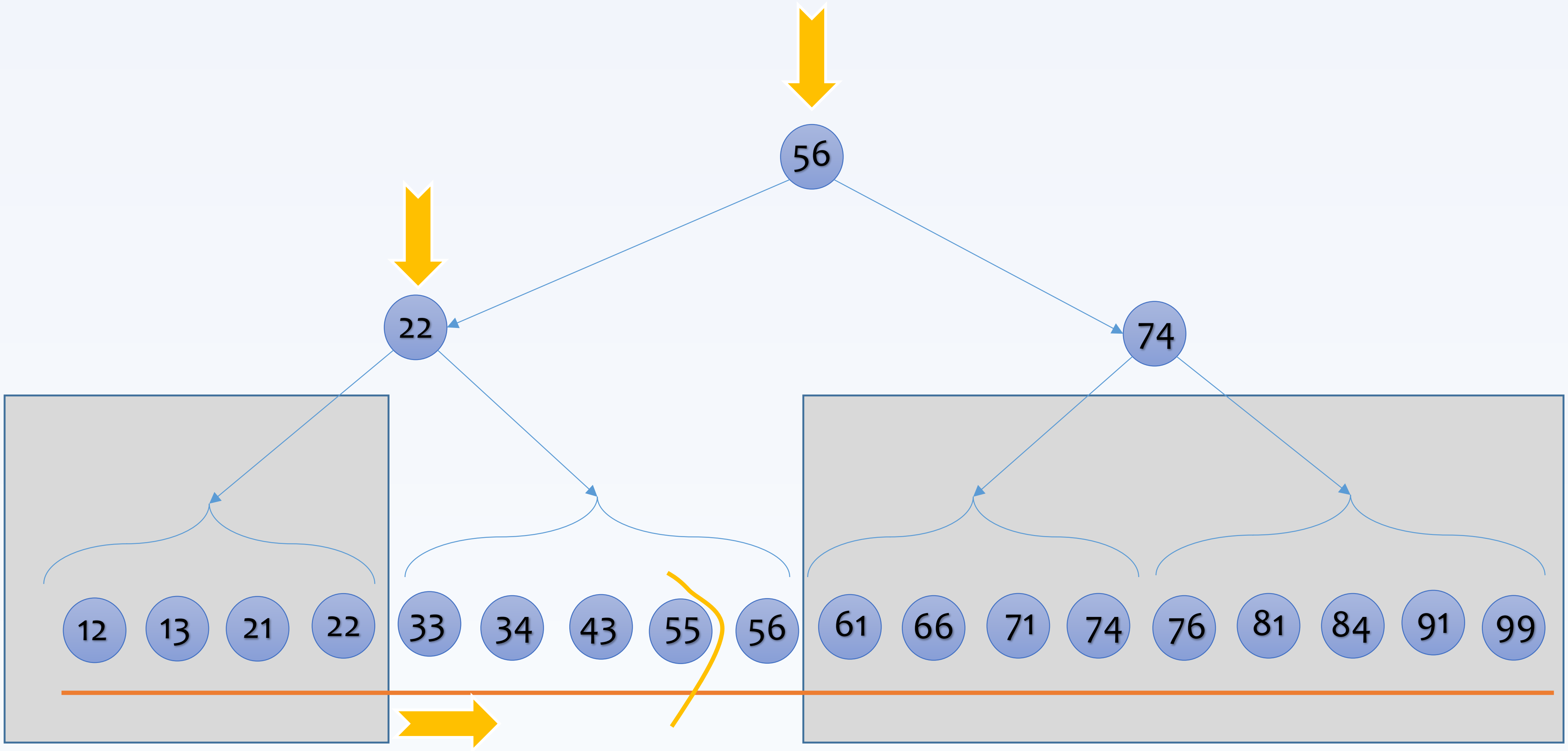
Searching for 76



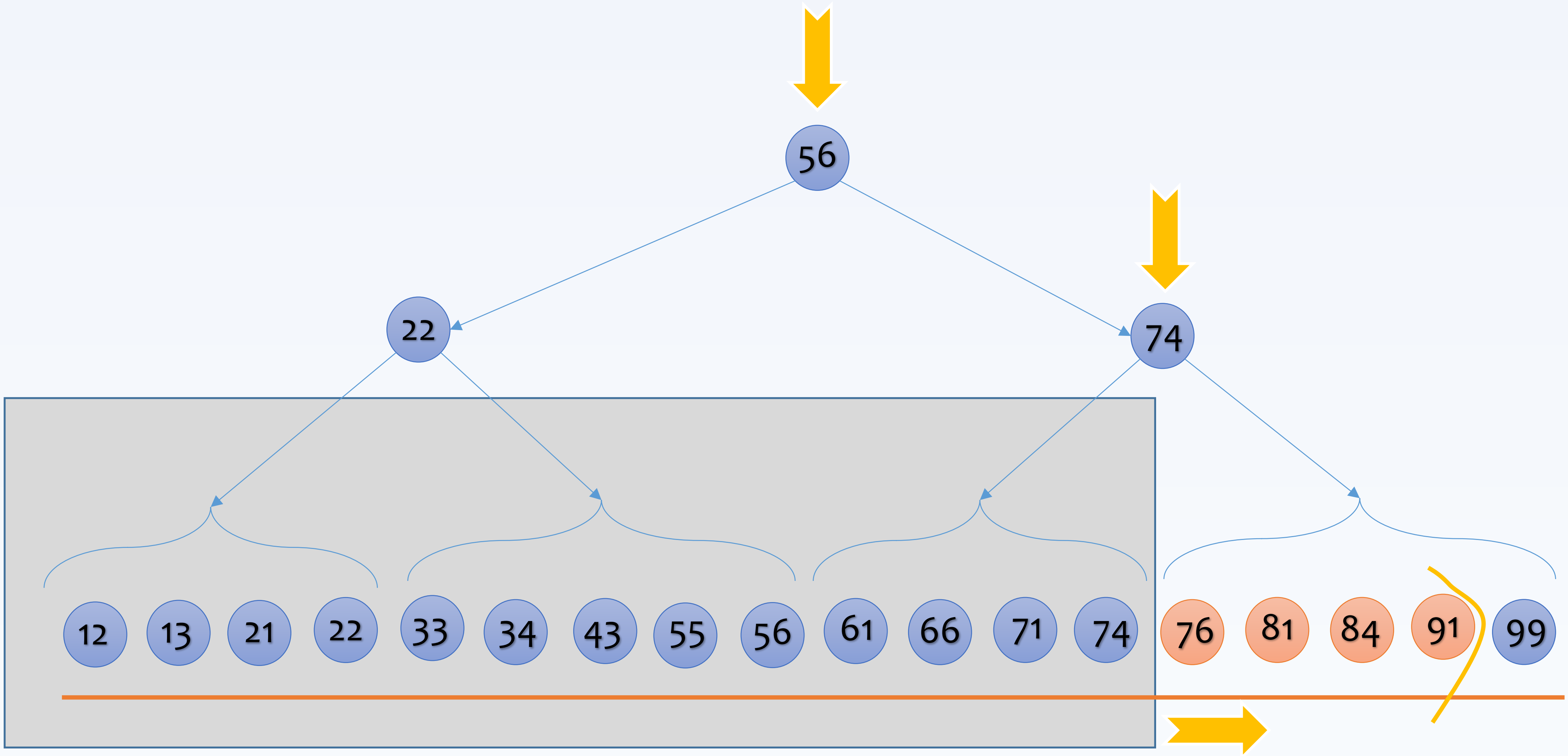
Searching for 76



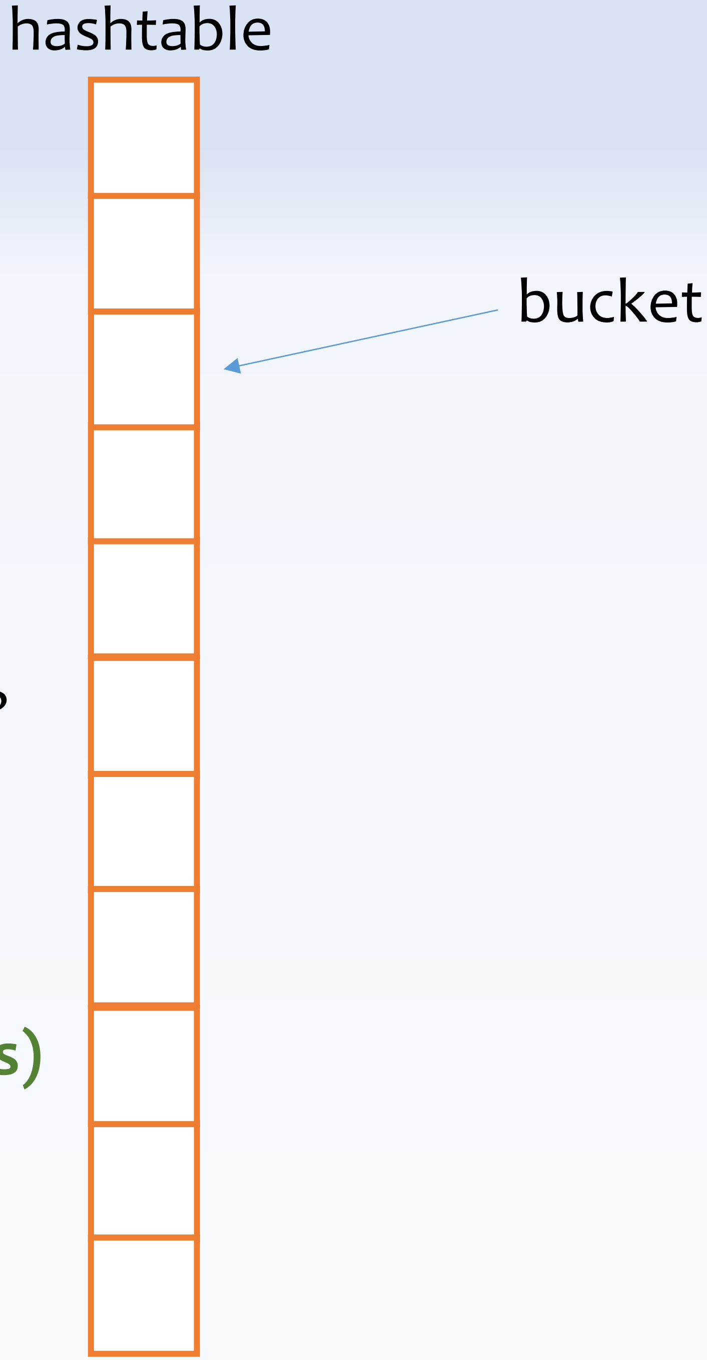
Searching for 44?
(could we have an early termination?)



Searching for 76-91



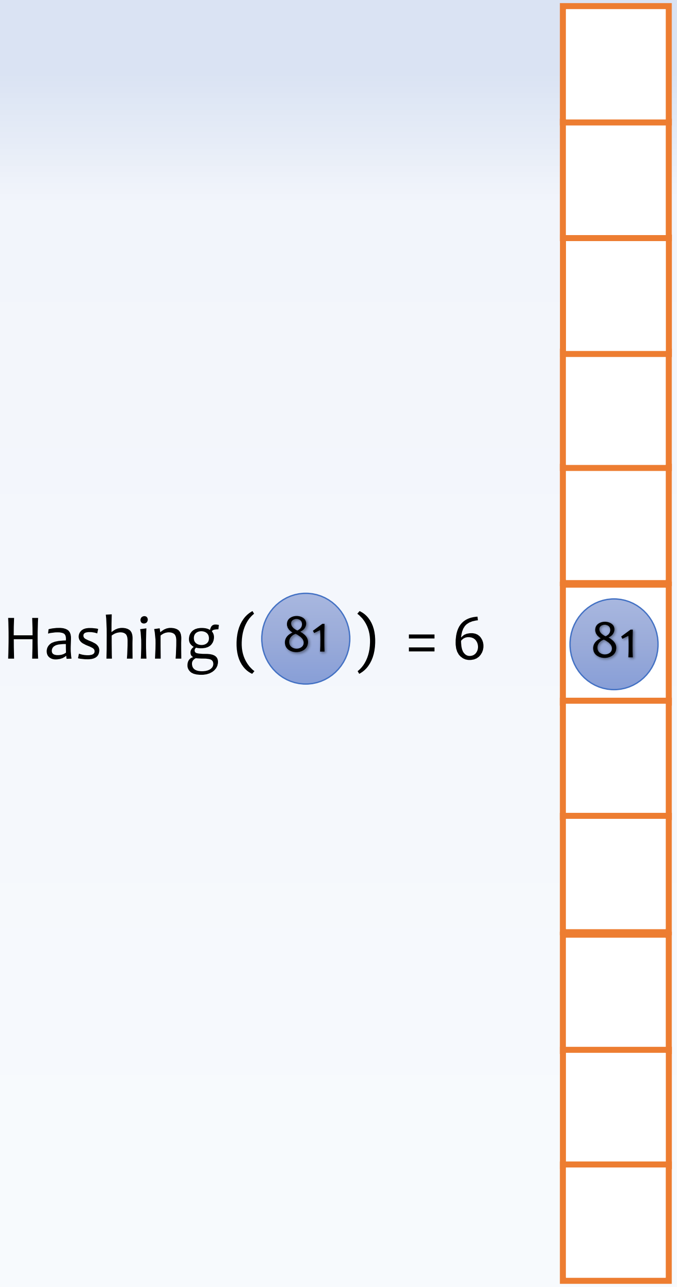
Could we spread the data cleverly to improve the search?



Hashing (●) = ?

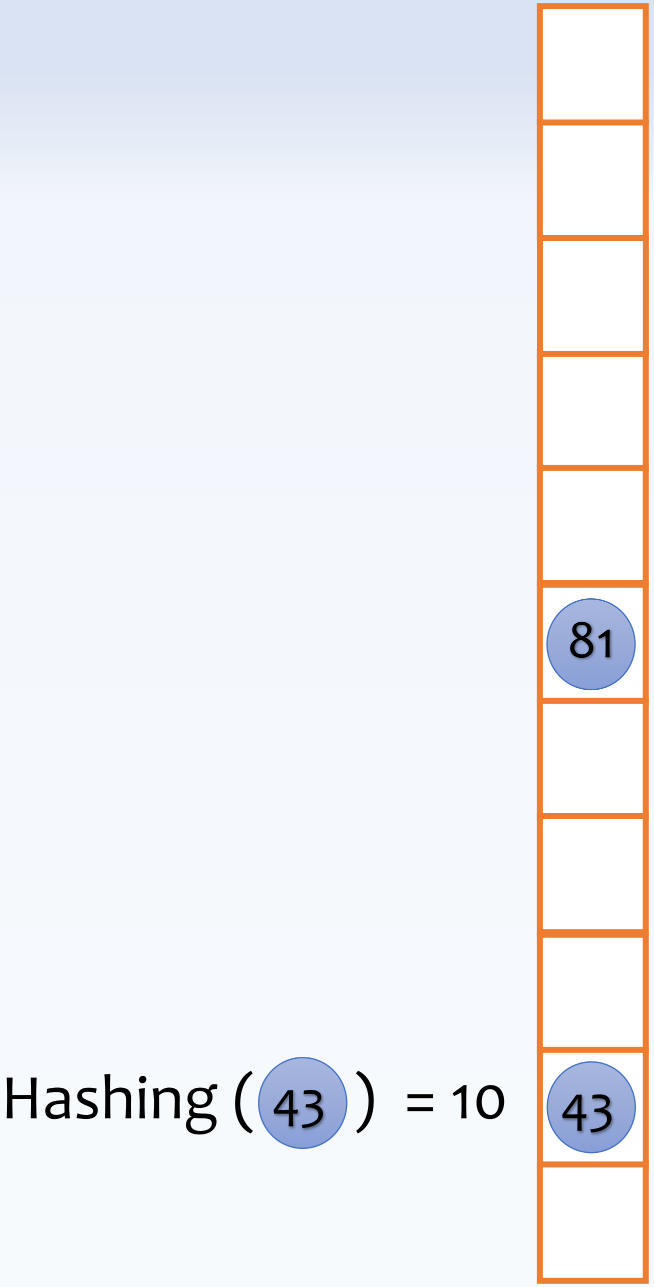
(returns a value between 1 to n, where n is the number of buckets)

Inserting 81



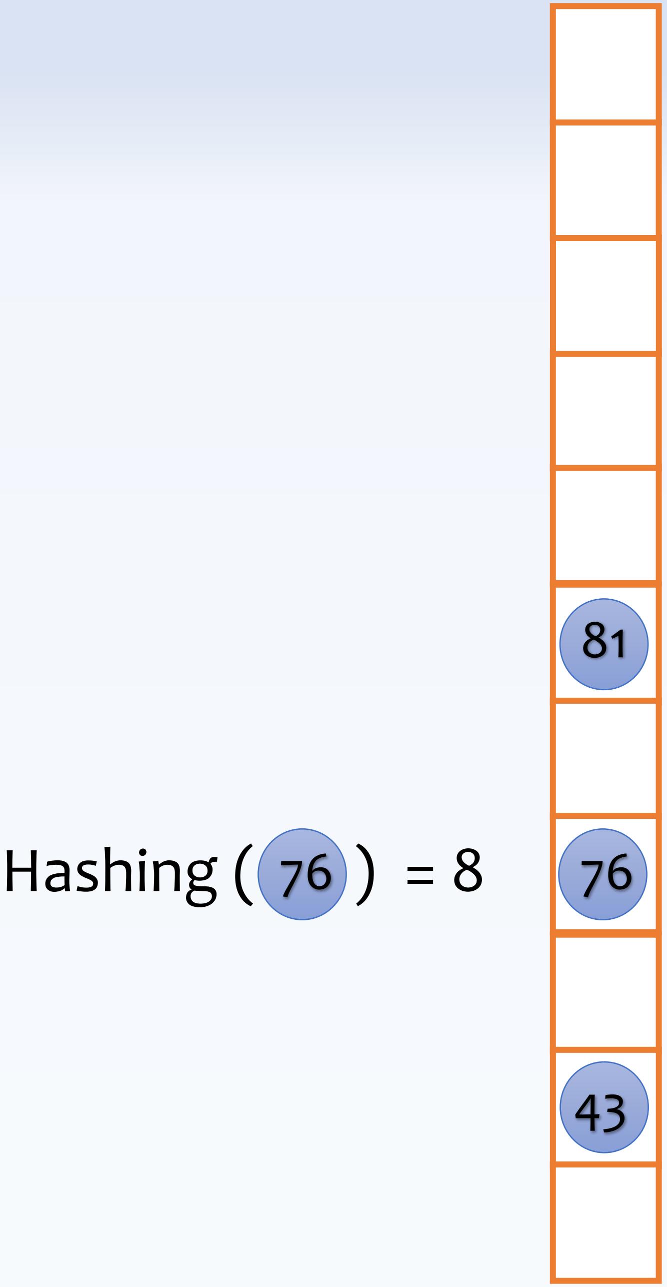
Hashing (81) = 6

Inserting 43

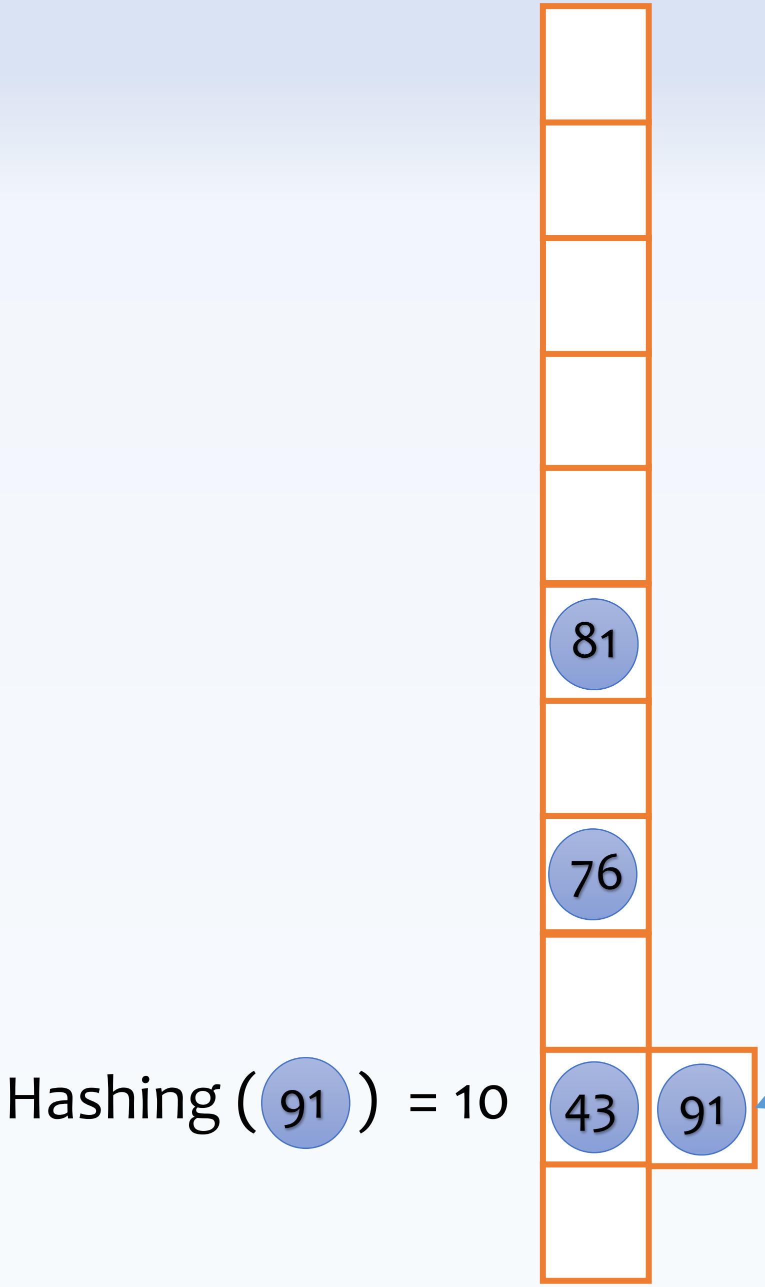


Hashing (43) = 10

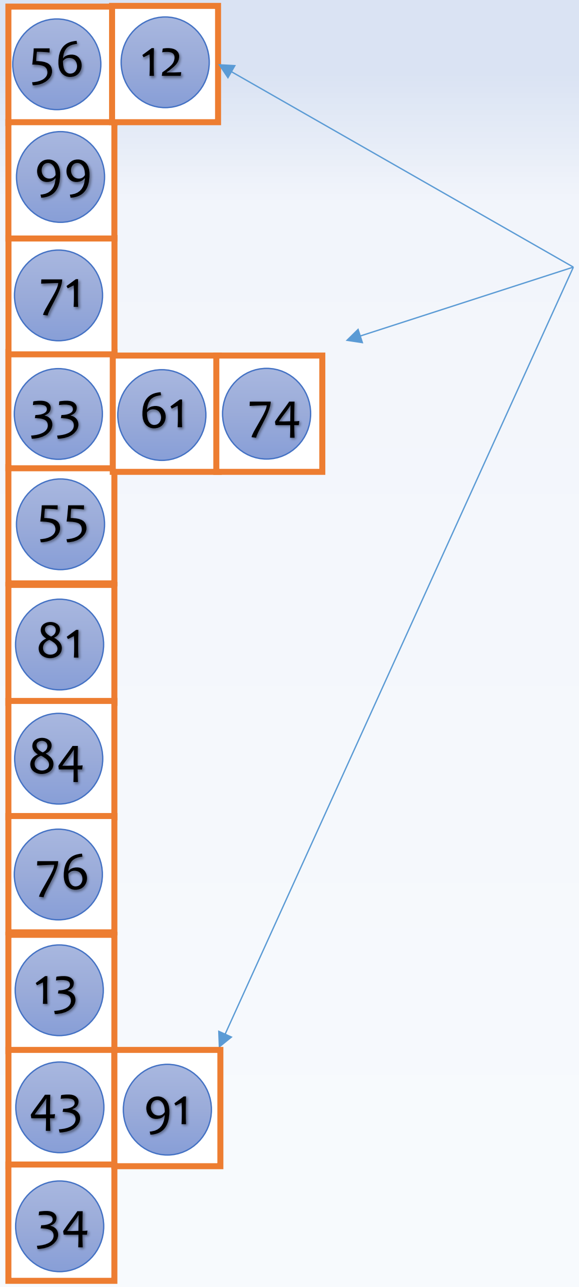
Inserting 76



Inserting 91



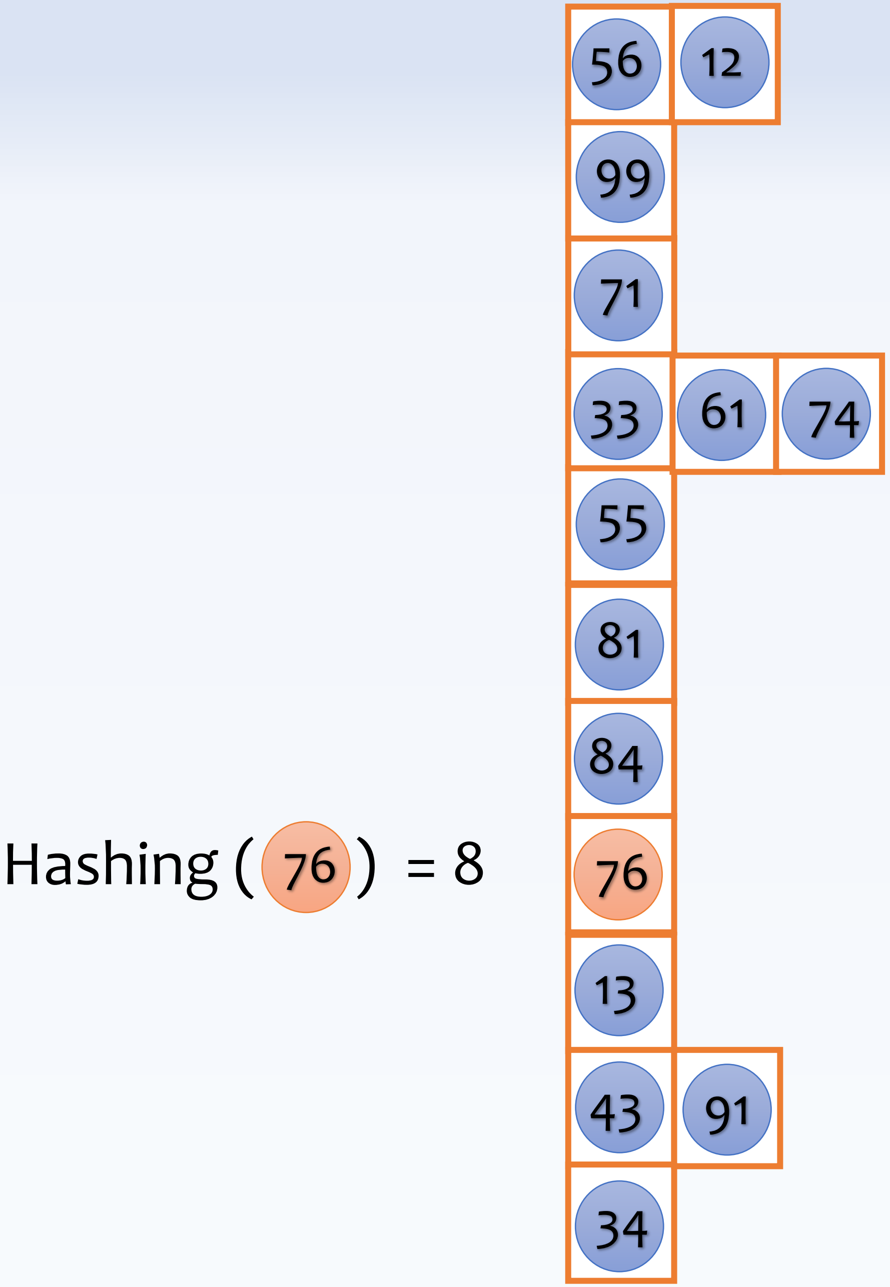
collisions
(when multiple values
hash to the same bucket)

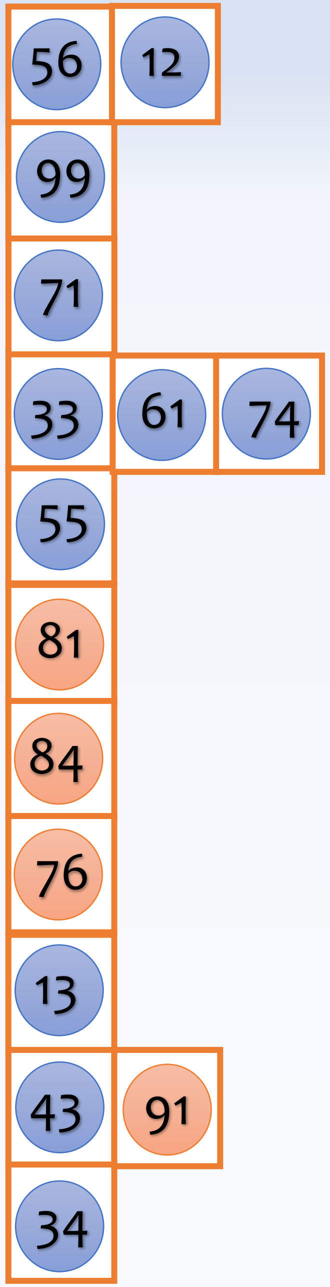


collisions
(when multiple values
hash to the same bucket)

Searching for 76

(now we can have a constant lookup cost)



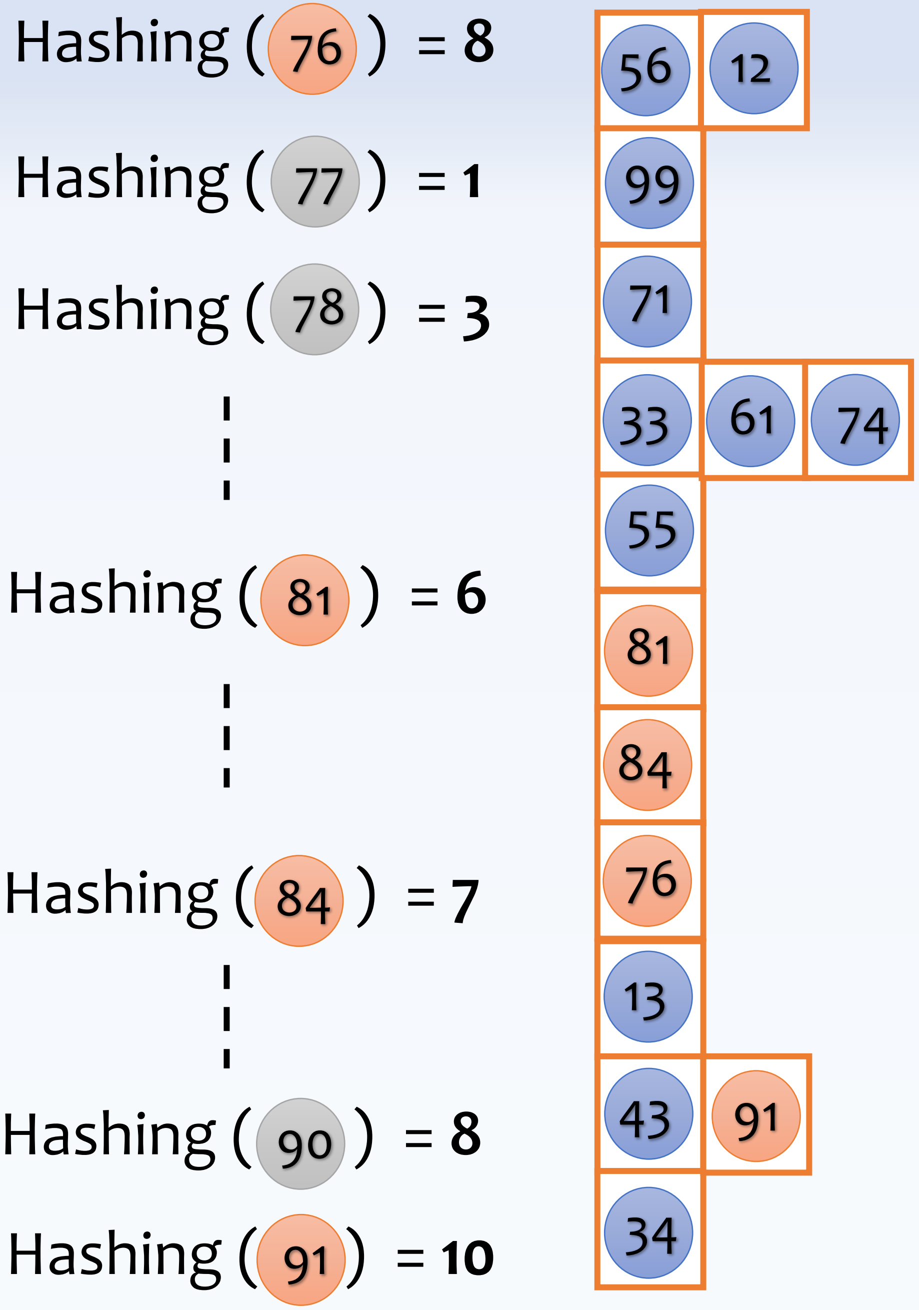


Searching for 76-91?

Could we instead search for 76, 77, 78, ..., 90, 91?

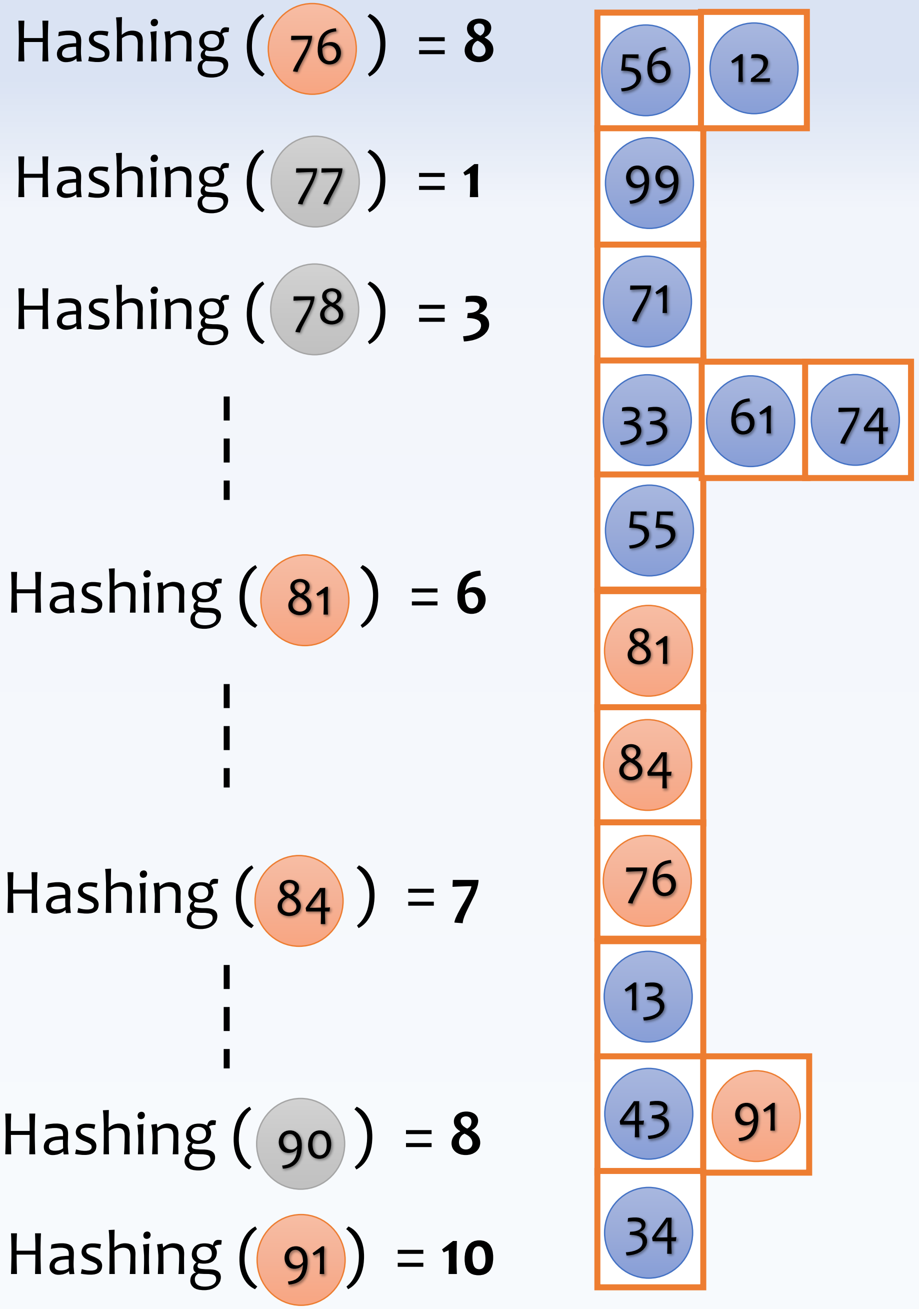
Searching for 76-91

Could we instead search for 76, 77, 78, ..., 90, 91?



Searching for 76-91

How about 76.01, 76.02, 76.03, … ?
(simply not practical)



Hashing (76) = 8

Hashing (77) = 1

Hashing (78) = 3

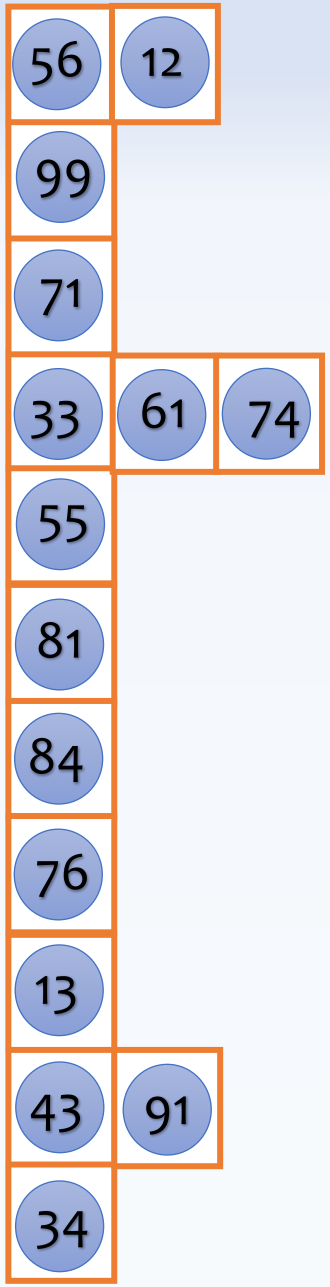
Hashing (81) = 6

Hashing (84) = 7

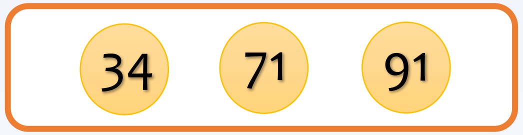
Hashing (90) = 8

Hashing (91) = 10

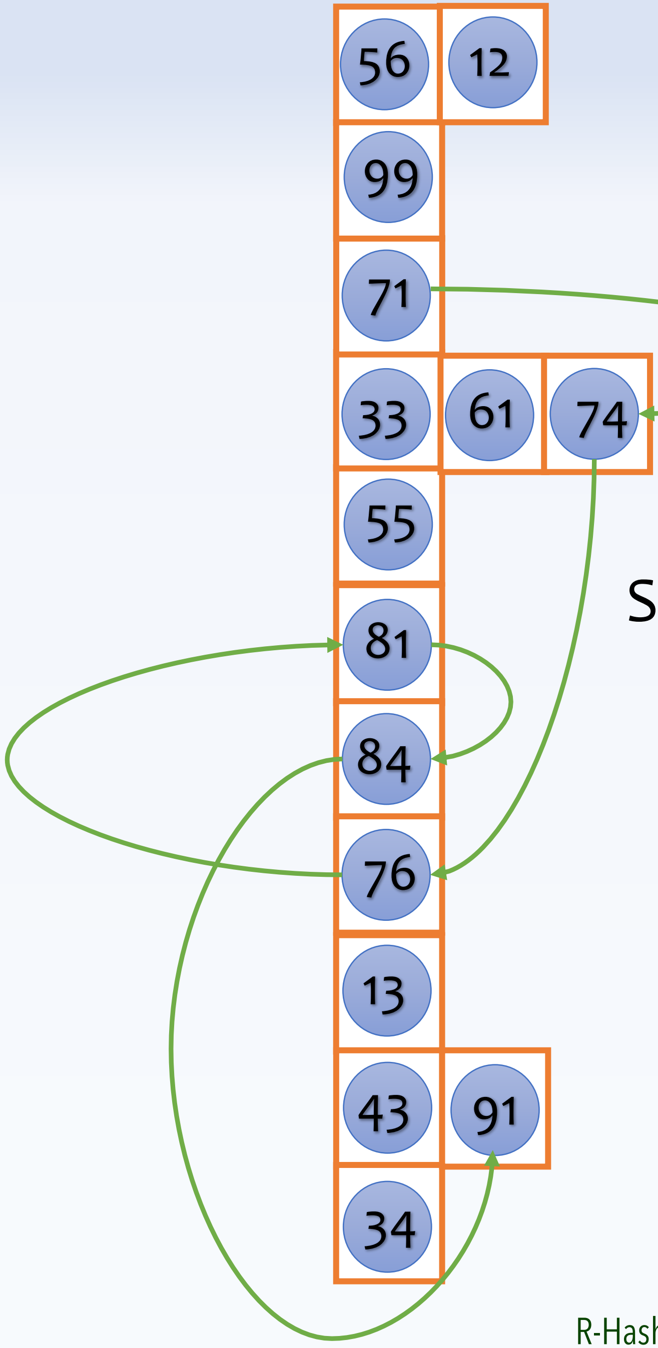
Could we imagine a new design to support searching for a range of values efficiently?



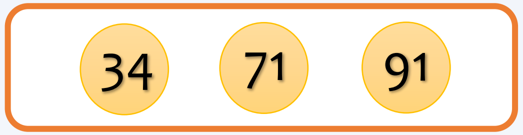
Let’s promote a subset of values as seeds



R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*



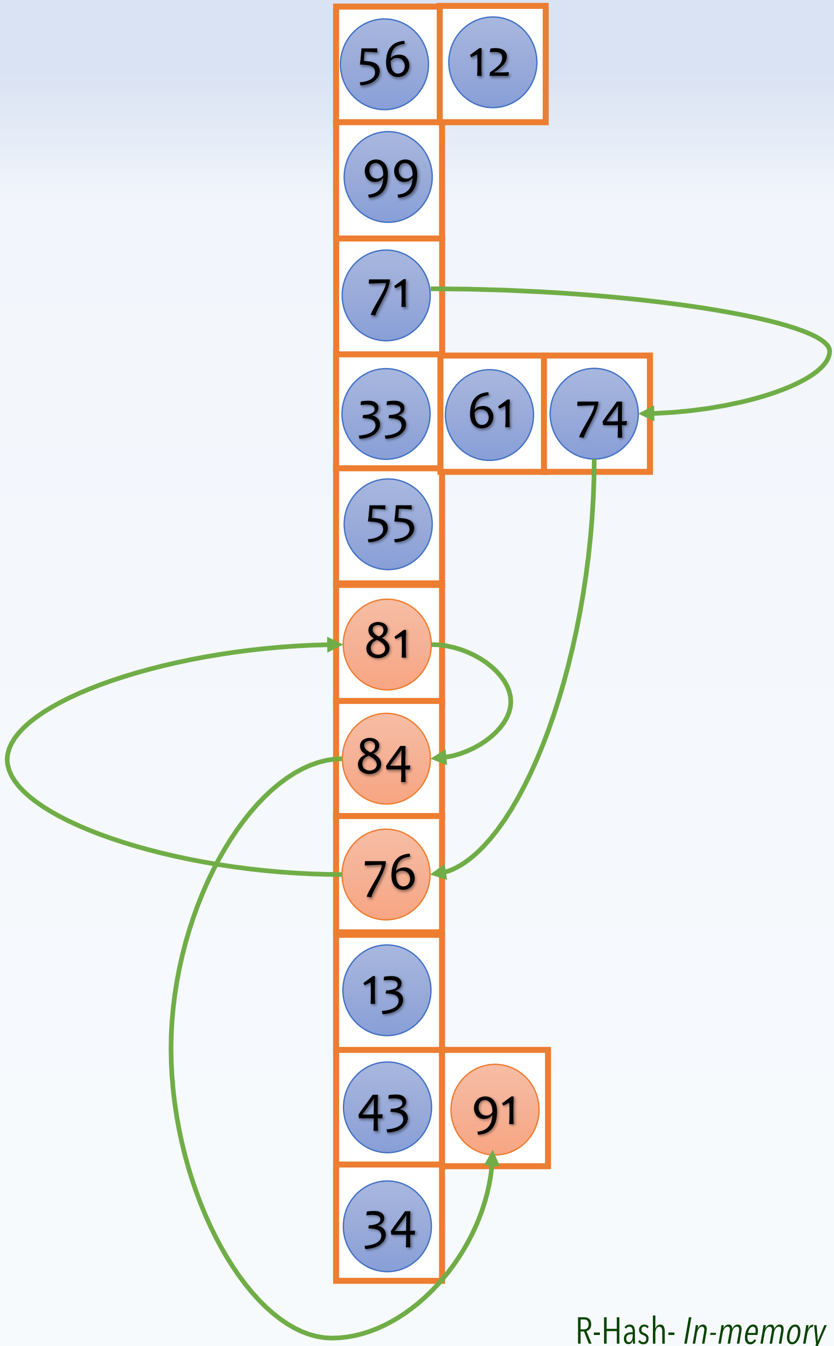
Let’s promote a subset of values as seeds



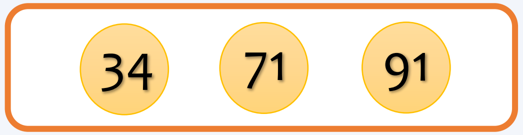
Suppose every value points to its next larger value

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

Searching for 76-91

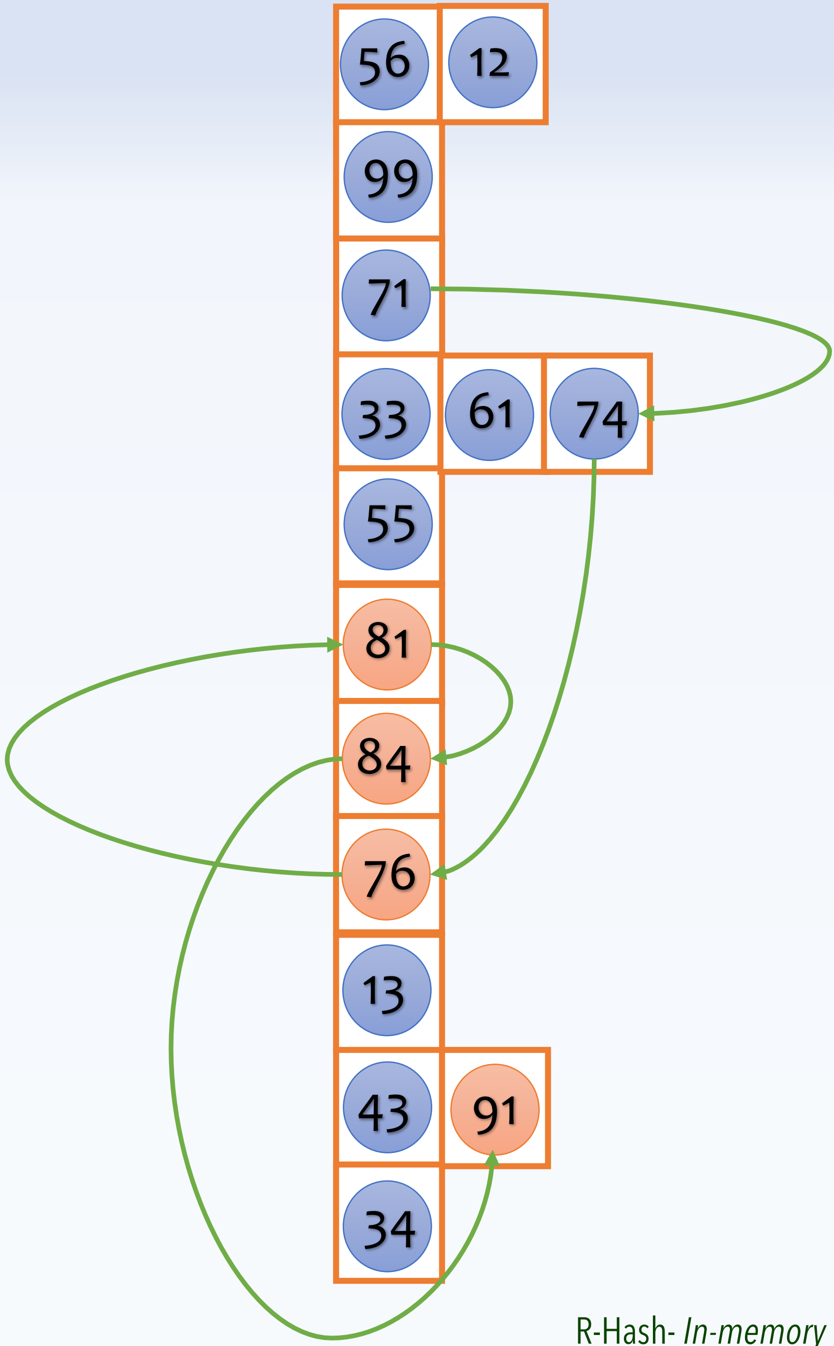


sorted seeds

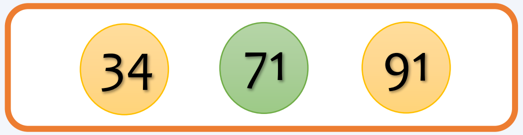


R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

Searching for 76-91



sorted seeds

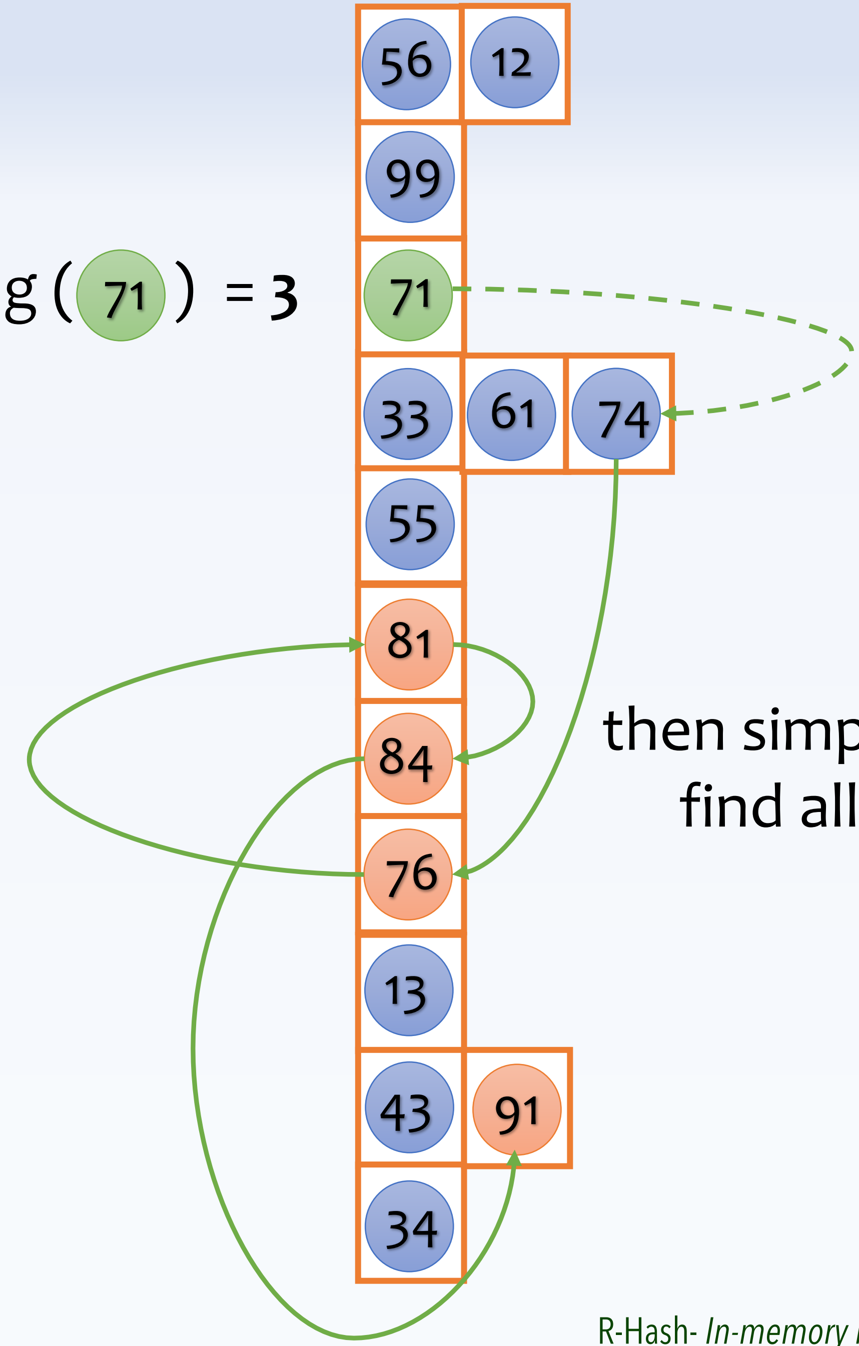


Find the largest seed smaller than 76: 71

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

Searching for 76-91

Hashing (71) = 3



sorted seeds

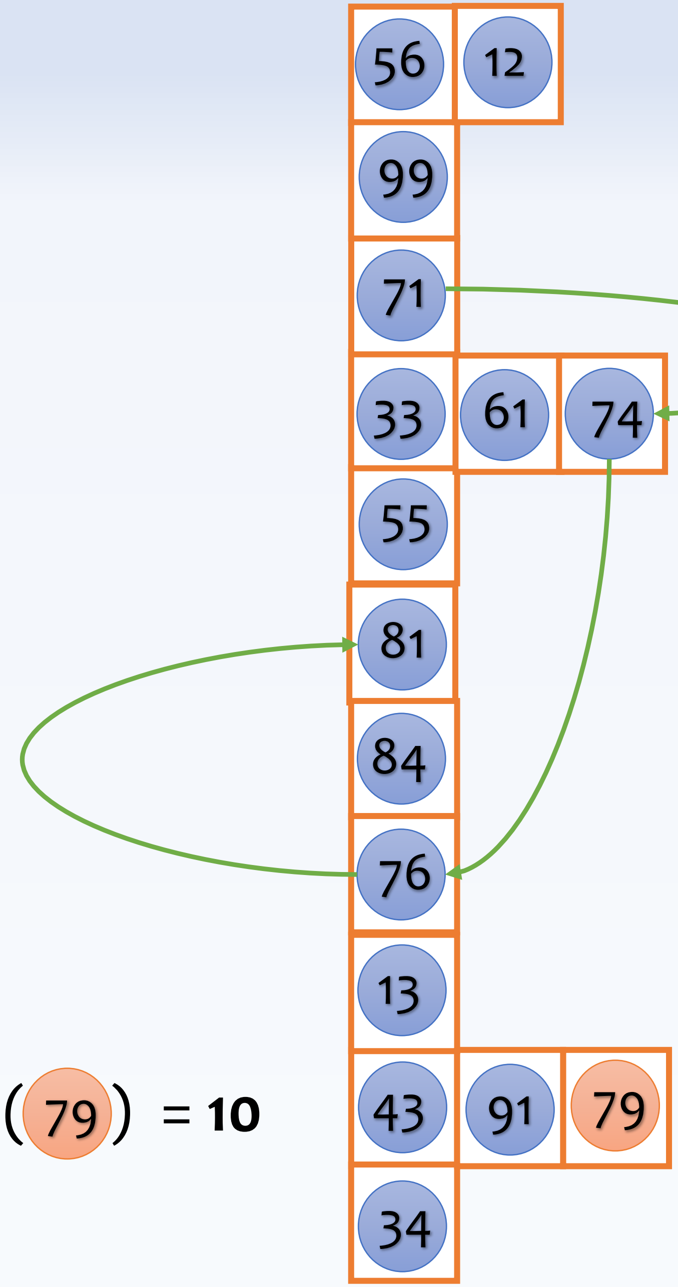
34 71 91

Find the largest seed smaller than 76: 71

then simply follow the pointers to find all values between 76-91

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

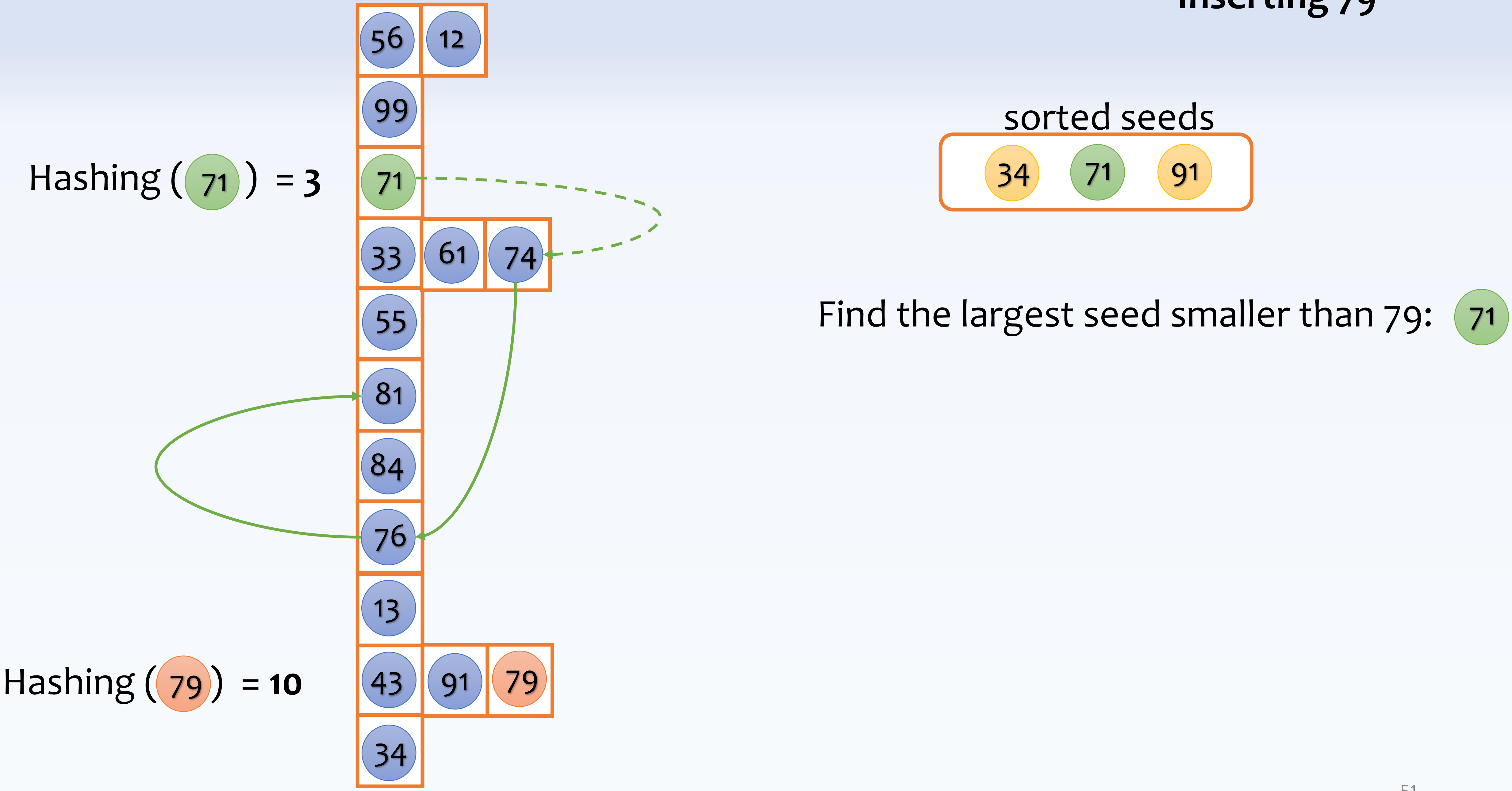
Inserting 79



Hashing (79) = **10**

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

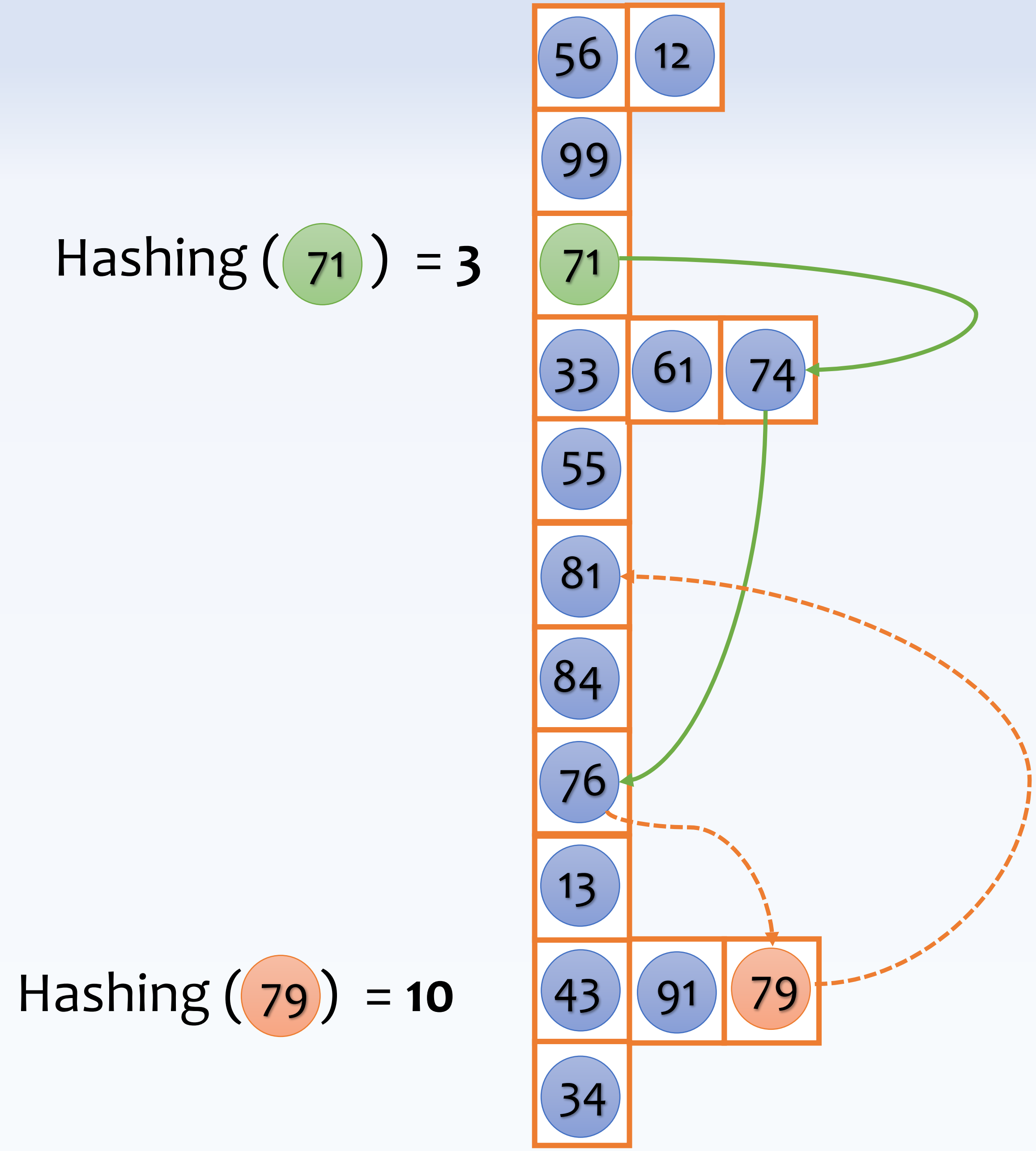
Inserting 79



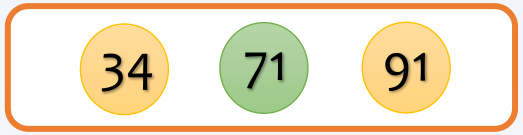
Find the largest seed smaller than 79: 71

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

Inserting 79



sorted seeds

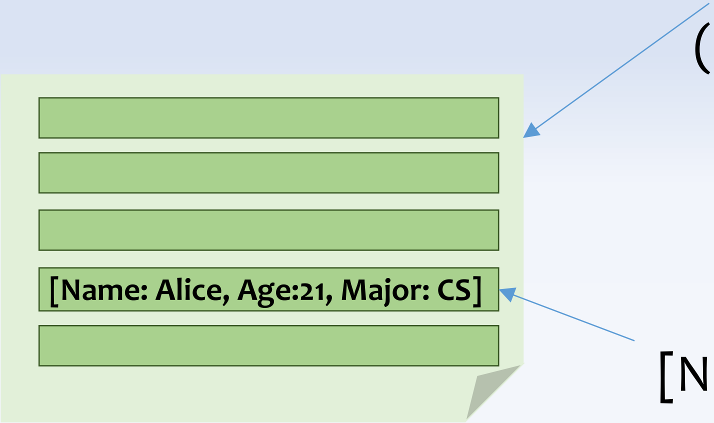


Find the largest seed smaller than 79: 71

adjust the pointers accordingly

R-Hash- *In-memory latch-free index structure* B Bhattacharjee, M Canim, M. Sadoghi, *US Patent 9,858,303*

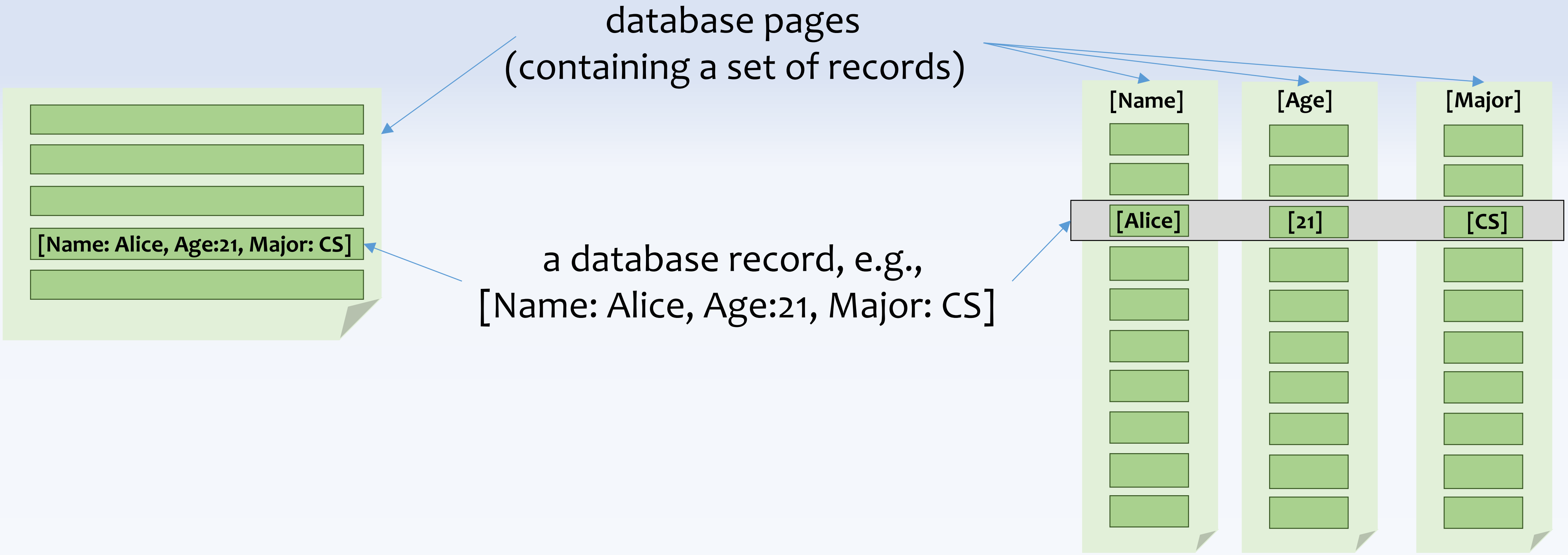
Database Storage Layouts
(how likely that we need an index for range queries?)



database pages
(containing a set of records)

a database record, e.g.,
[Name: Alice, Age:21, Major: CS]

Row-based Layout

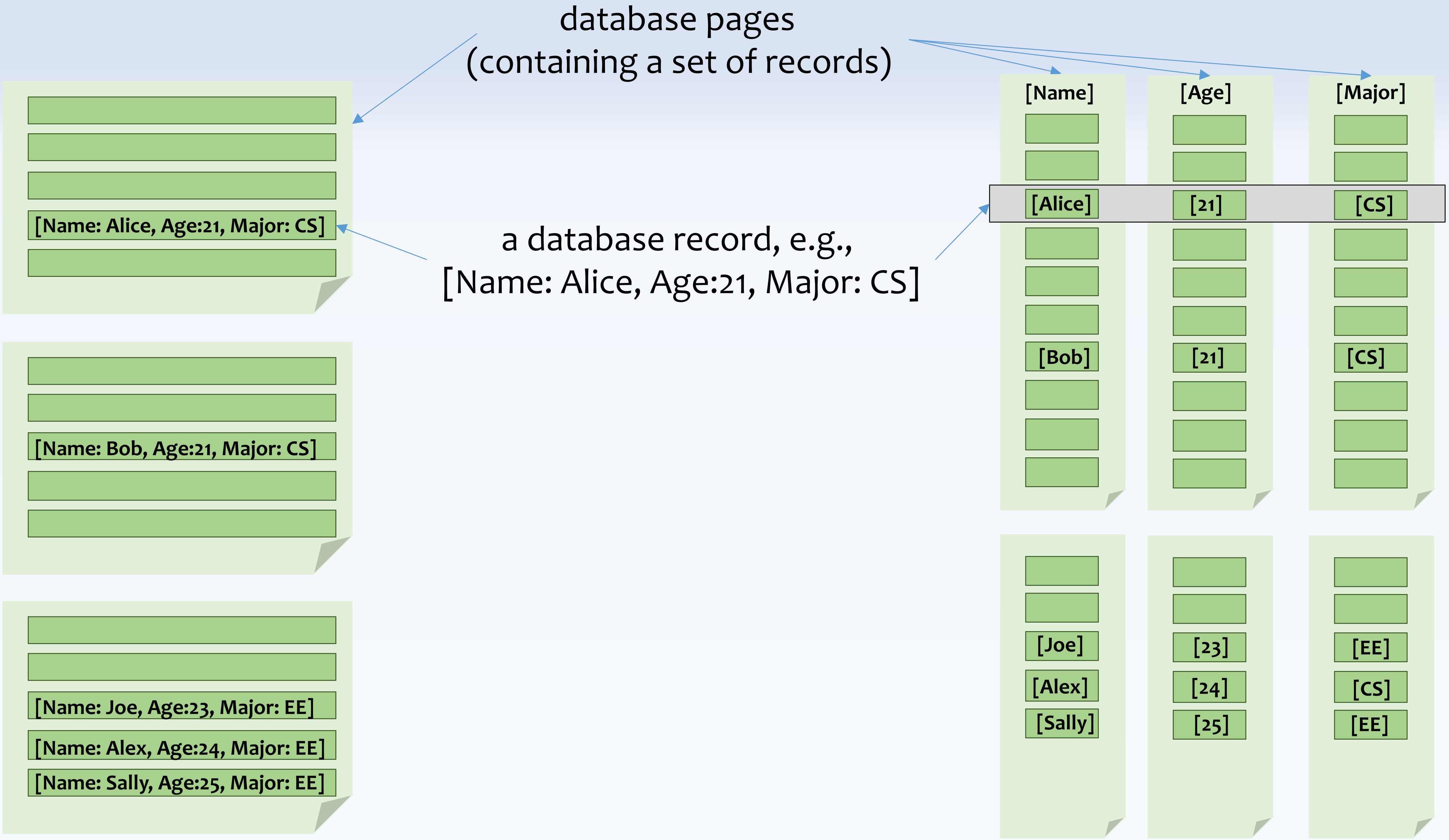


database pages
(containing a set of records)

a database record, e.g.,
[Name: Alice, Age:21, Major: CS]

Row-based Layout

Column-based Layout



database pages
(containing a set of records)

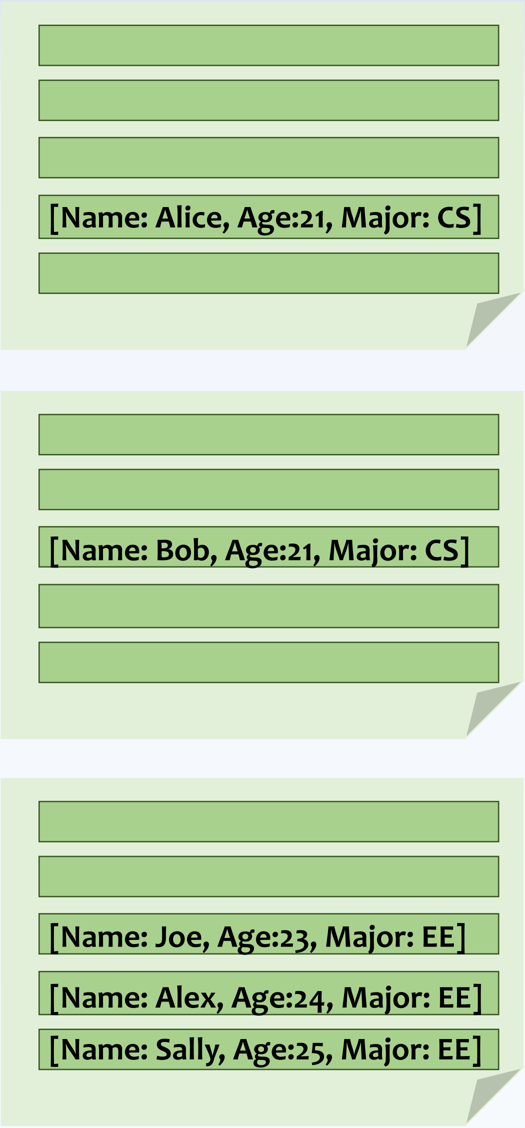
a database record, e.g.,
[Name: Alice, Age:21, Major: CS]

[Name: Alice, Age:21, Major: CS]
[Name: Bob, Age:21, Major: CS]
[Name: Joe, Age:23, Major: EE]
[Name: Alex, Age:24, Major: EE]
[Name: Sally, Age:25, Major: EE]
[Name]
[Age]
[Major]
[Alice]
[21]
[CS]
[Bob]
[21]
[CS]
[Joe]
[23]
[EE]
[Alex]
[24]
[CS]
[Sally]
[25]
[EE]

Row-based Layout

Column-based Layout

Searching for all students between the age of 21 to 24 (may return many students)



[Name: Alice, Age:21, Major: CS]

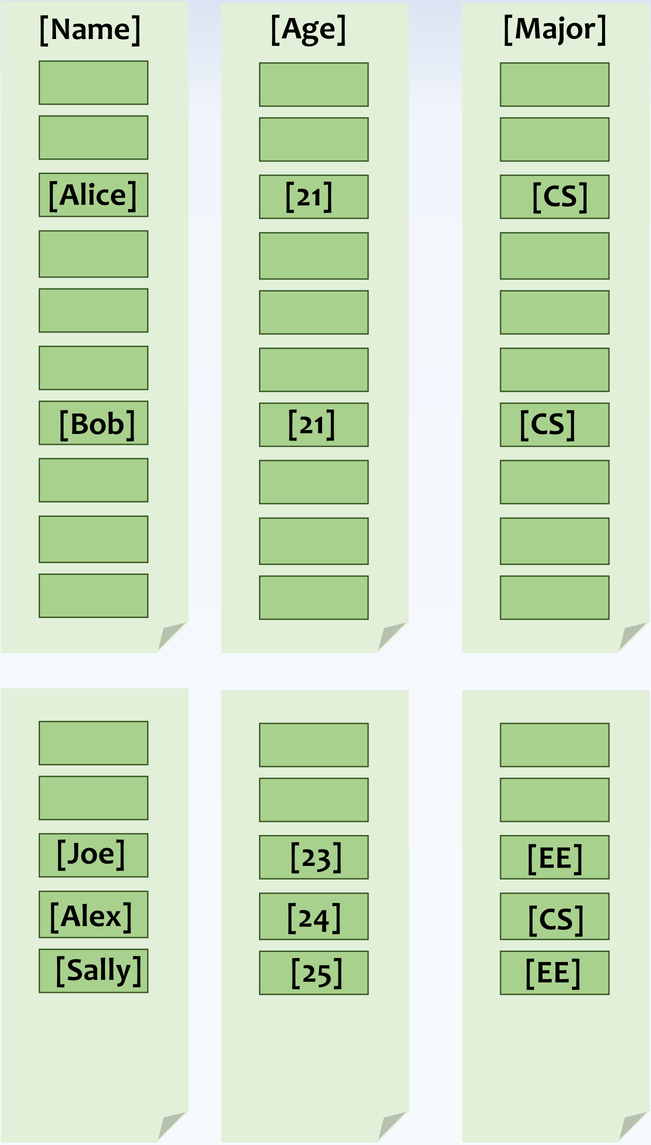
[Name: Bob, Age:21, Major: CS]

[Name: Joe, Age:23, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Sally, Age:25, Major: EE]

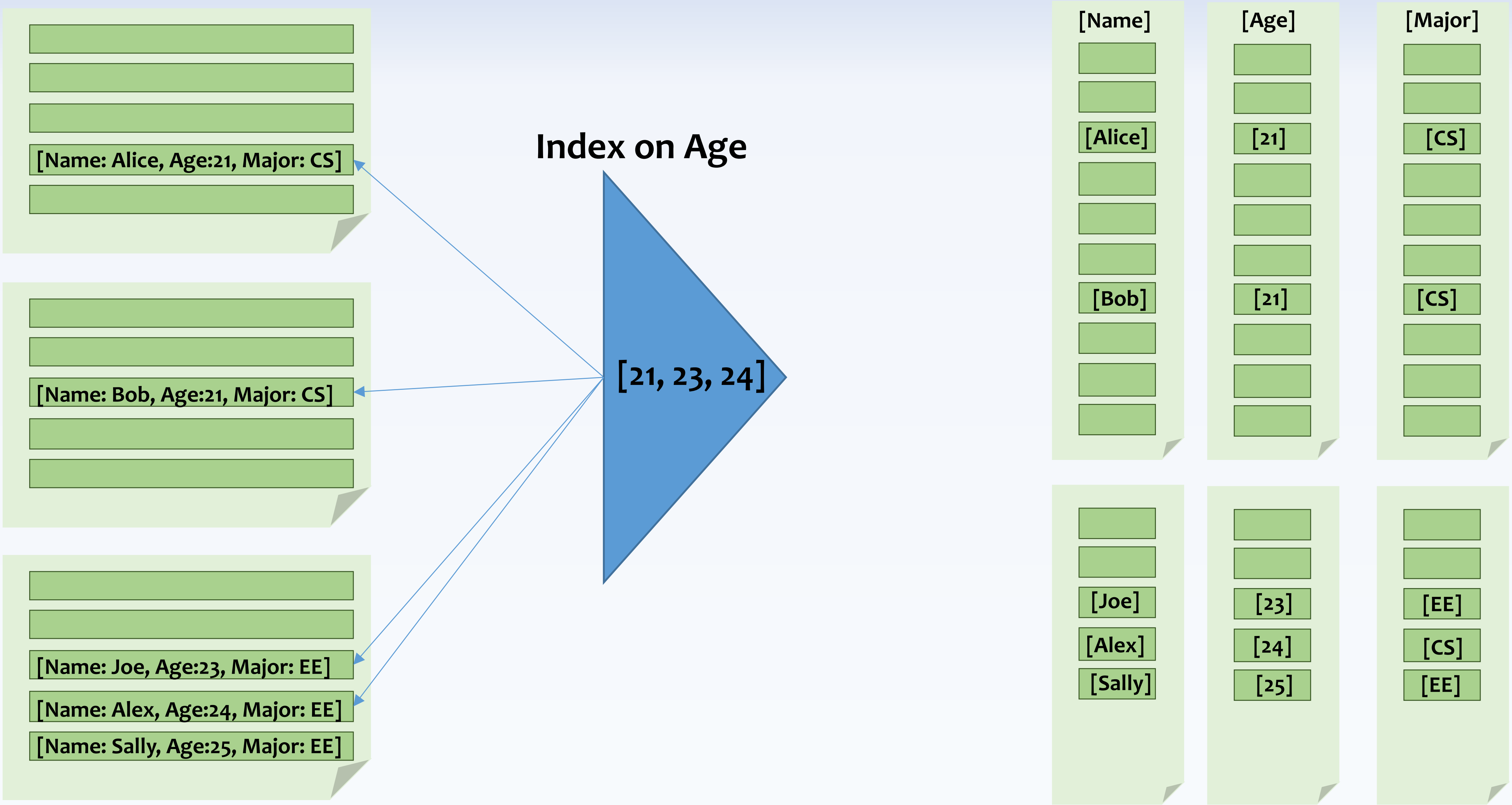
Row-based Layout



[Name]	[Age]	[Major]
[Alice]	[21]	[CS]
[Bob]	[21]	[CS]
[Joe]	[23]	[EE]
[Alex]	[24]	[CS]
[Sally]	[25]	[EE]

Column-based Layout

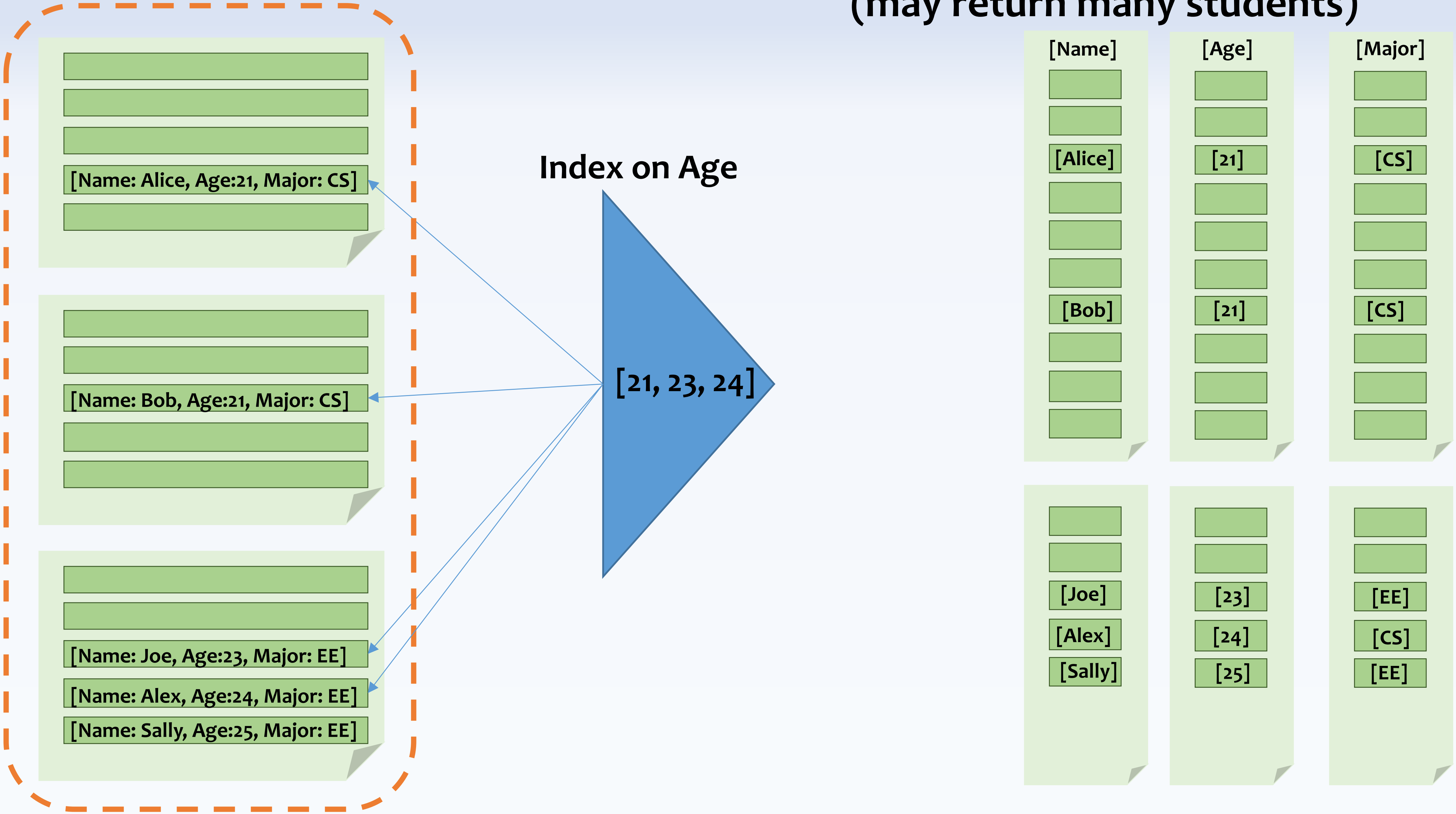
Searching for all students between the age of 21 to 24 (may return many students)


[Name: Alice, Age:21, Major: CS]
[Name: Bob, Age:21, Major: CS]
[Name: Joe, Age:23, Major: EE]
[Name: Alex, Age:24, Major: EE]
[Name: Sally, Age:25, Major: EE]
Index on Age
[21, 23, 24]
[Name]
[Age]
[Major]
[Alice]
[21]
[CS]
[Bob]
[21]
[CS]
[Joe]
[23]
[EE]
[Alex]
[24]
[CS]
[Sally]
[25]
[EE]

Row-based Layout

Column-based Layout

Searching for all students between the age of 21 to 24 (may return many students)



[Name: Alice, Age:21, Major: CS]

[Name: Bob, Age:21, Major: CS]

[Name: Joe, Age:23, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Sally, Age:25, Major: EE]

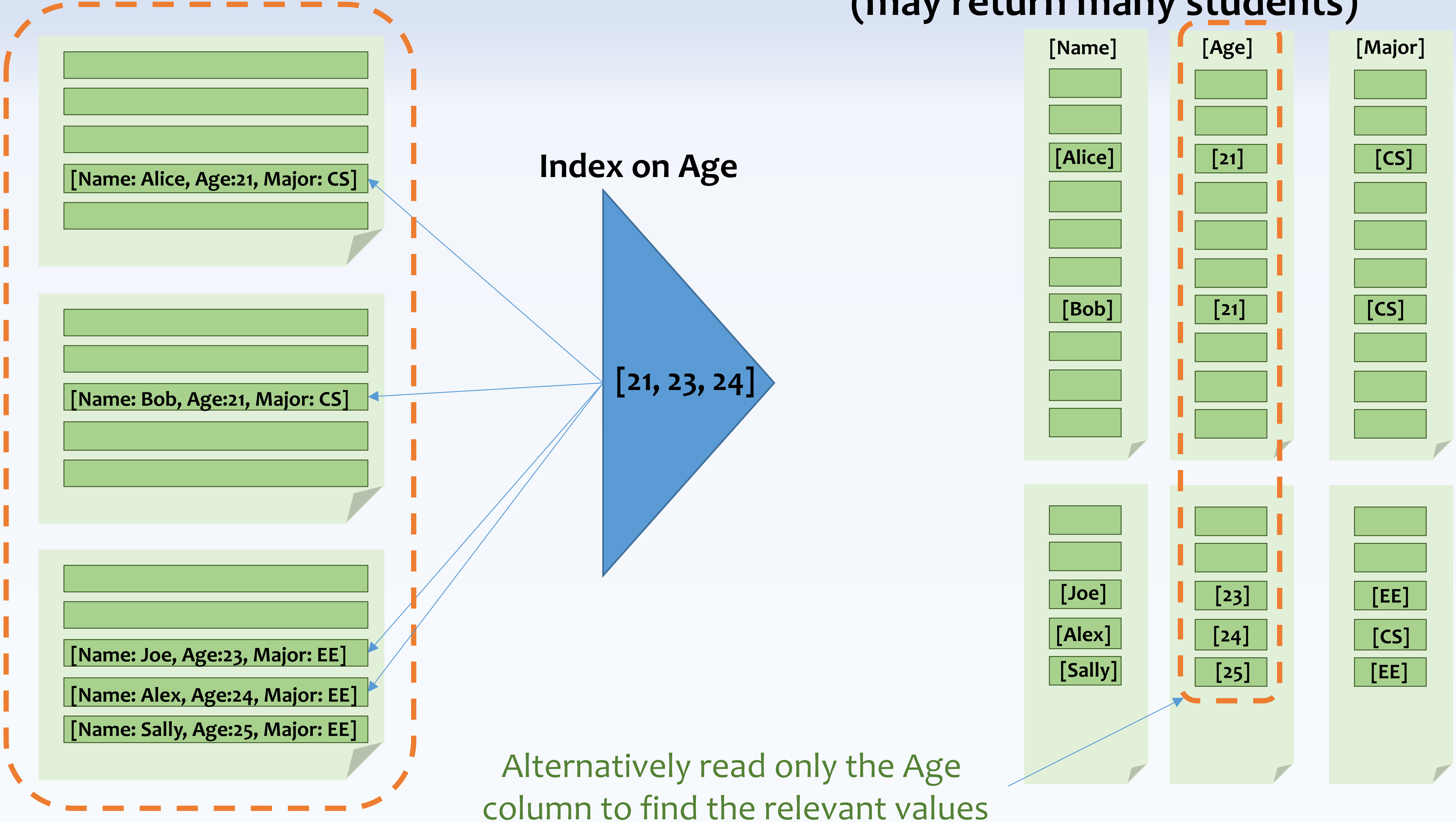
Index on Age

[21, 23, 24]

Row-based Layout

Column-based Layout

Searching for all students between the age of 21 to 24 (may return many students)



[Name: Alice, Age:21, Major: CS]

[Name: Bob, Age:21, Major: CS]

[Name: Joe, Age:23, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Sally, Age:25, Major: EE]

Index on Age

[21, 23, 24]

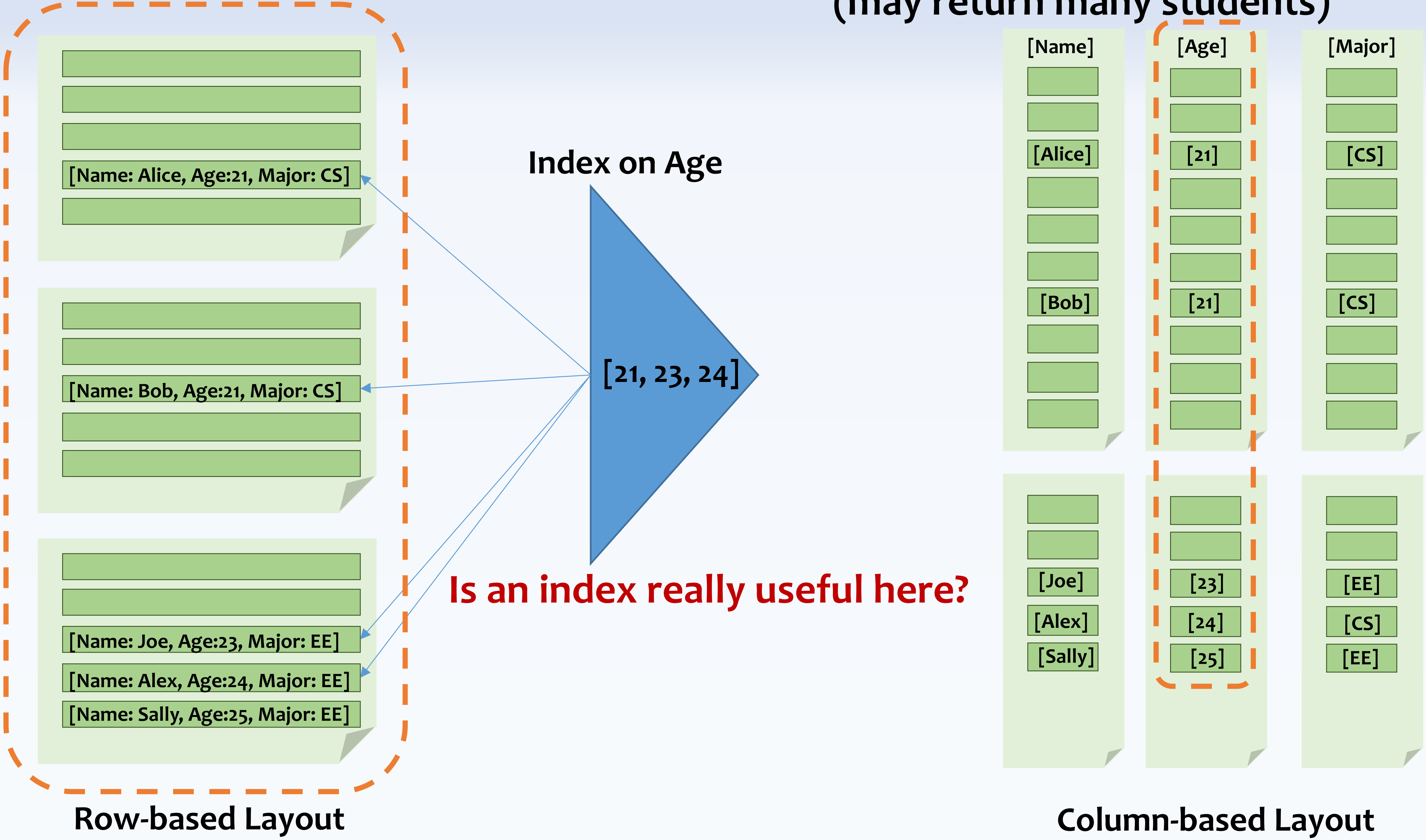
[Name]	[Age]	[Major]
[Alice]	[21]	[CS]
[Bob]	[21]	[CS]
[Joe]	[23]	[EE]
[Alex]	[24]	[CS]
[Sally]	[25]	[EE]

Alternatively read only the Age column to find the relevant values

Row-based Layout

Column-based Layout

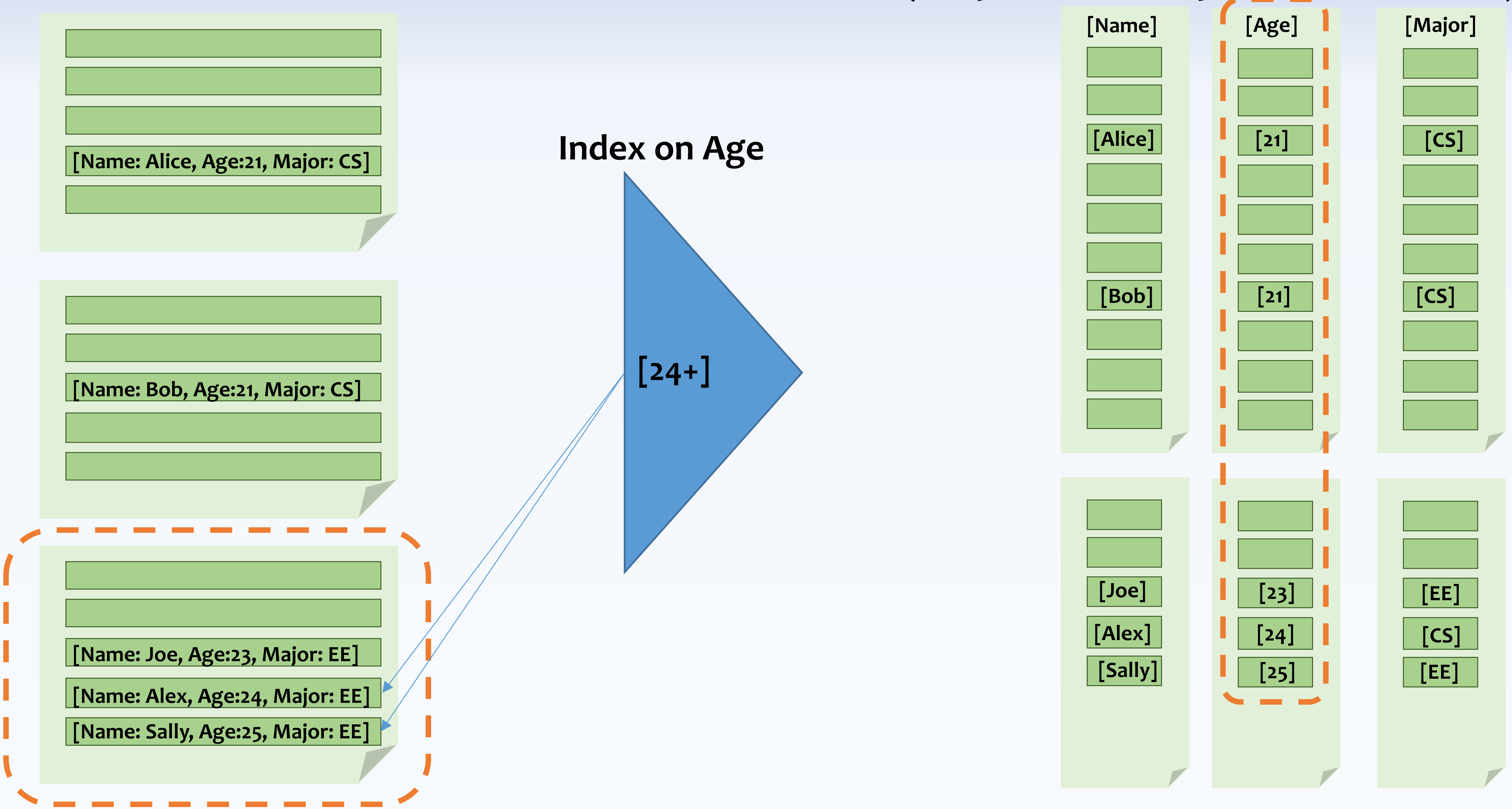
Searching for all students between the age of 21 to 24 (may return many students)



[Name: Alice, Age:21, Major: CS]
[Name: Bob, Age:21, Major: CS]
[Name: Joe, Age:23, Major: EE]
[Name: Alex, Age:24, Major: EE]
[Name: Sally, Age:25, Major: EE]
Row-based Layout
Index on Age
[21, 23, 24]
[Name]
[Alice]
[Bob]
[Joe]
[Alex]
[Sally]
[Age]
[21]
[21]
[23]
[24]
[25]
[Major]
[CS]
[CS]
[EE]
[CS]
[EE]
Column-based Layout

Is an index really useful here?

Searching for all students over the age of 24 (may return only a few students)



[Name: Alice, Age:21, Major: CS]

[Name: Bob, Age:21, Major: CS]

[Name: Joe, Age:23, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Sally, Age:25, Major: EE]

Index on Age

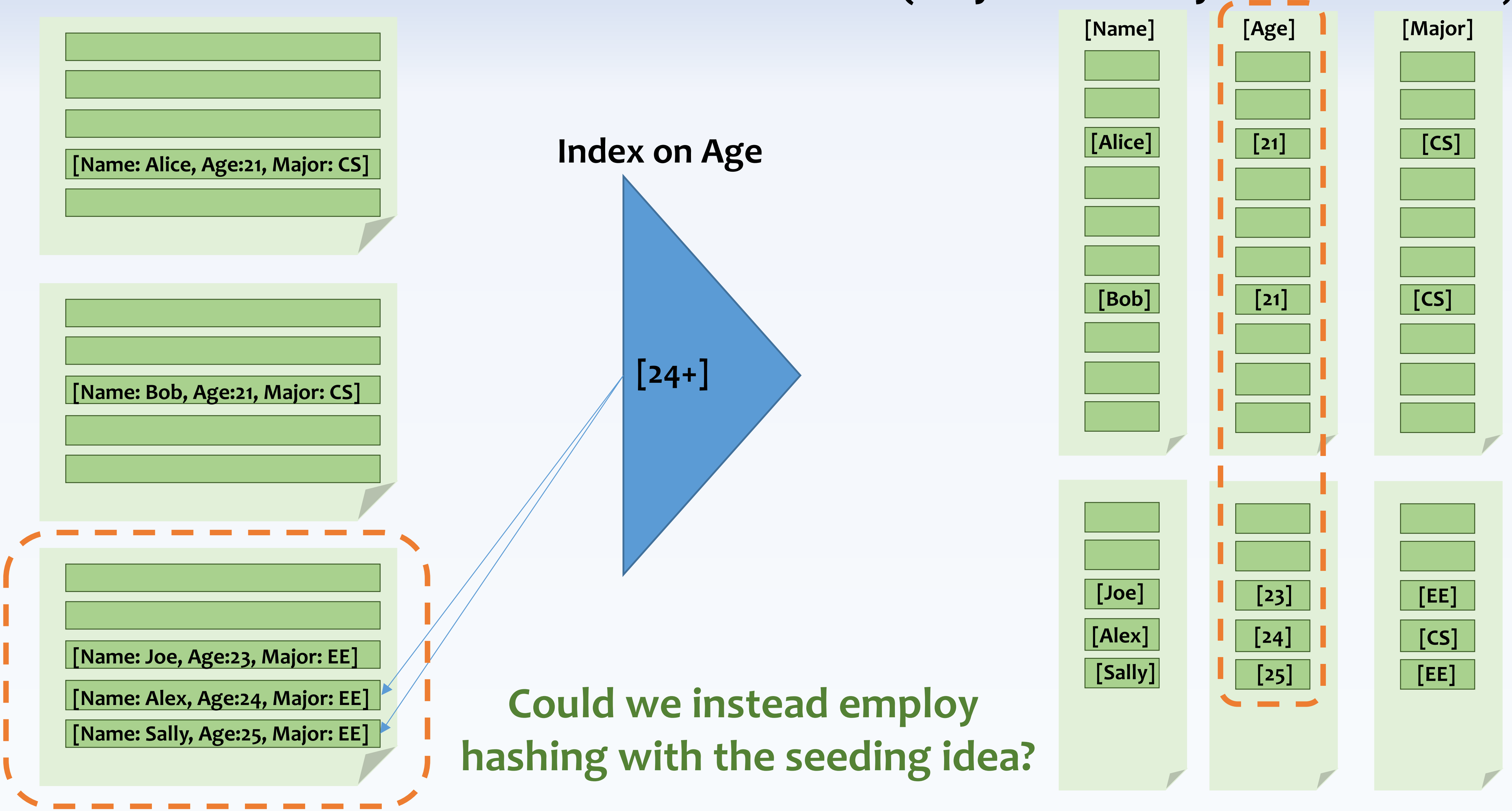
[24+]

[Name]	[Age]	[Major]
[Alice]	[21]	[CS]
[Bob]	[21]	[CS]
[Joe]	[23]	[EE]
[Alex]	[24]	[CS]
[Sally]	[25]	[EE]

Row-based Layout

Column-based Layout

Searching for all students over the age of 24 (may return only a few students)



[Name: Alice, Age:21, Major: CS]

[Name: Bob, Age:21, Major: CS]

[Name: Joe, Age:23, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Sally, Age:25, Major: EE]

Index on Age

[24+]

[Name]	[Age]	[Major]
[Alice]	[21]	[CS]
[Bob]	[21]	[CS]
[Joe]	[23]	[EE]
[Alex]	[24]	[CS]
[Sally]	[25]	[EE]

Could we instead employ hashing with the seeding idea?

Row-based Layout

Column-based Layout

Thank You
Questions?