

# *Storage & Indexing in Modern Databases*

**ECS 165A – Winter 2021**

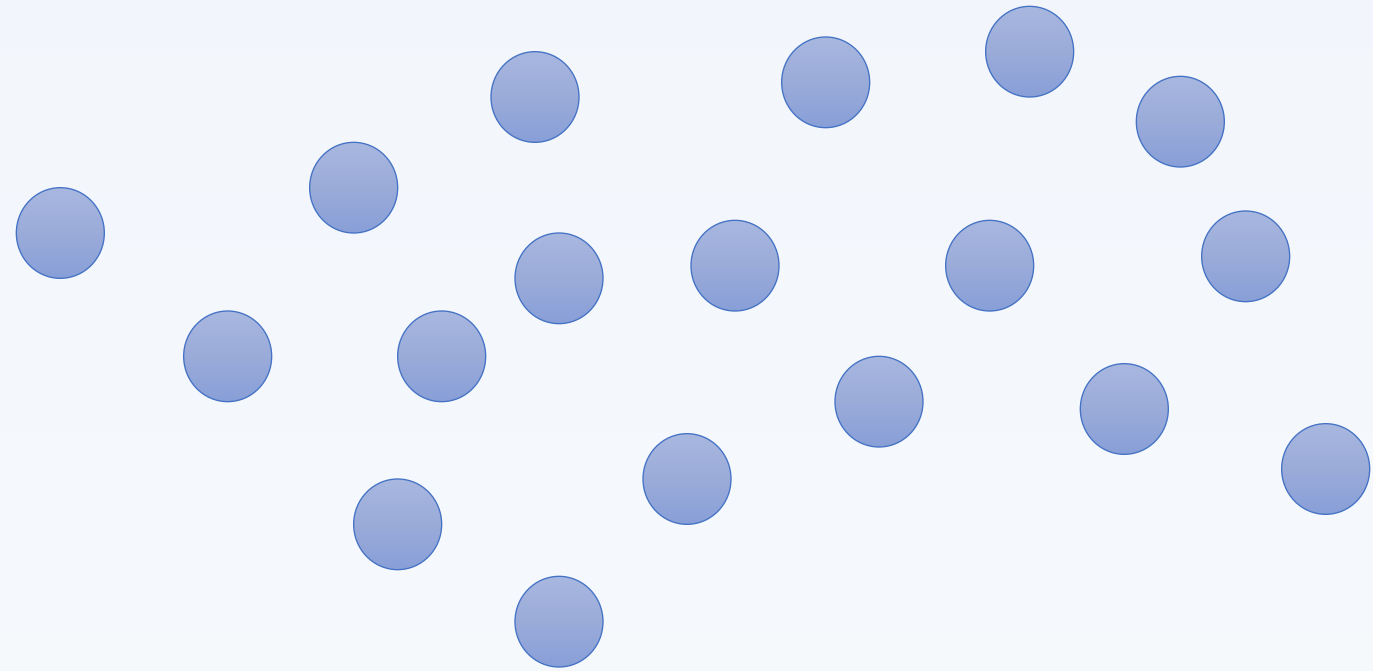


**Mohammad Sadoghi**  
Exploratory Systems Lab  
Department of Computer Science

**UC DAVIS**  
UNIVERSITY OF CALIFORNIA

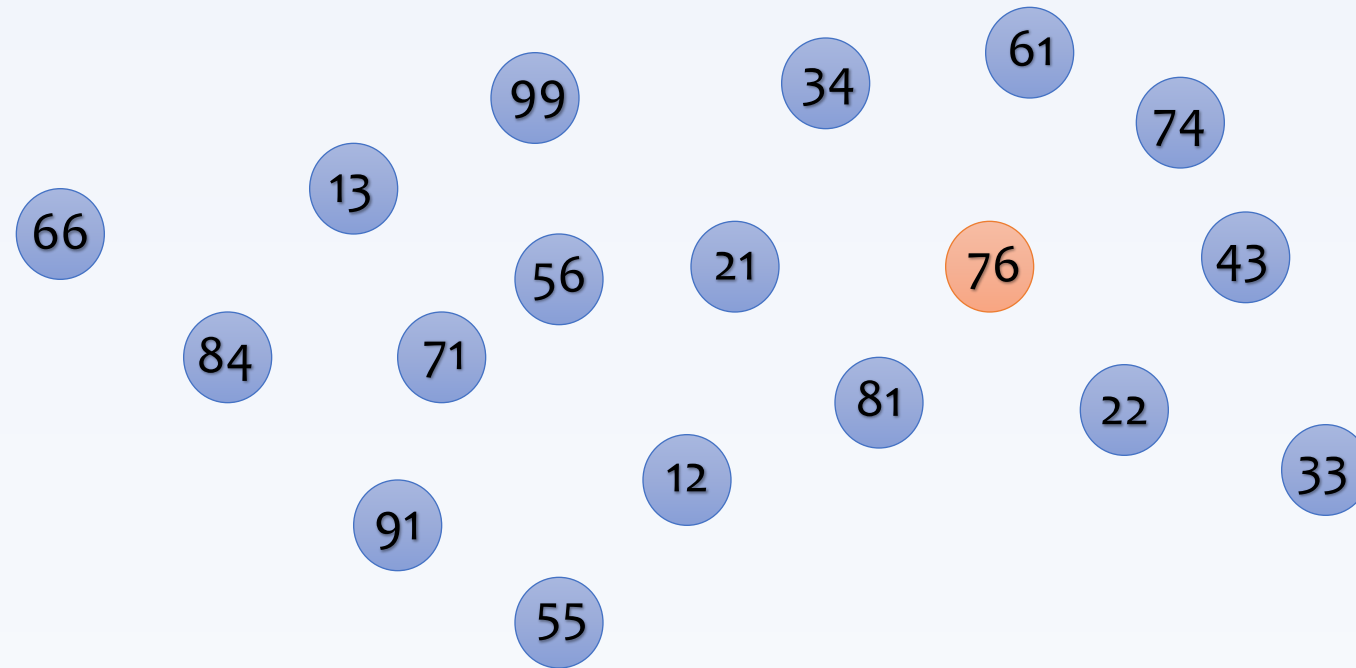


**How to quickly search for the desired information?**

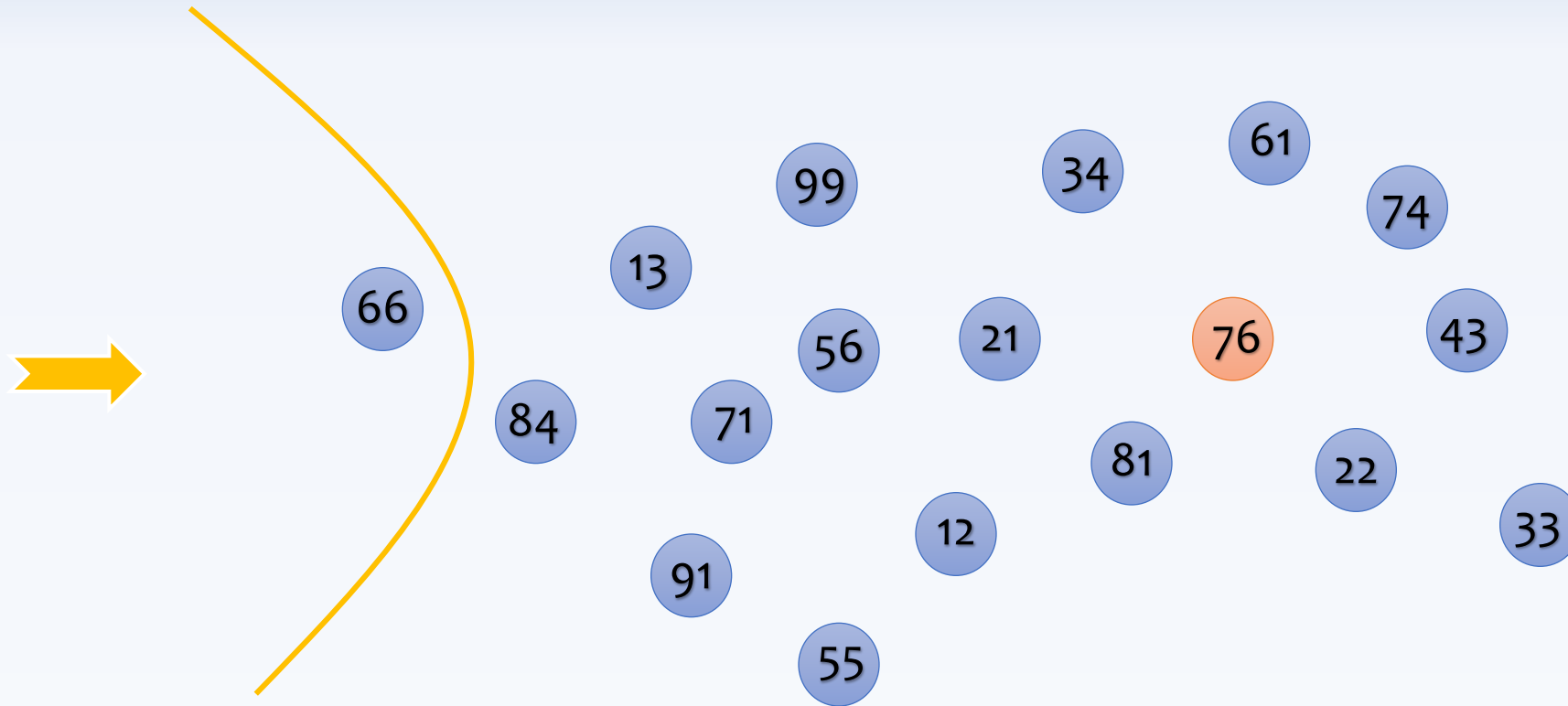




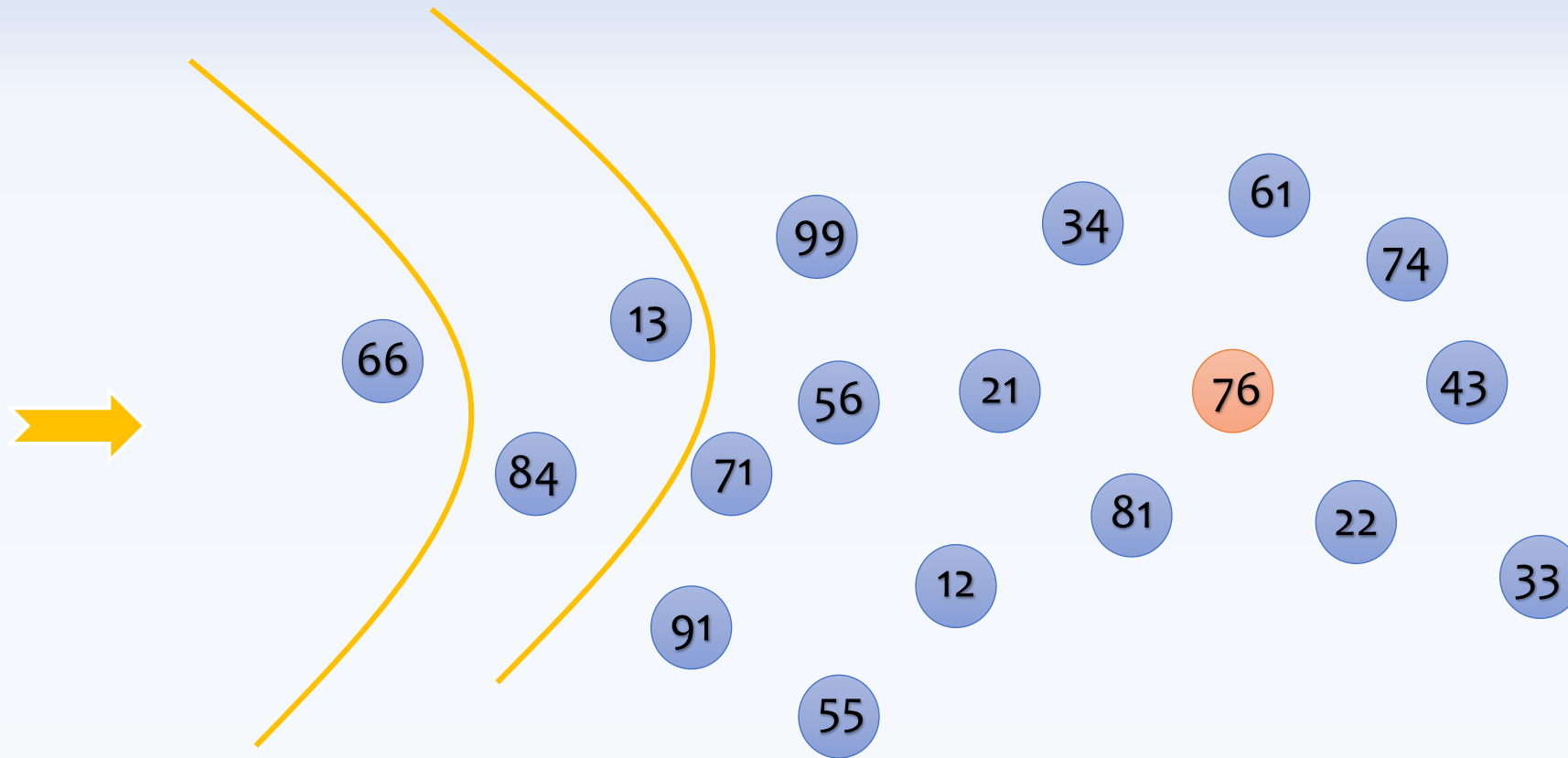
# Searching for 76



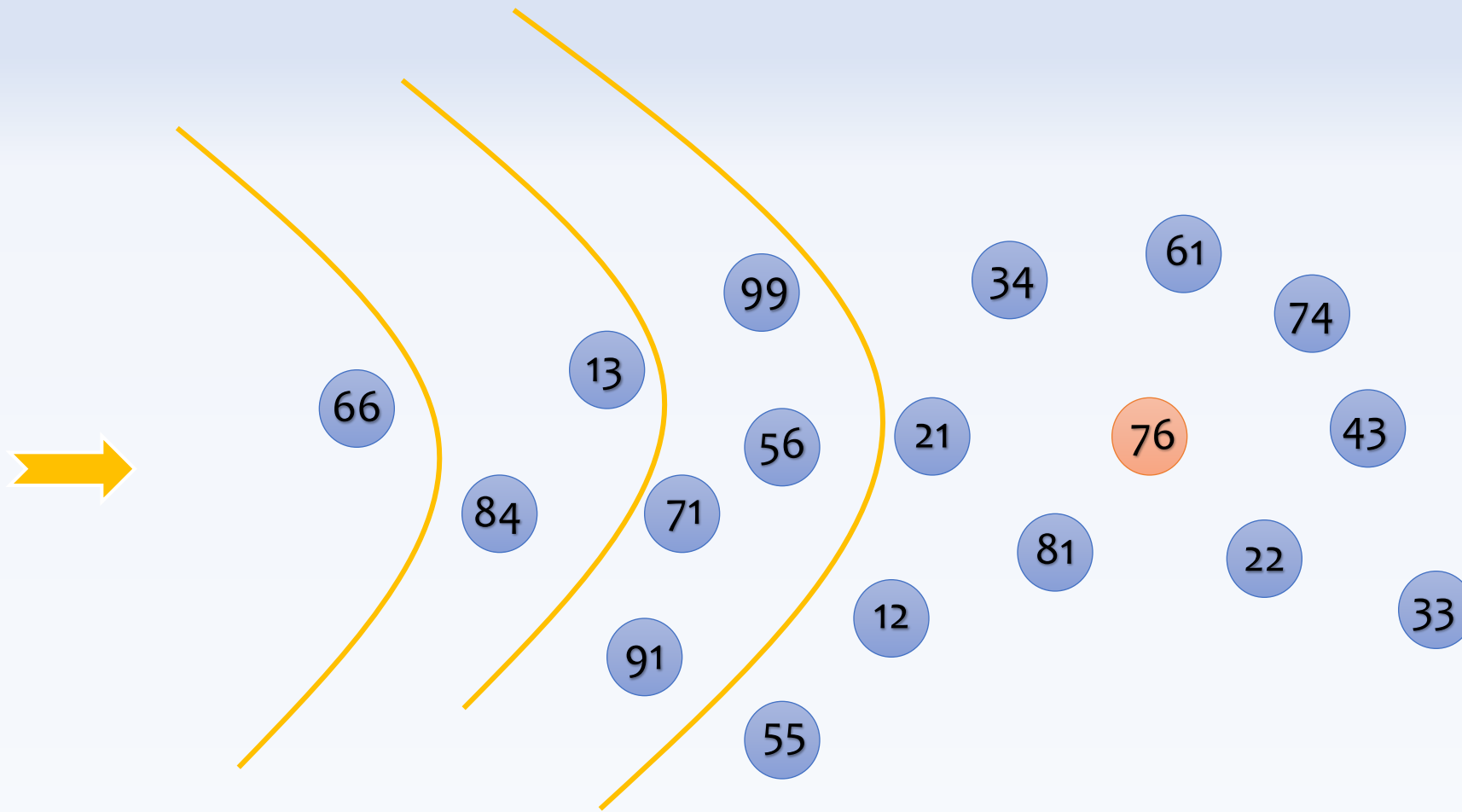
# Searching for 76



# Searching for 76

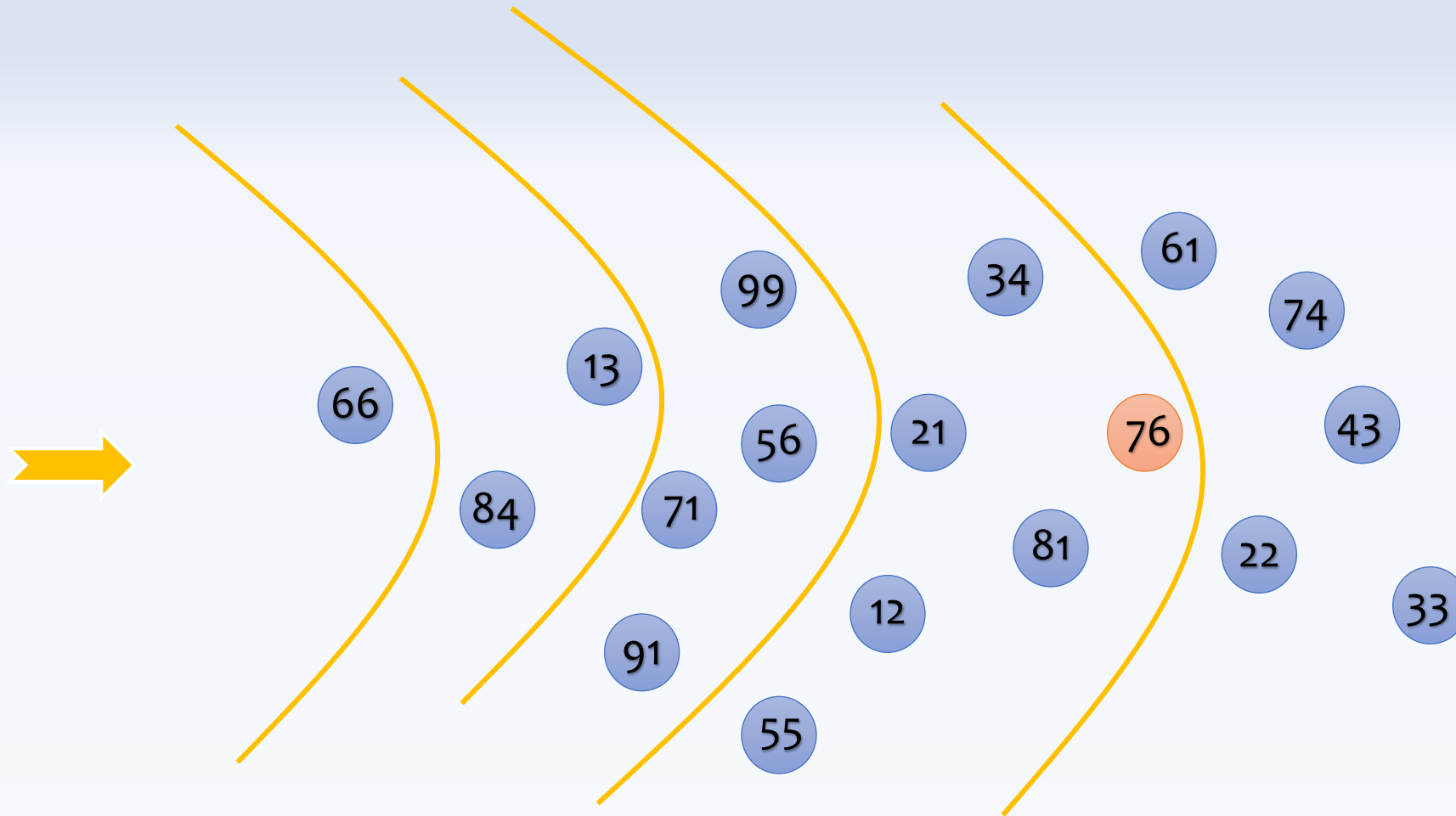


# Searching for 76

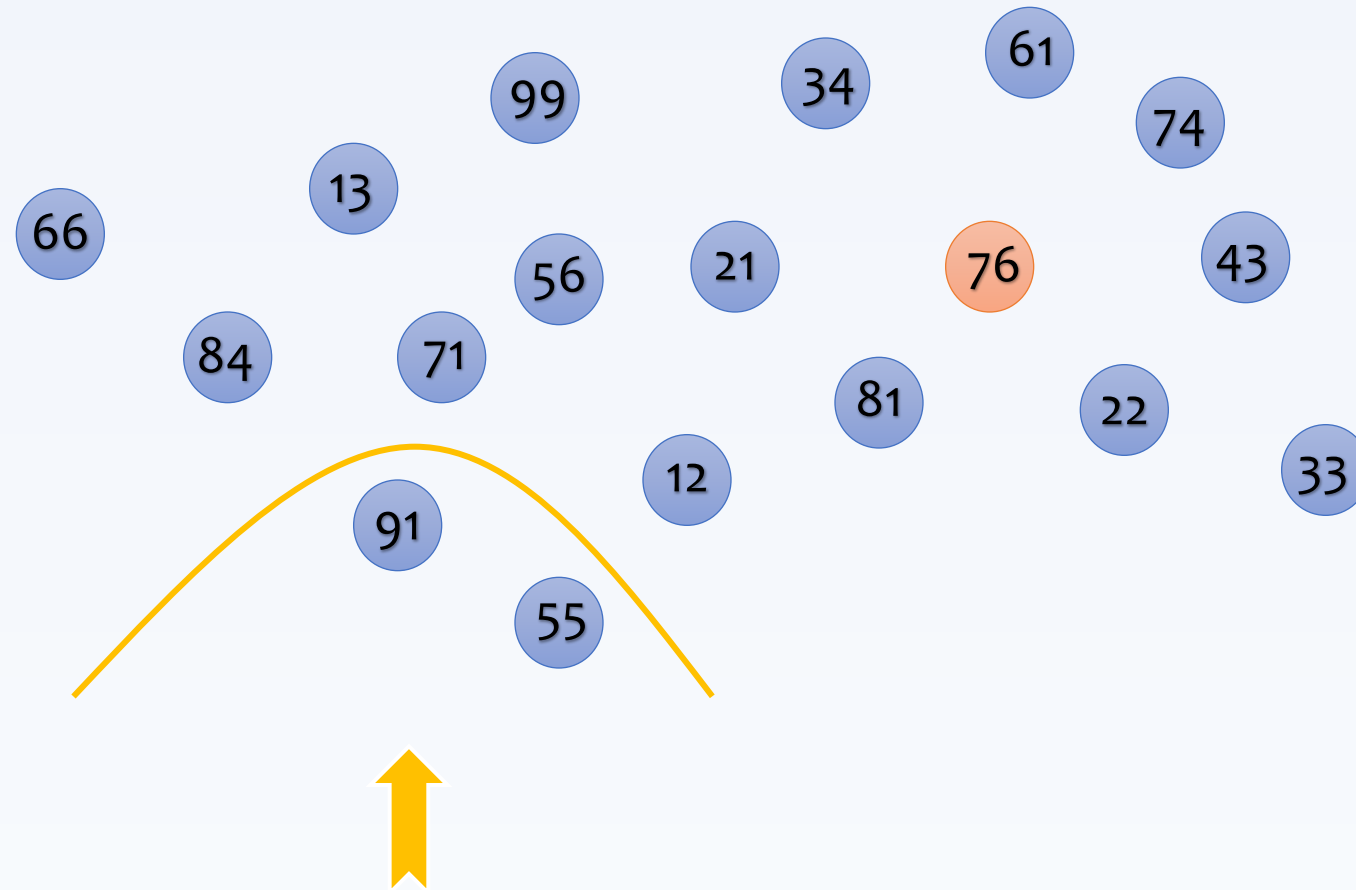




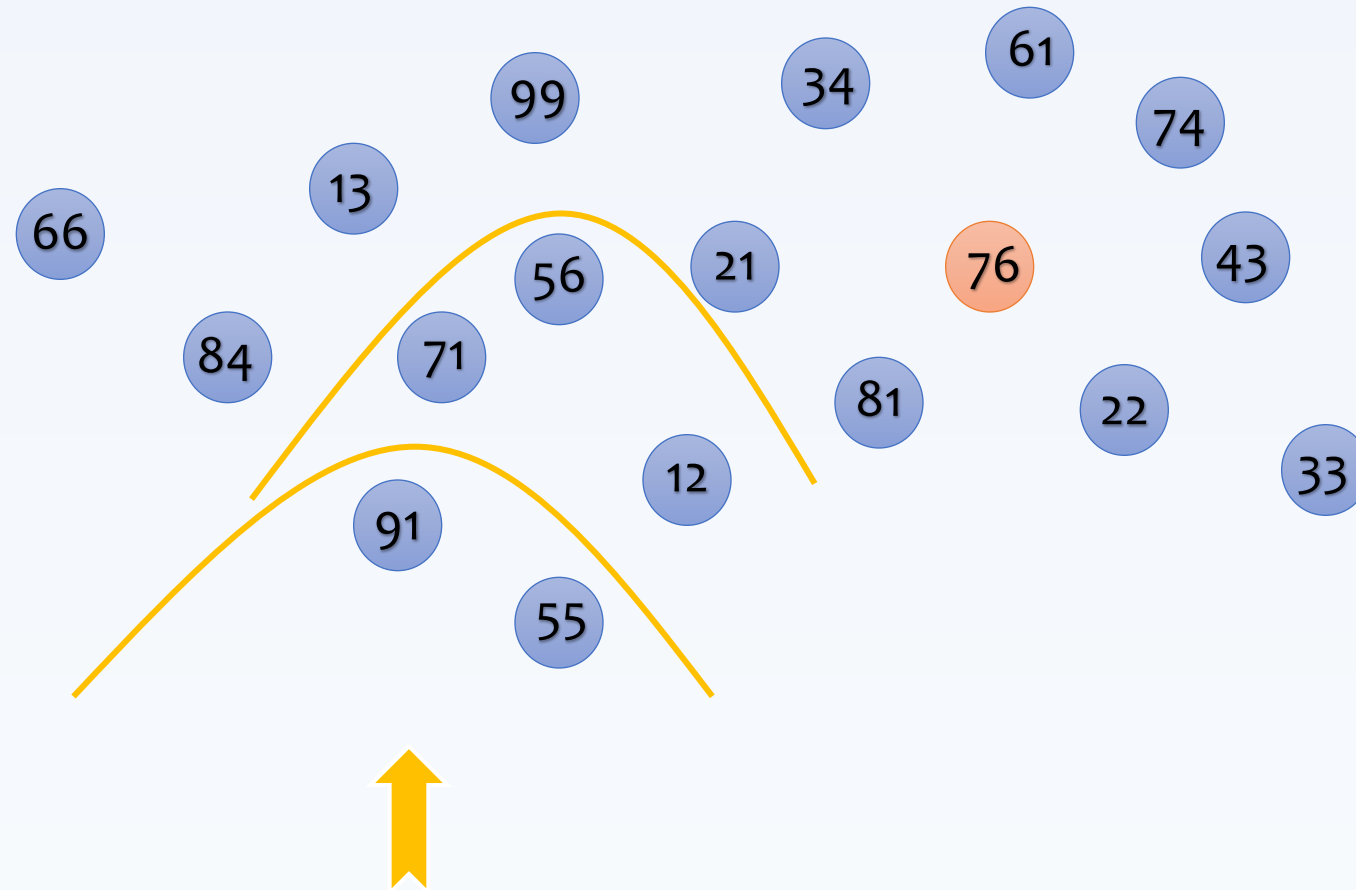
# Searching for 76



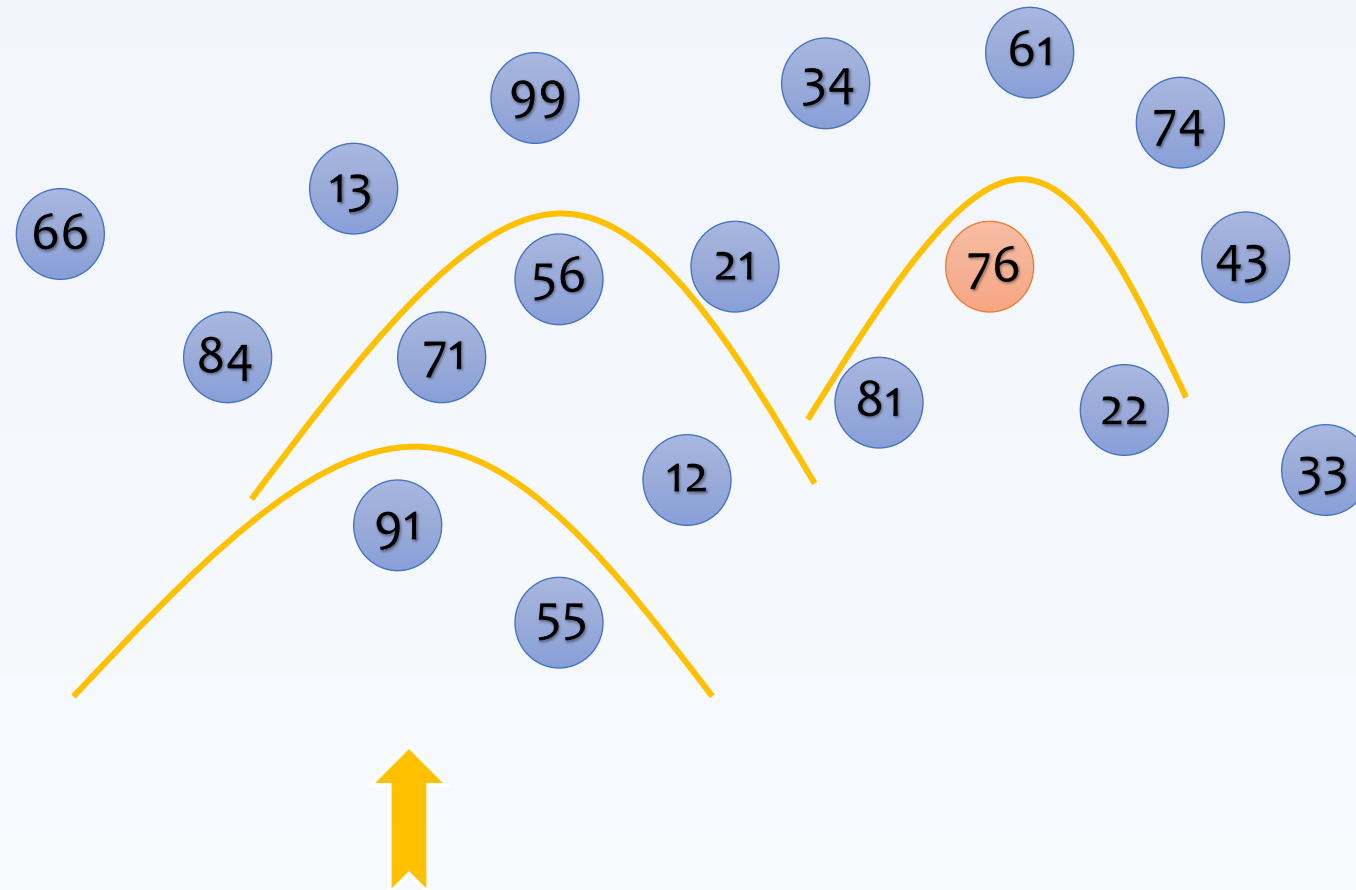
# Searching for 76



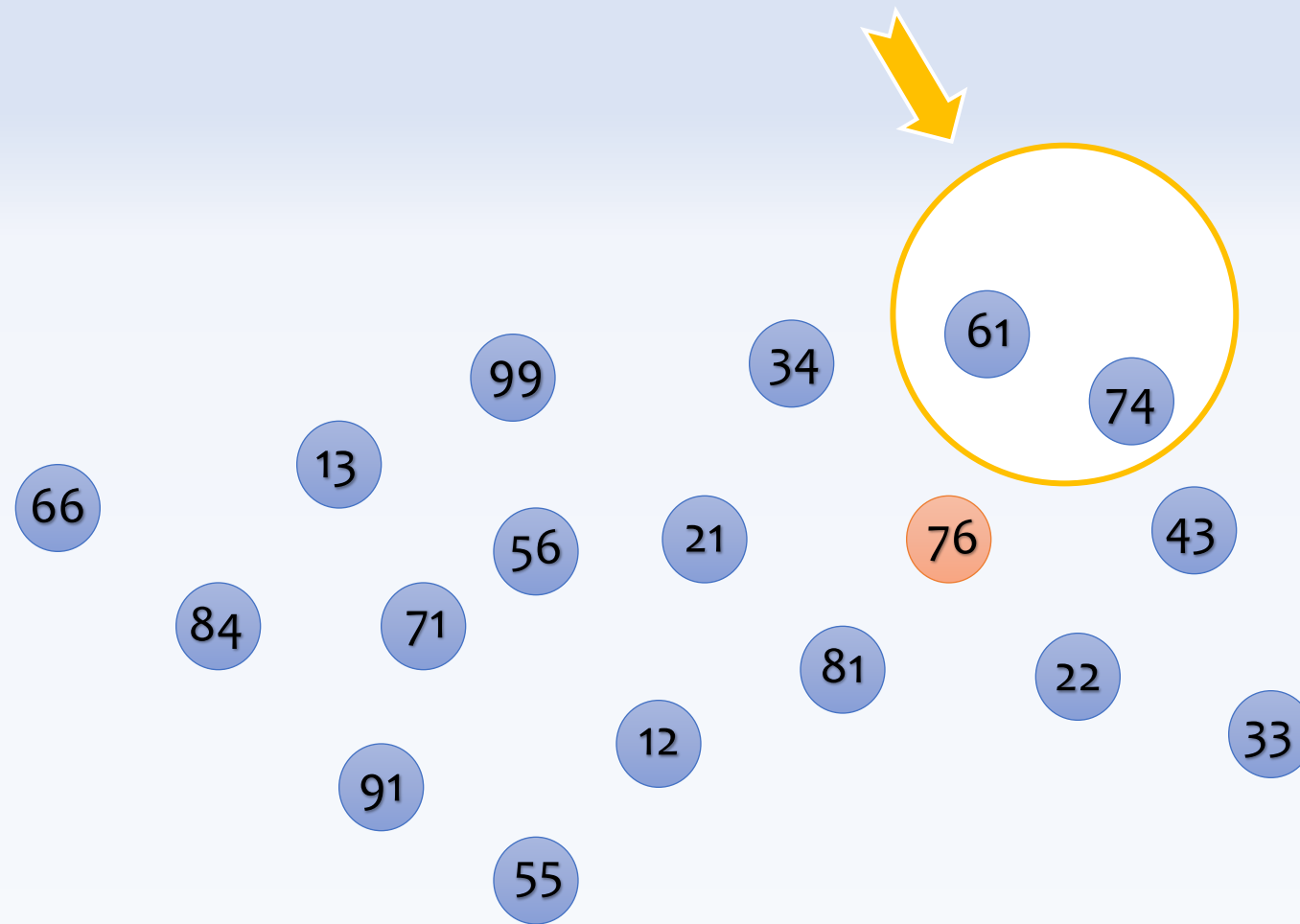
# Searching for 76



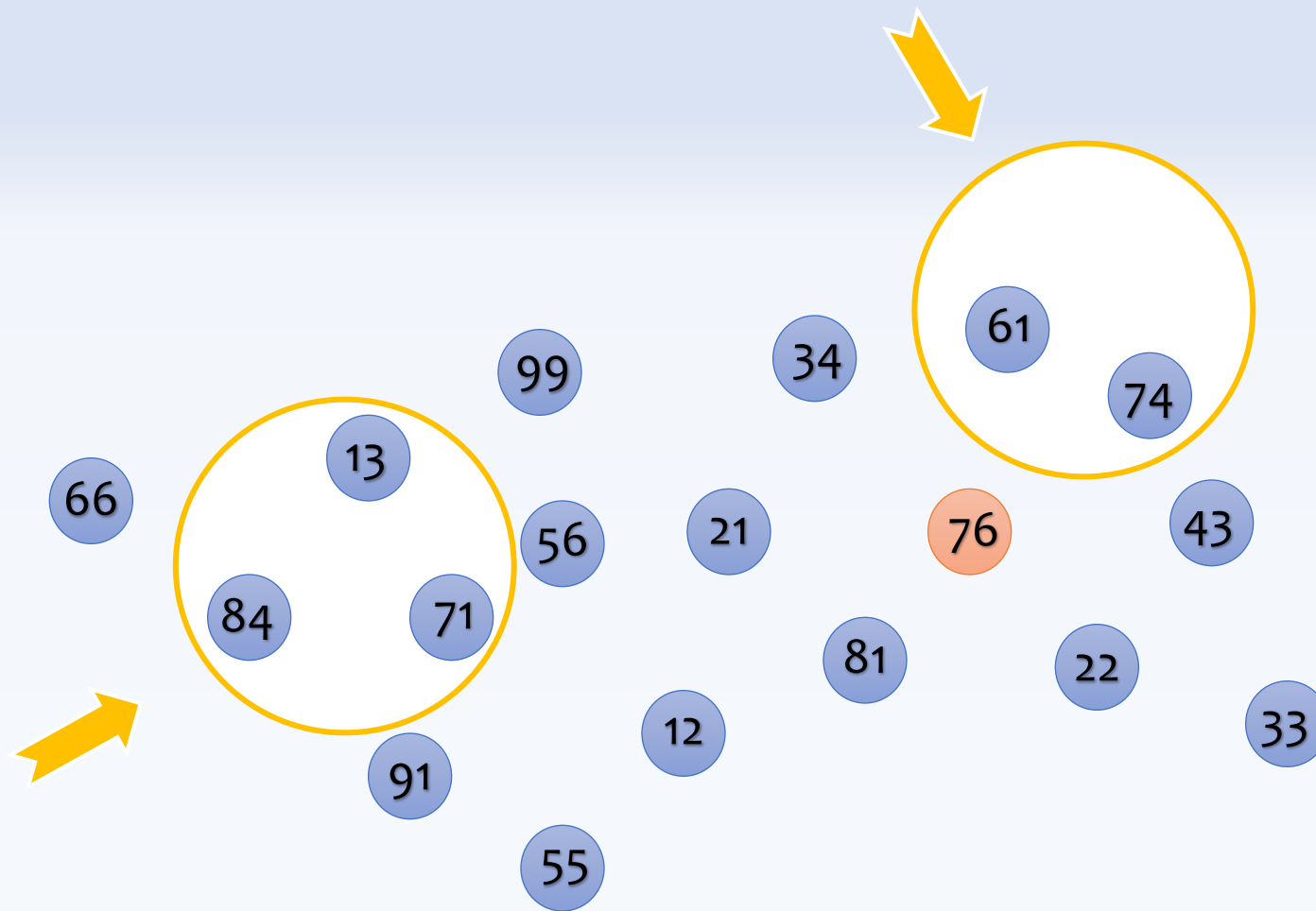
# Searching for 76



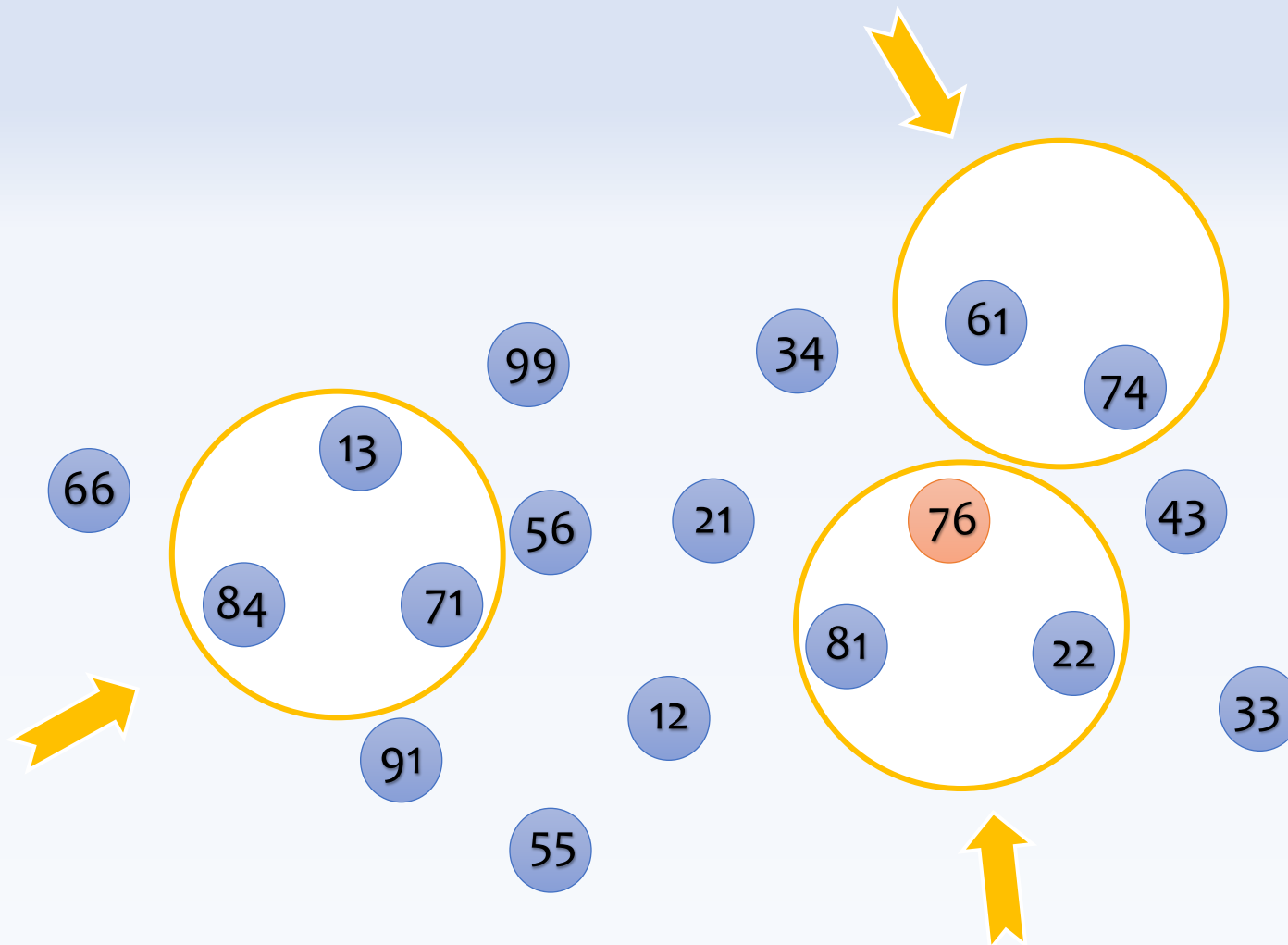
# Searching for 76



# Searching for 76

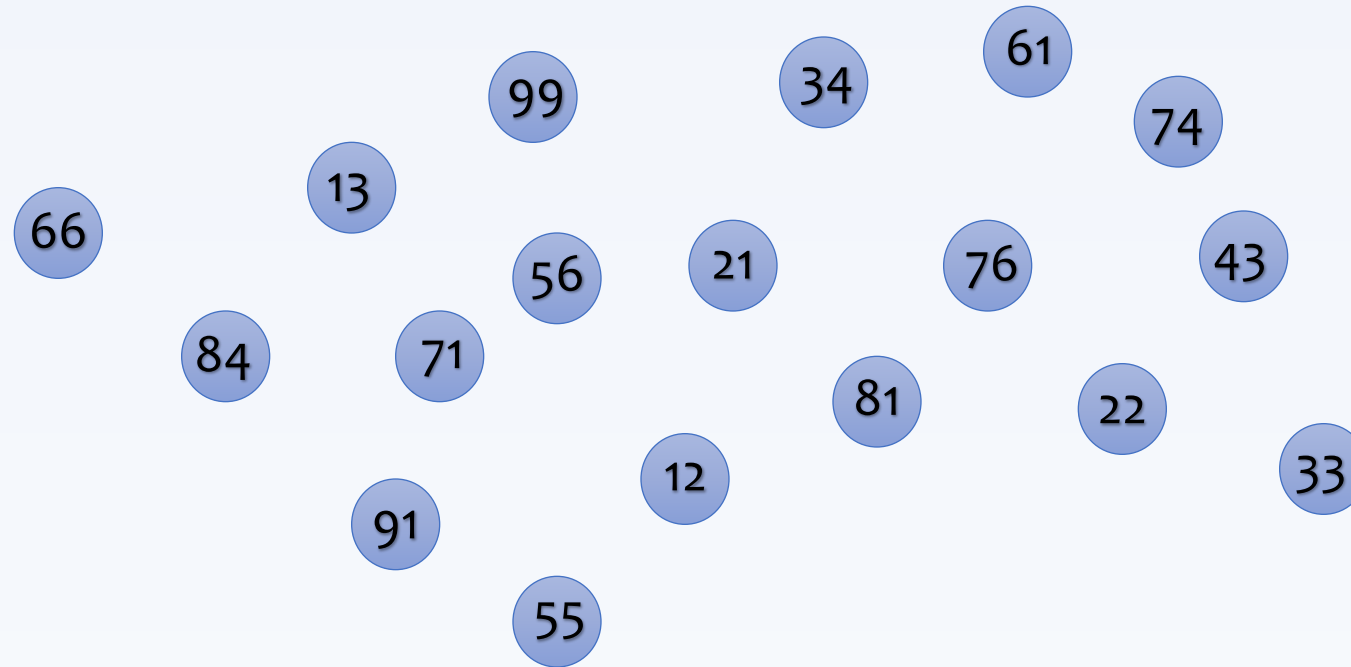


# Searching for 76



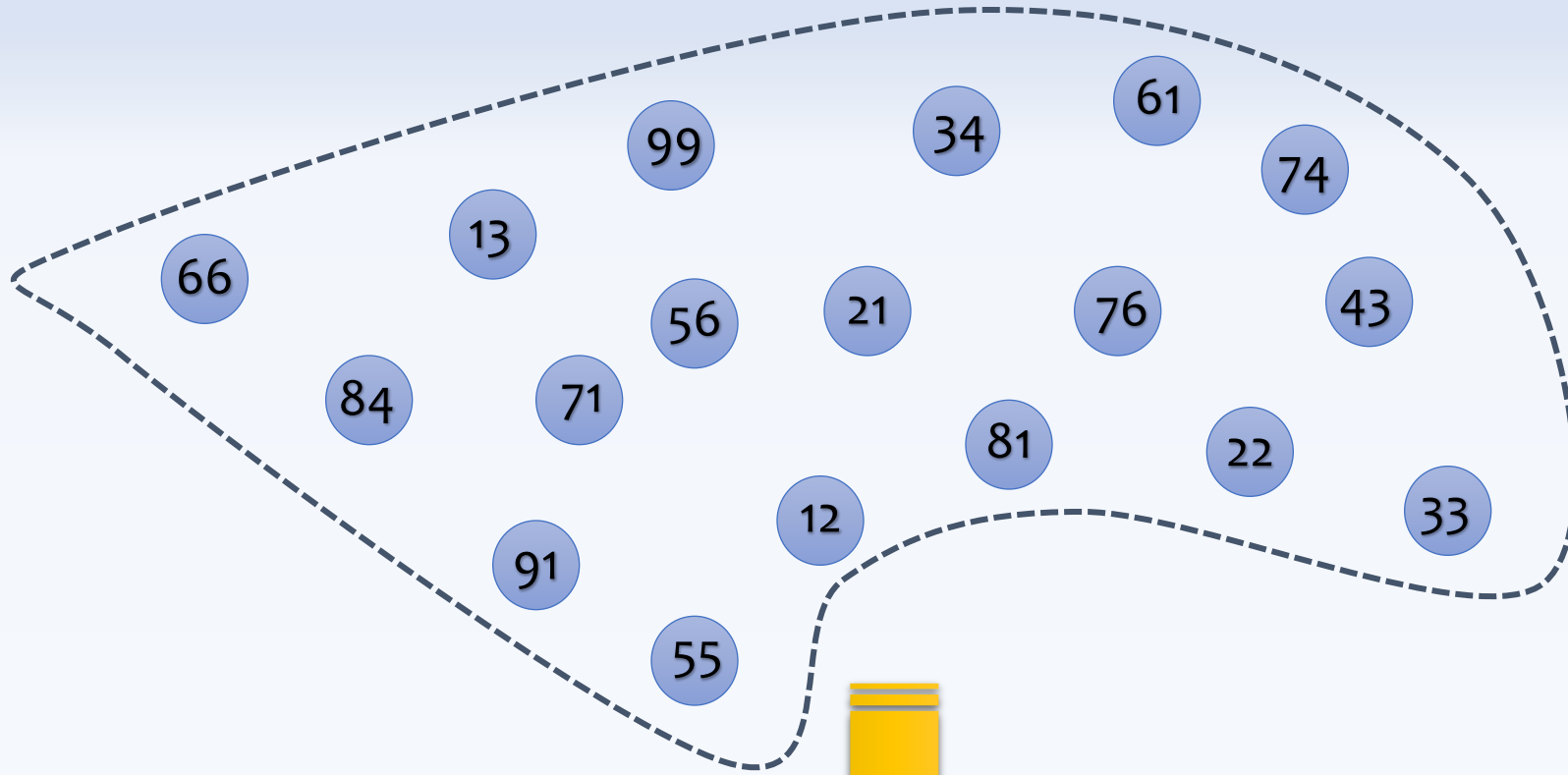
## Searching for 44?

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



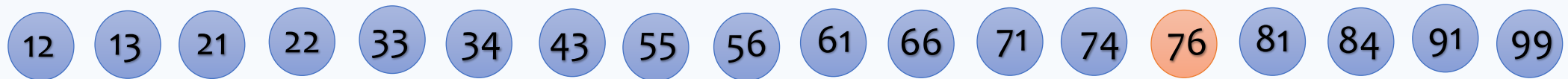


**Could we impose order to improve the search?**

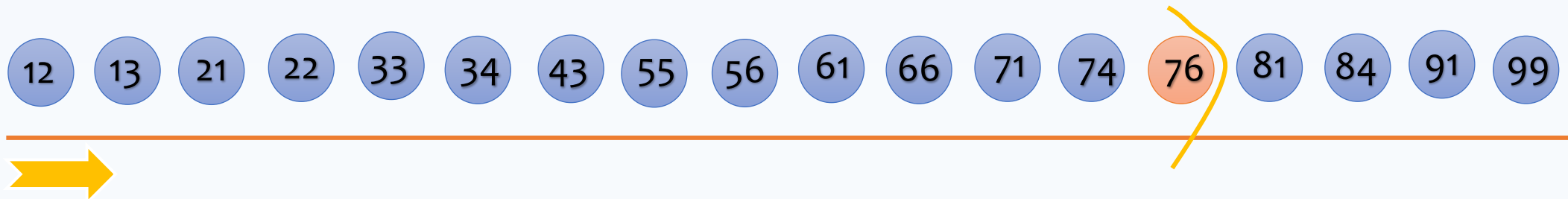


- 
- 12
  - 13
  - 21
  - 22
  - 33
  - 34
  - 43
  - 55
  - 56
  - 61
  - 66
  - 71
  - 74
  - 76
  - 81
  - 84
  - 91
  - 99
-

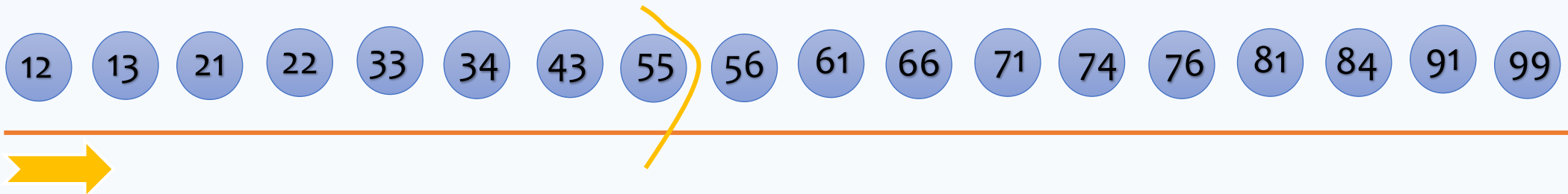
# Searching for 76



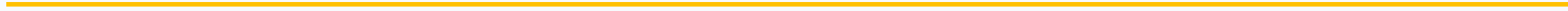
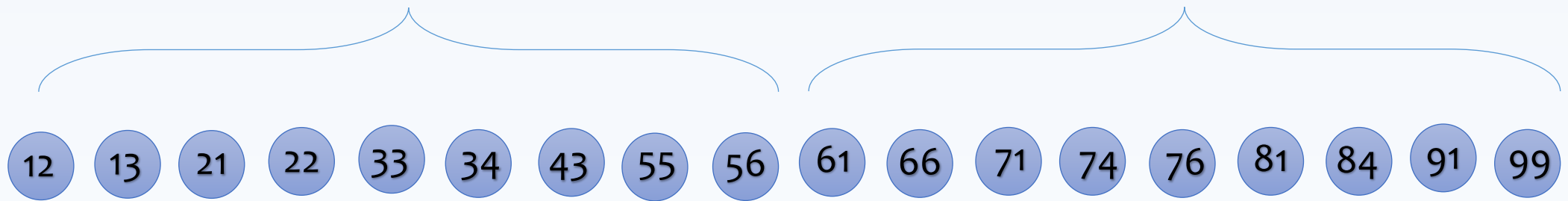
# Searching for 76

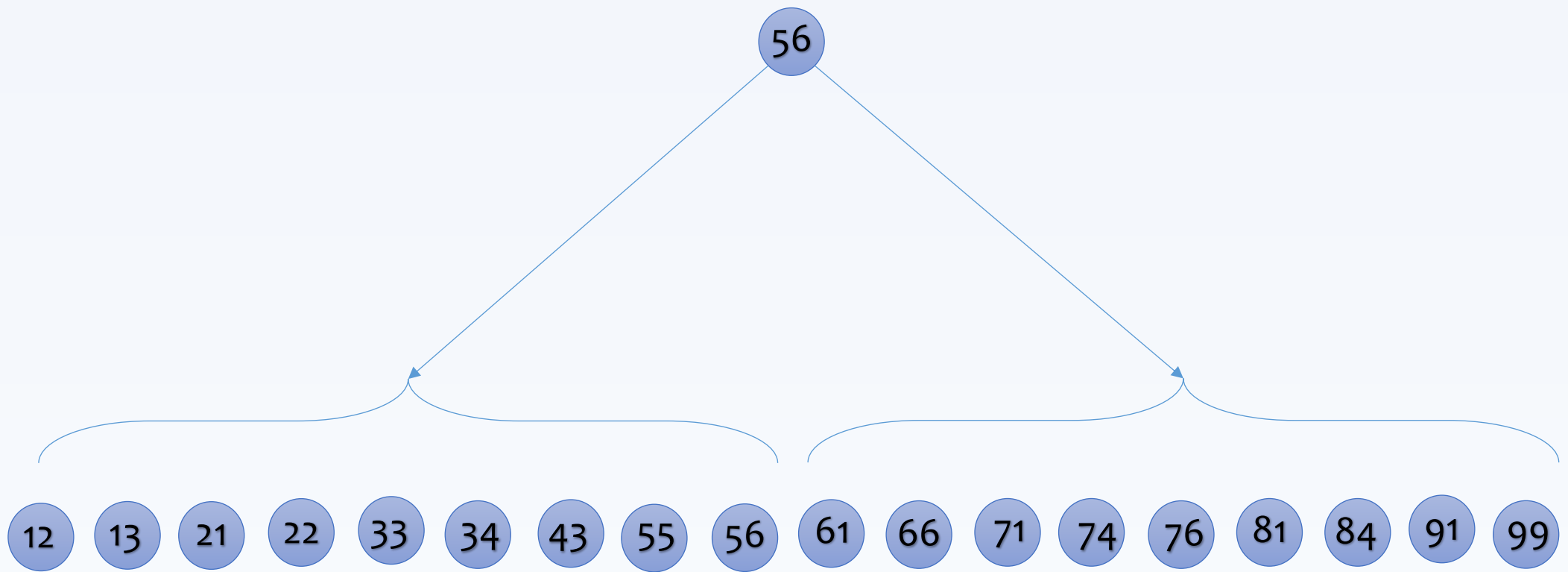


# Searching for 44? (could we have early termination?)



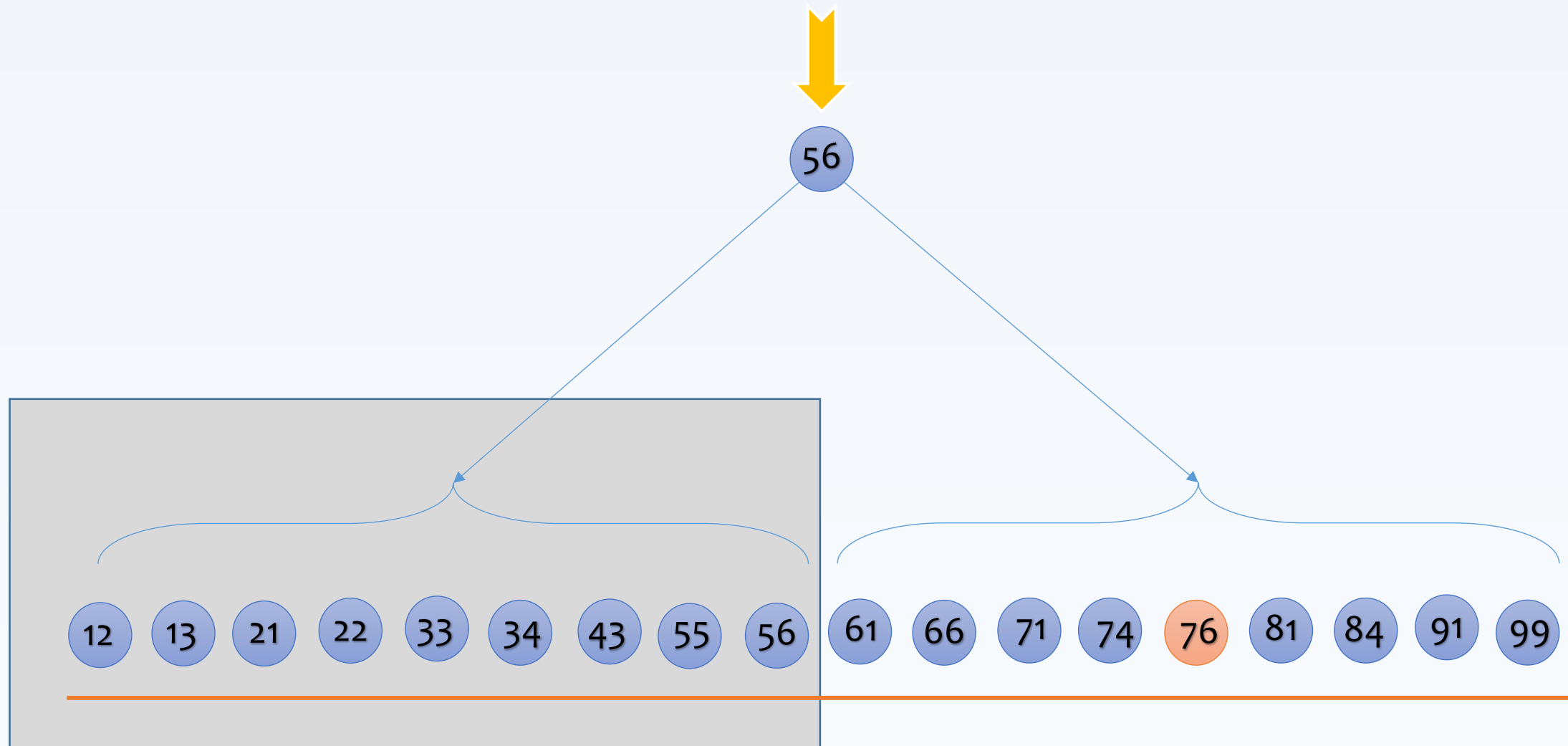
**Could we impose a structure to further improve the search?**



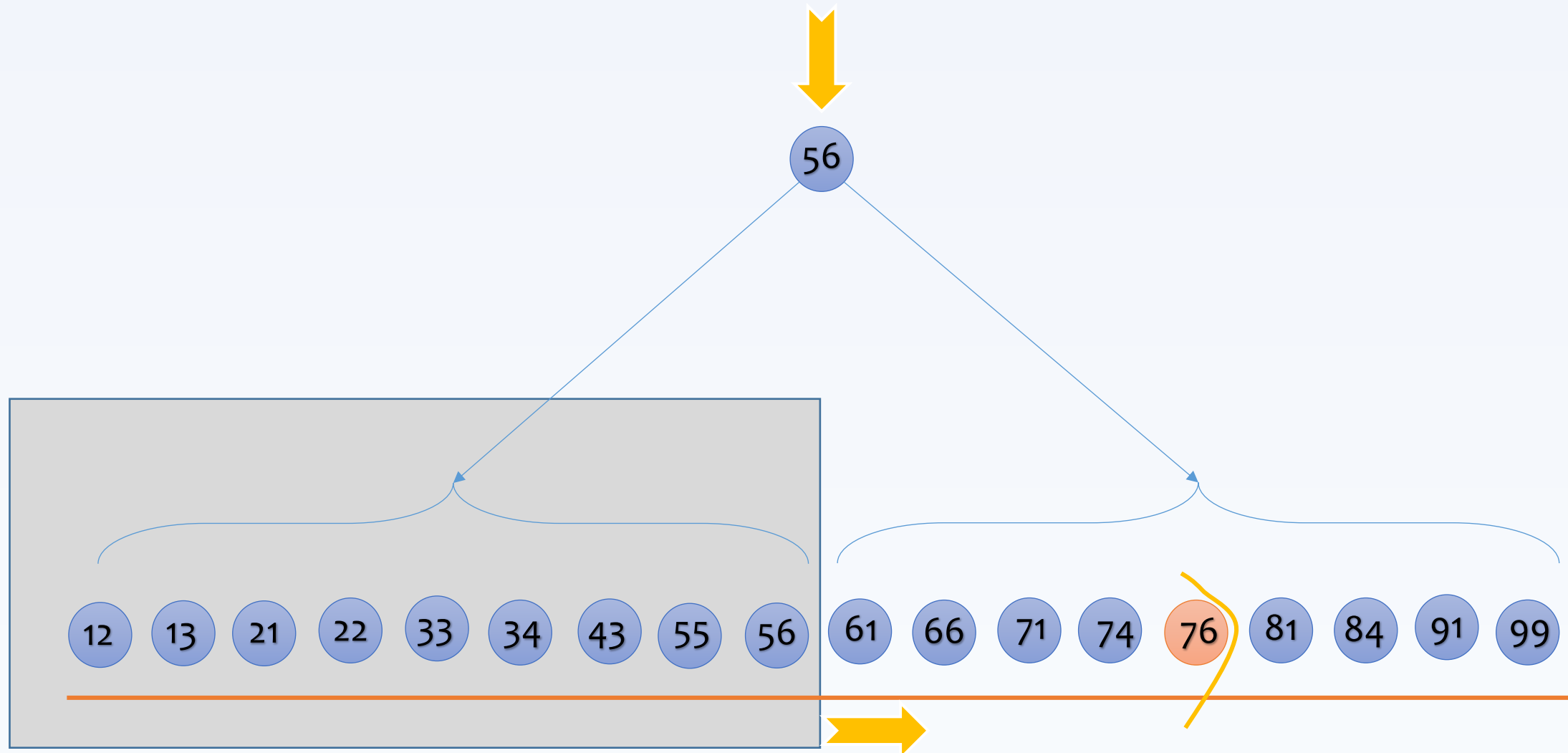


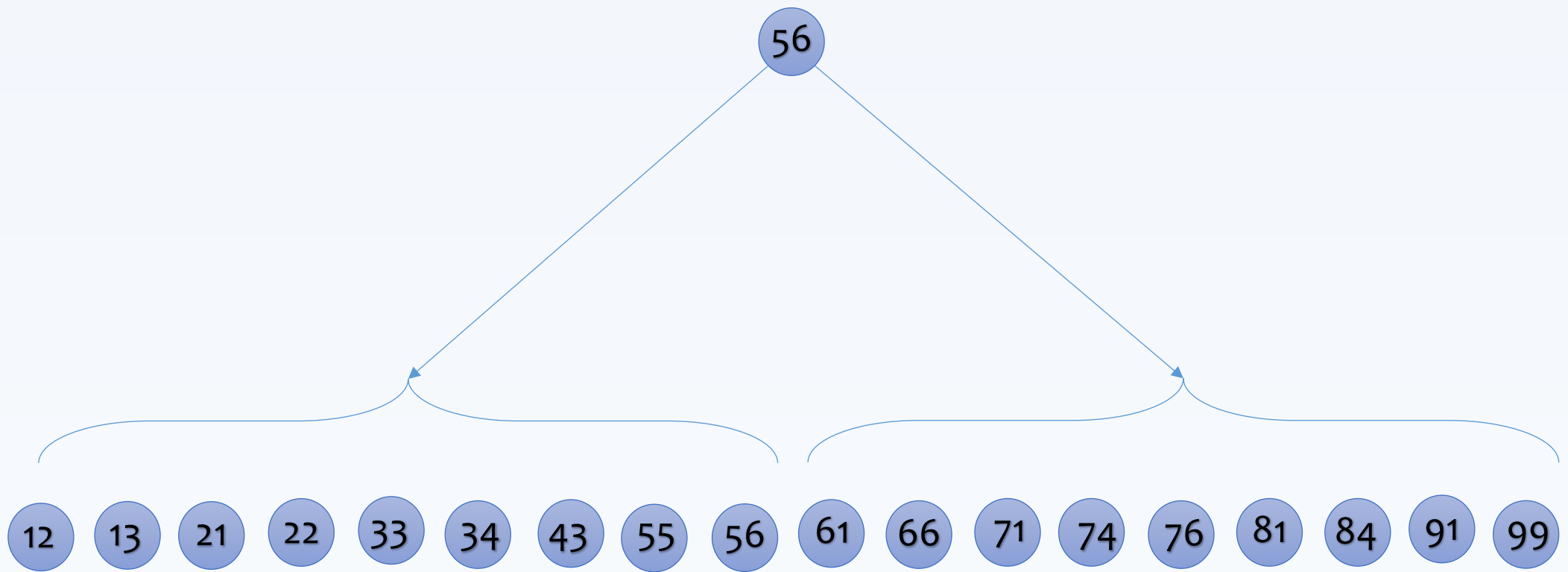


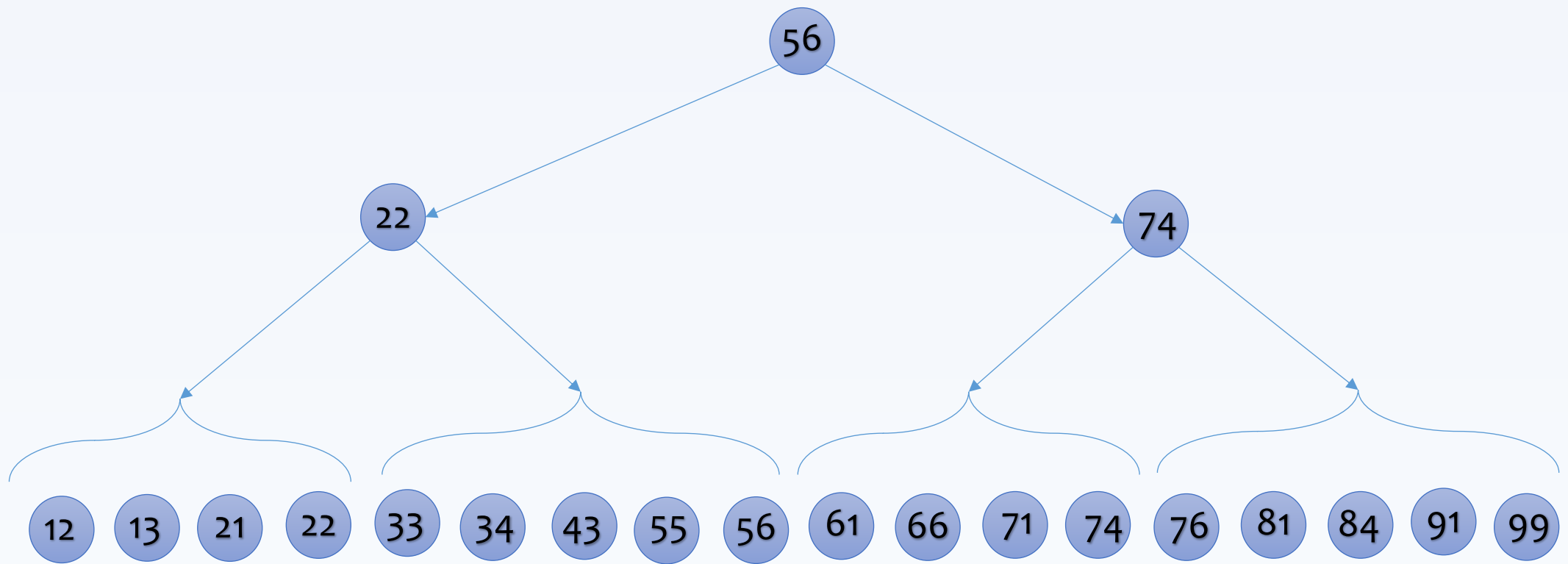
# Searching for 76



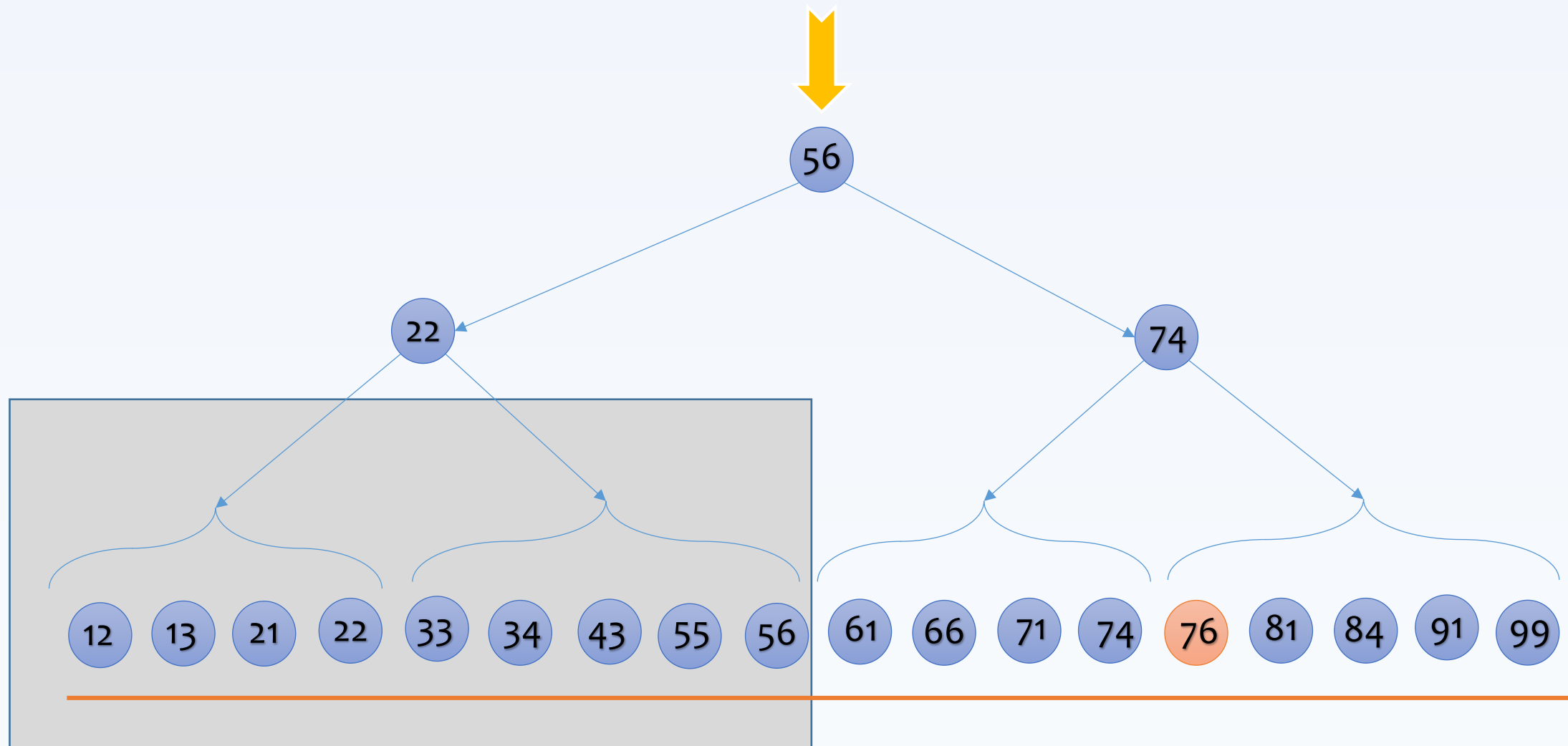
# Searching for 76



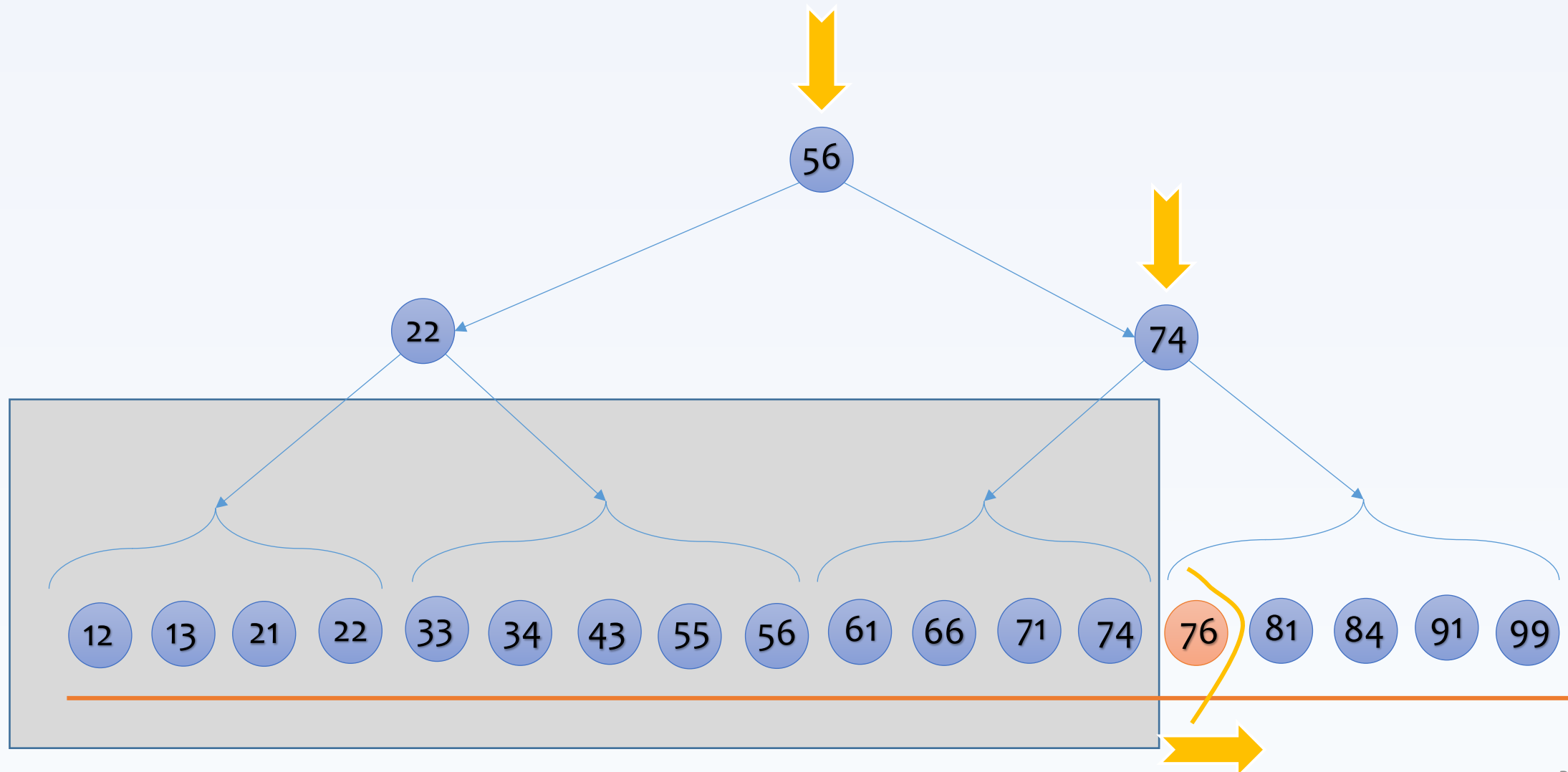




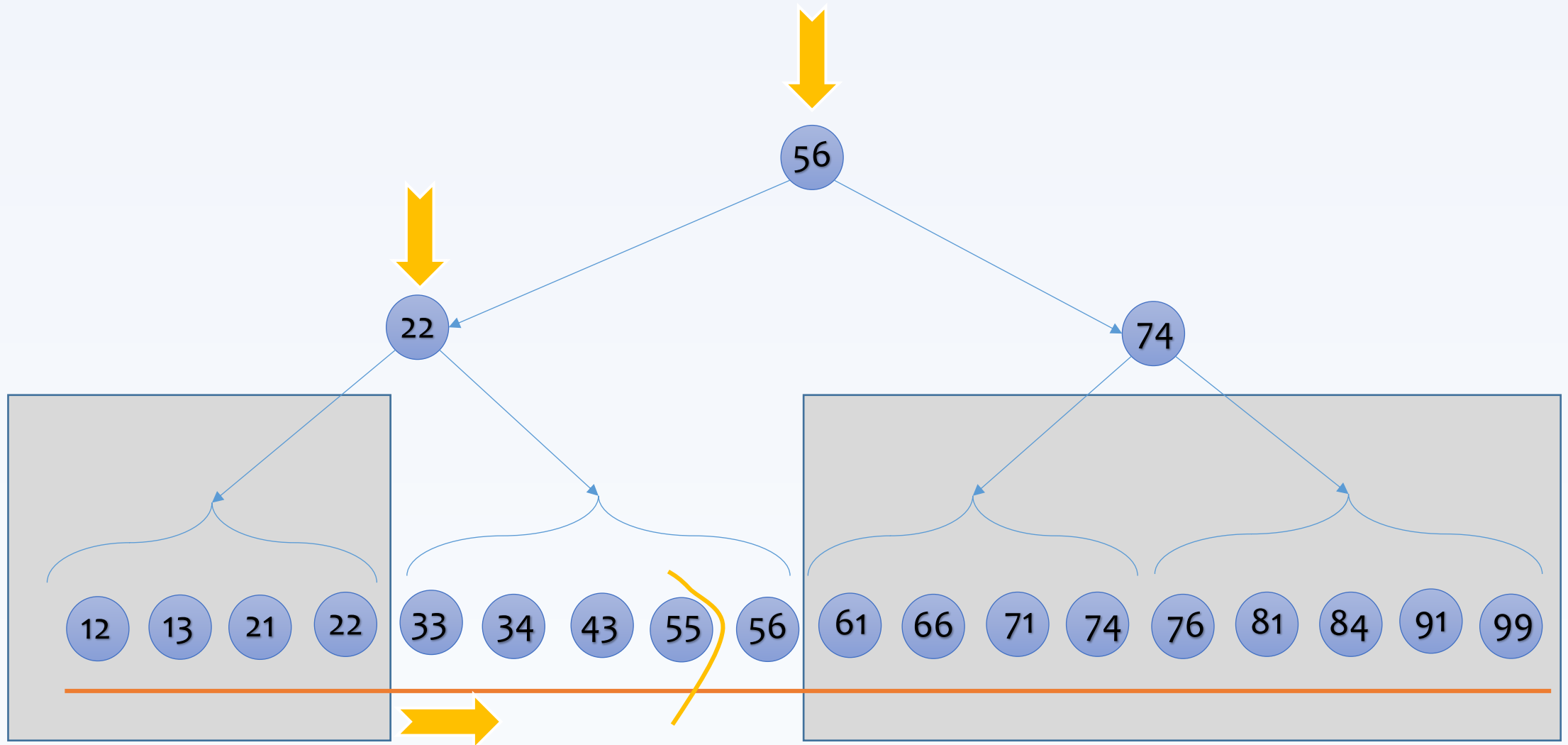
# Searching for 76



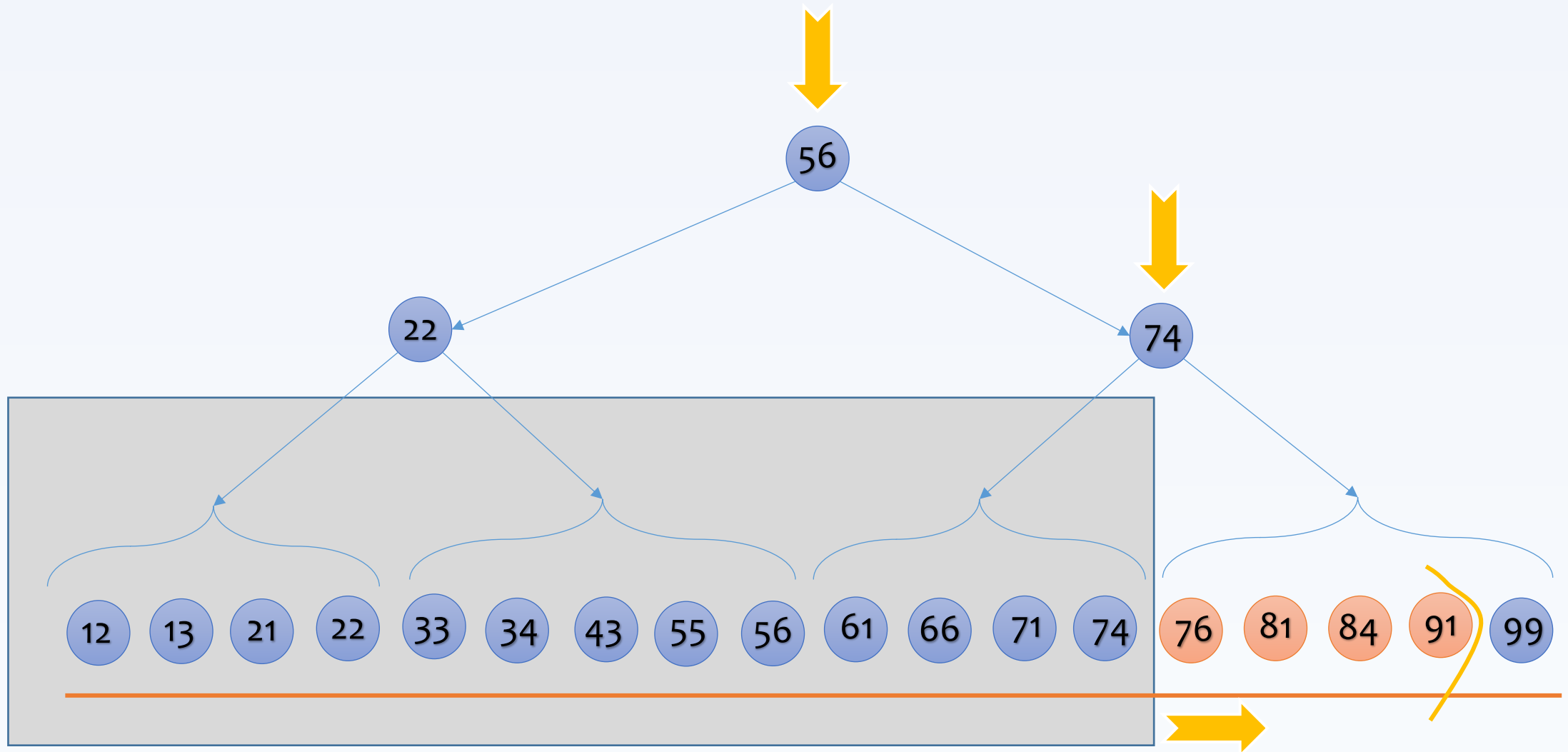
# Searching for 76



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



# Searching for 76-91





**Could we spread the data cleverly to improve the search?**

hashtable



bucket



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

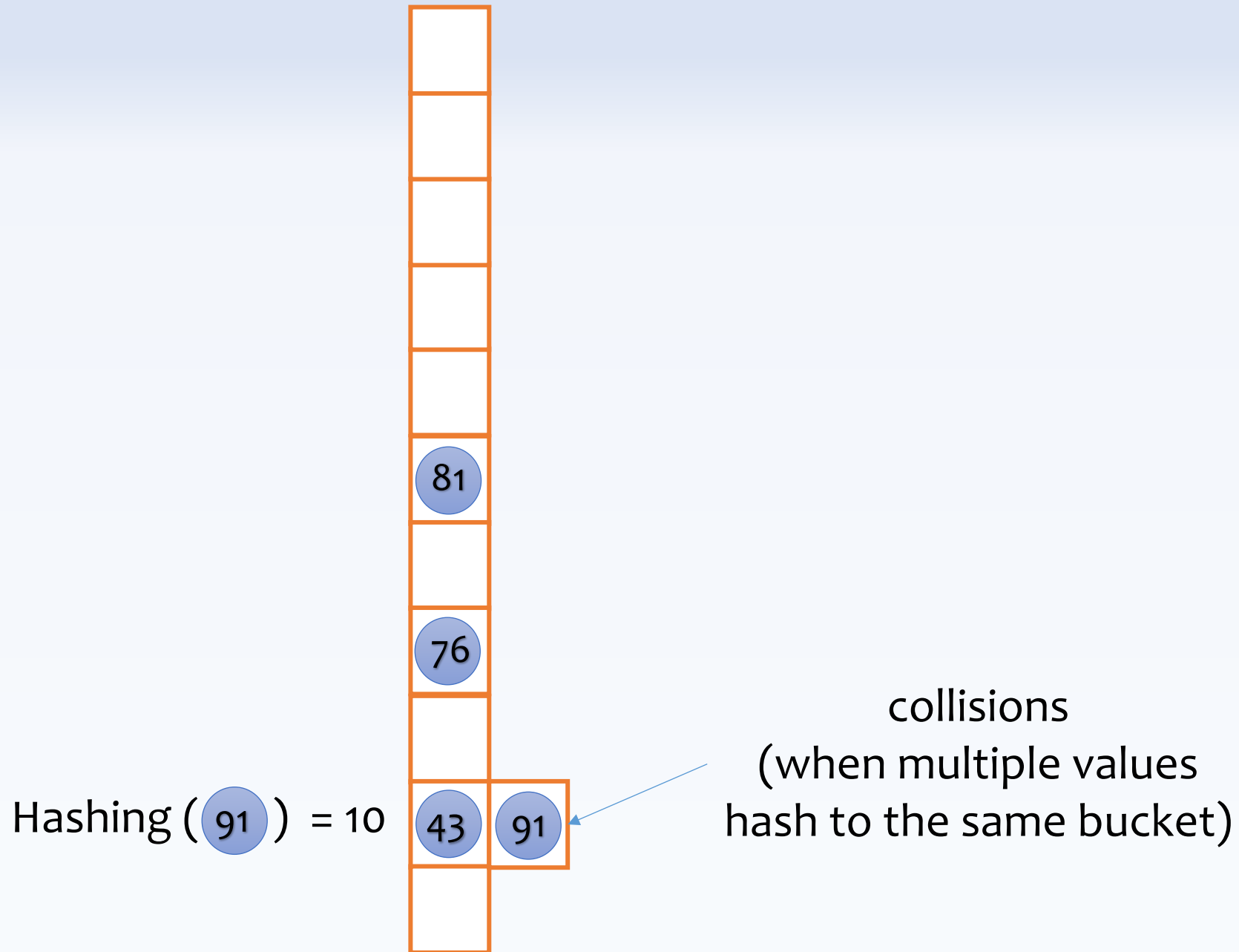


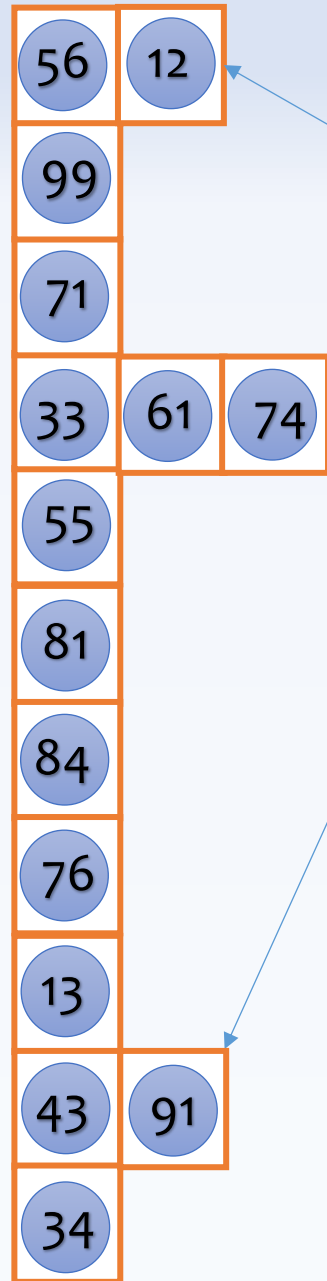
# Inserting 76



Hashing (76) = 8

# Inserting 91

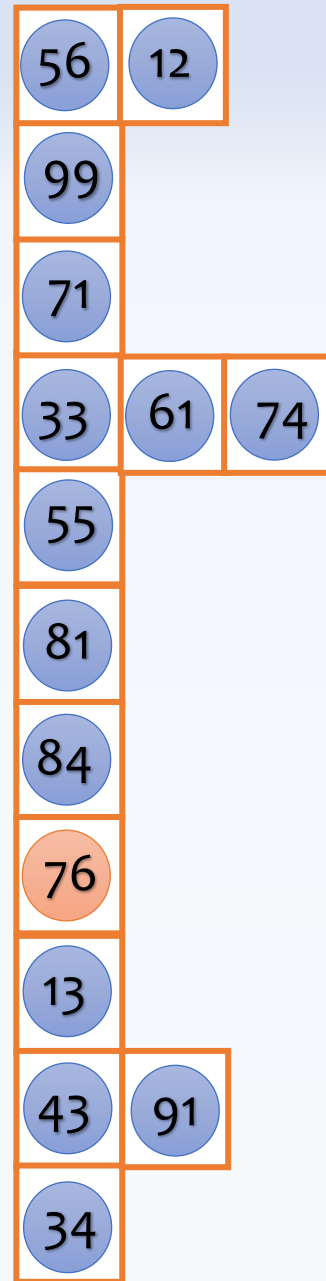




collisions  
(when multiple values  
hash to the same bucket)

# Searching for 76

(now we can have constant lookup cost)

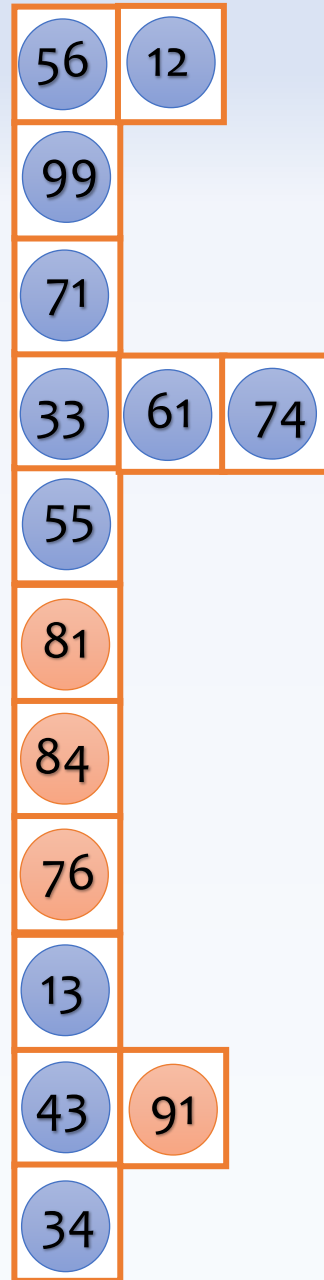


Hashing (76) = 8



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?

Hashing (76) = 8

Hashing (77) = 1

Hashing (78) = 3

⋮

Hashing (81) = 6

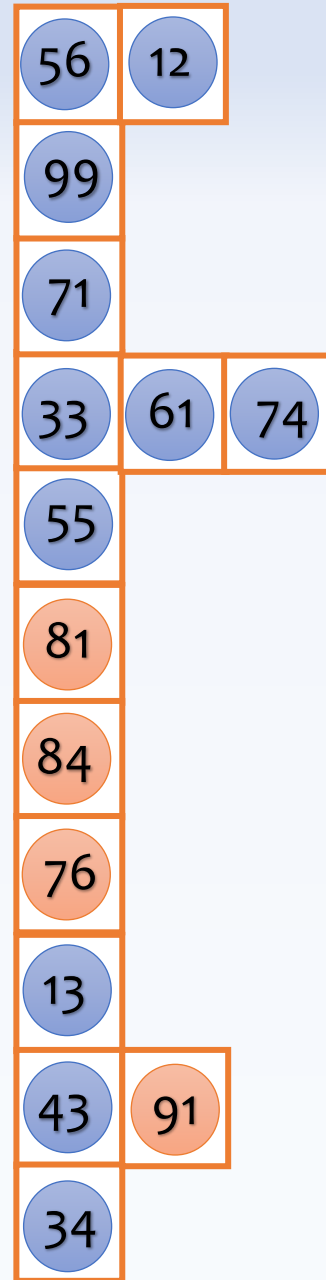
⋮

Hashing (84) = 7

⋮

Hashing (90) = 8

Hashing (91) = 10



## 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

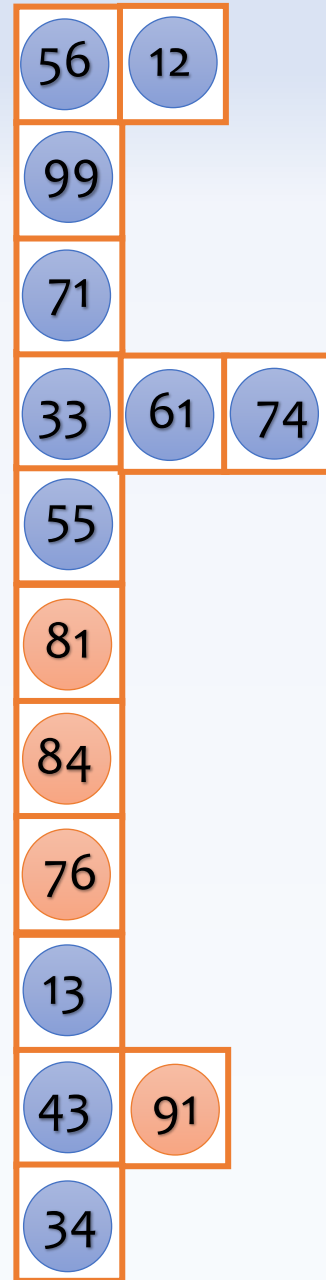
⋮

Hashing (84) = 7

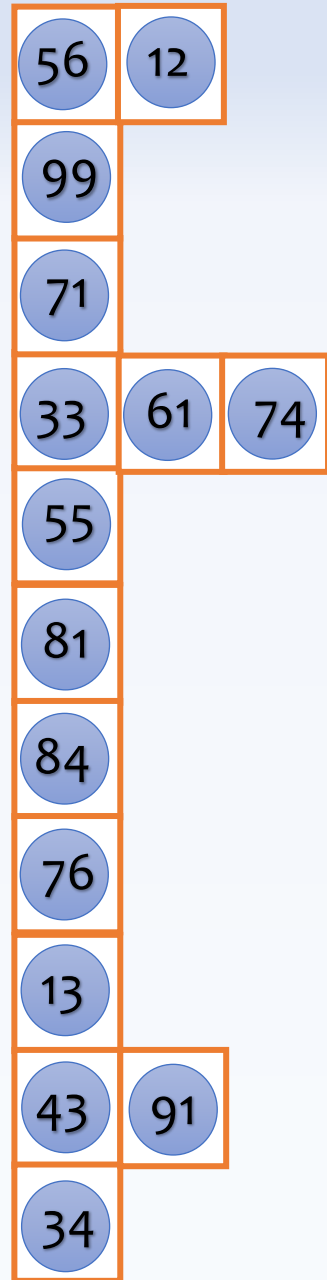
⋮

Hashing (90) = 8

Hashing (91) = 10

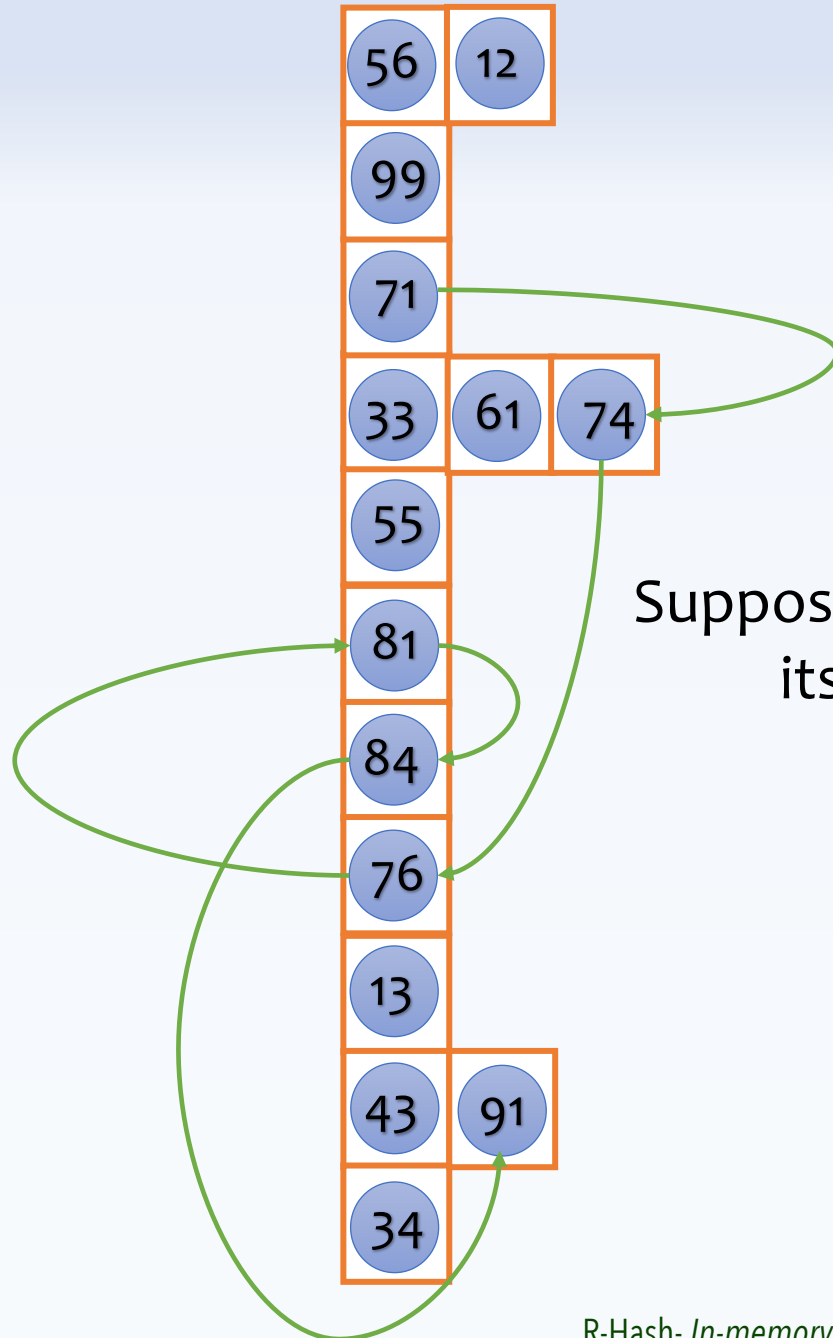


**Could we rethink the design to search  
for a range of values efficiently?**



Let's promote a subset of values as seeds



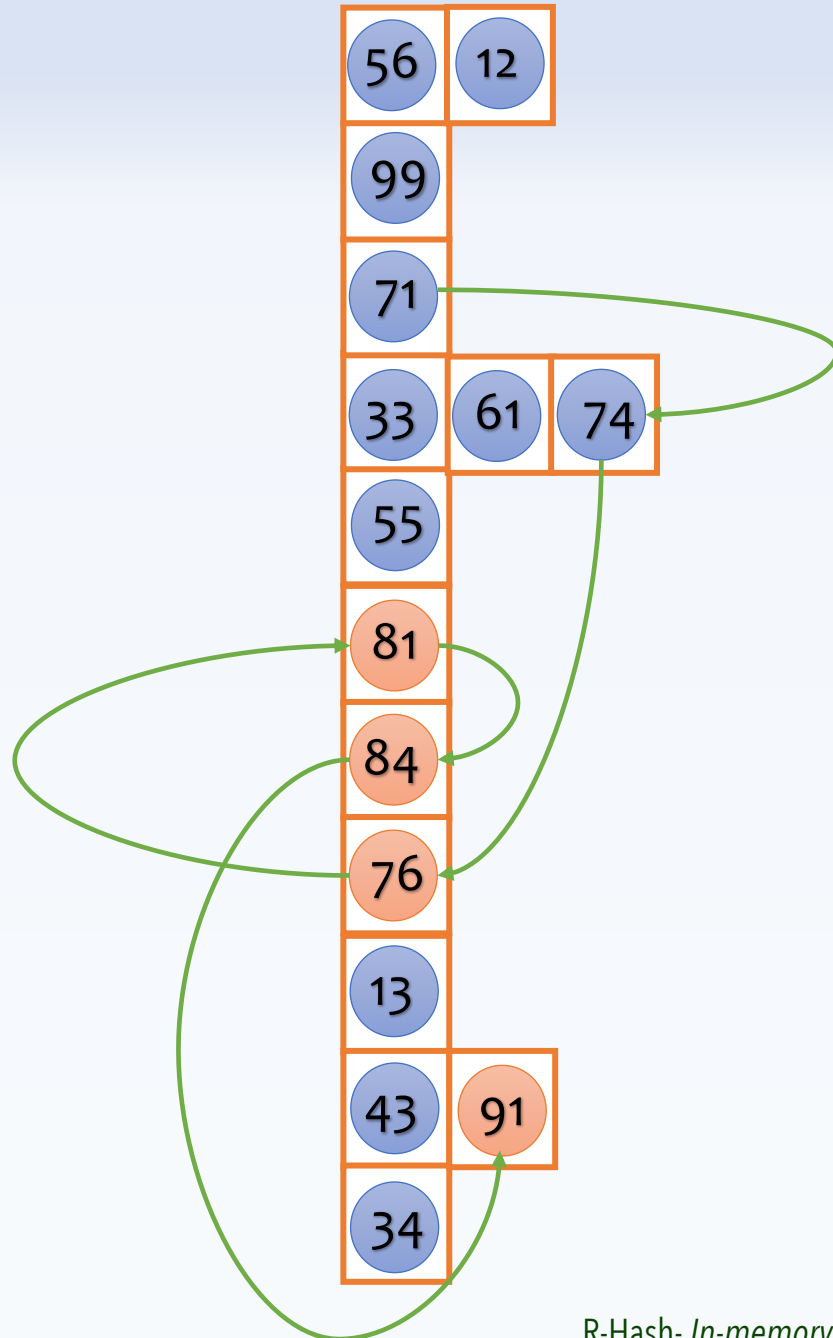


Let's promote a subset of values as seeds

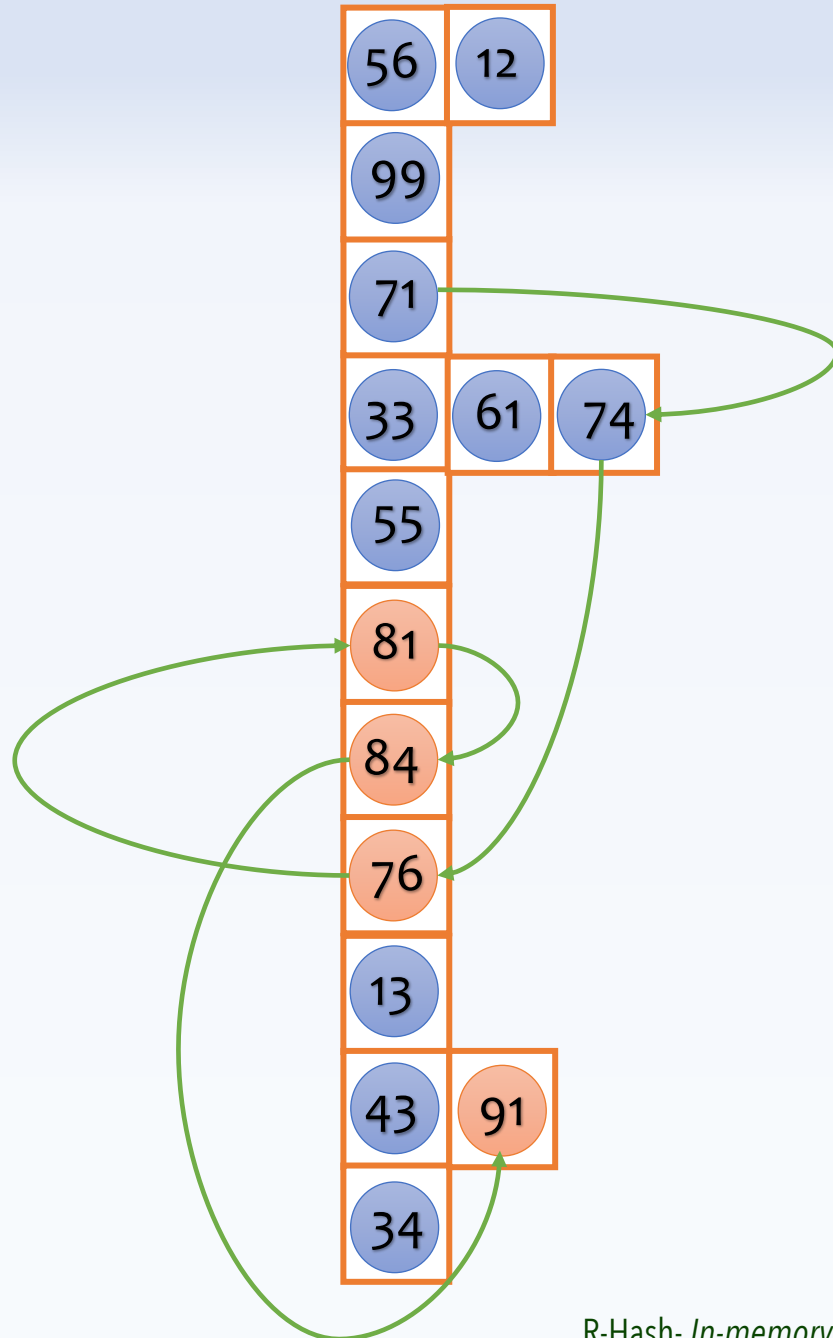


Suppose every value points to its next larger value

# Searching for 76-91



# Searching for 76-91



sorted seeds

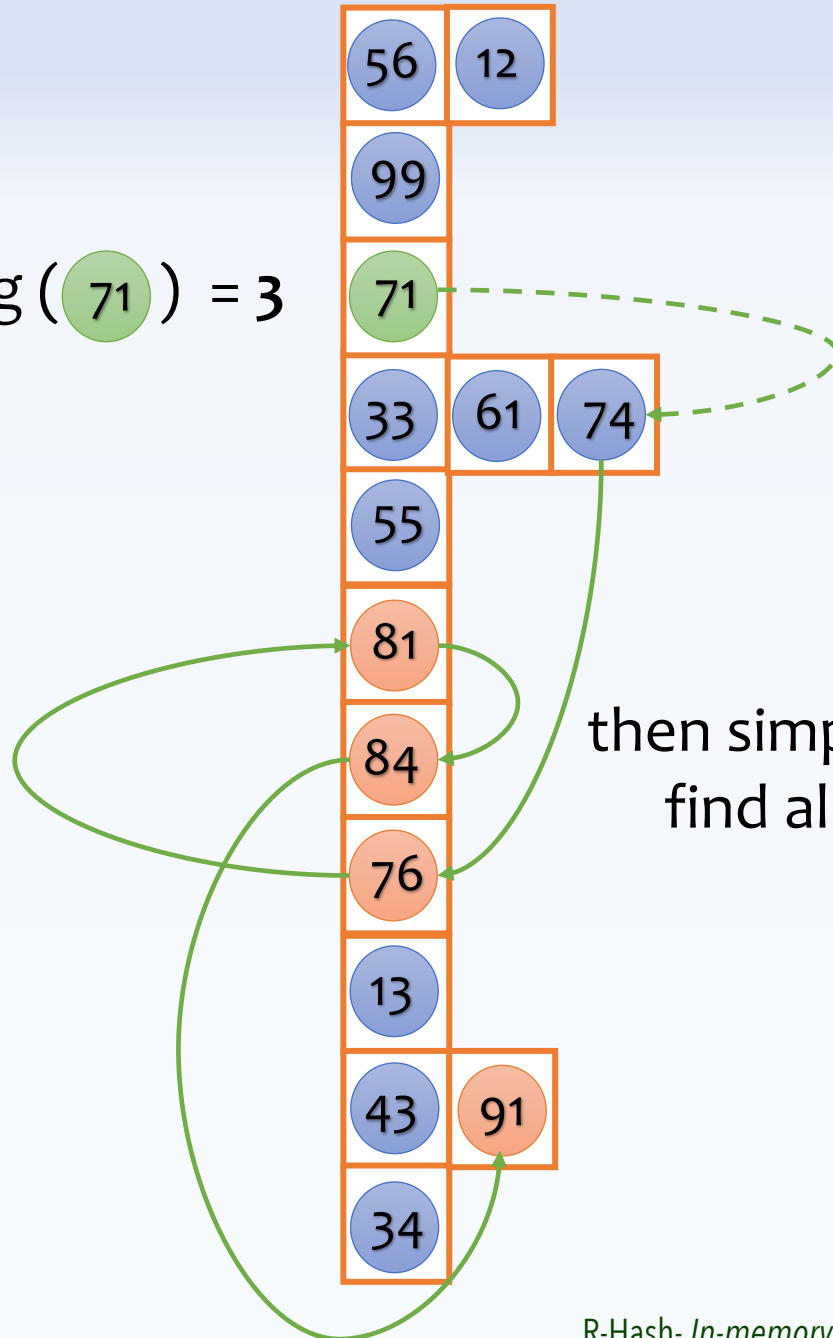


Find the largest seed smaller than 76: 71



# Searching for 76-91

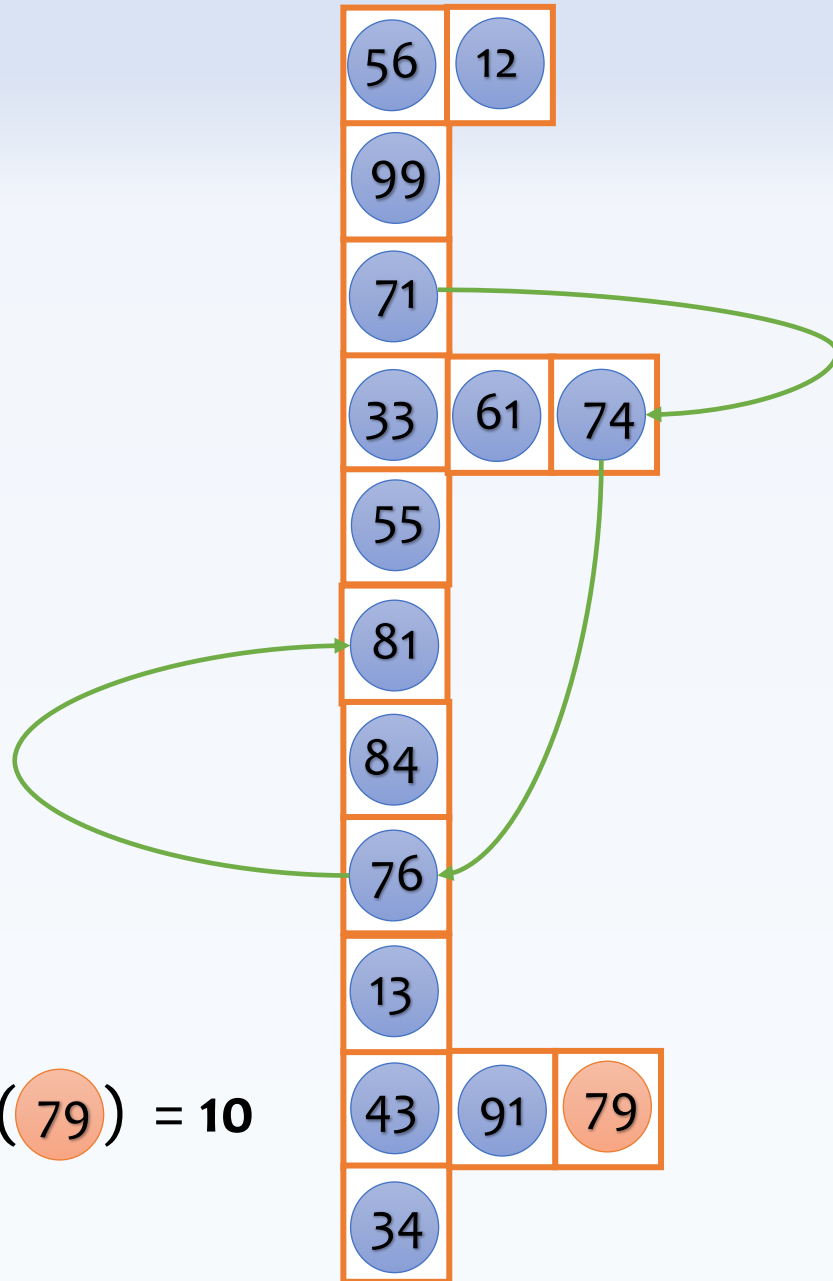
Hashing (71) = 3



Find the largest seed smaller than 76: 71

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

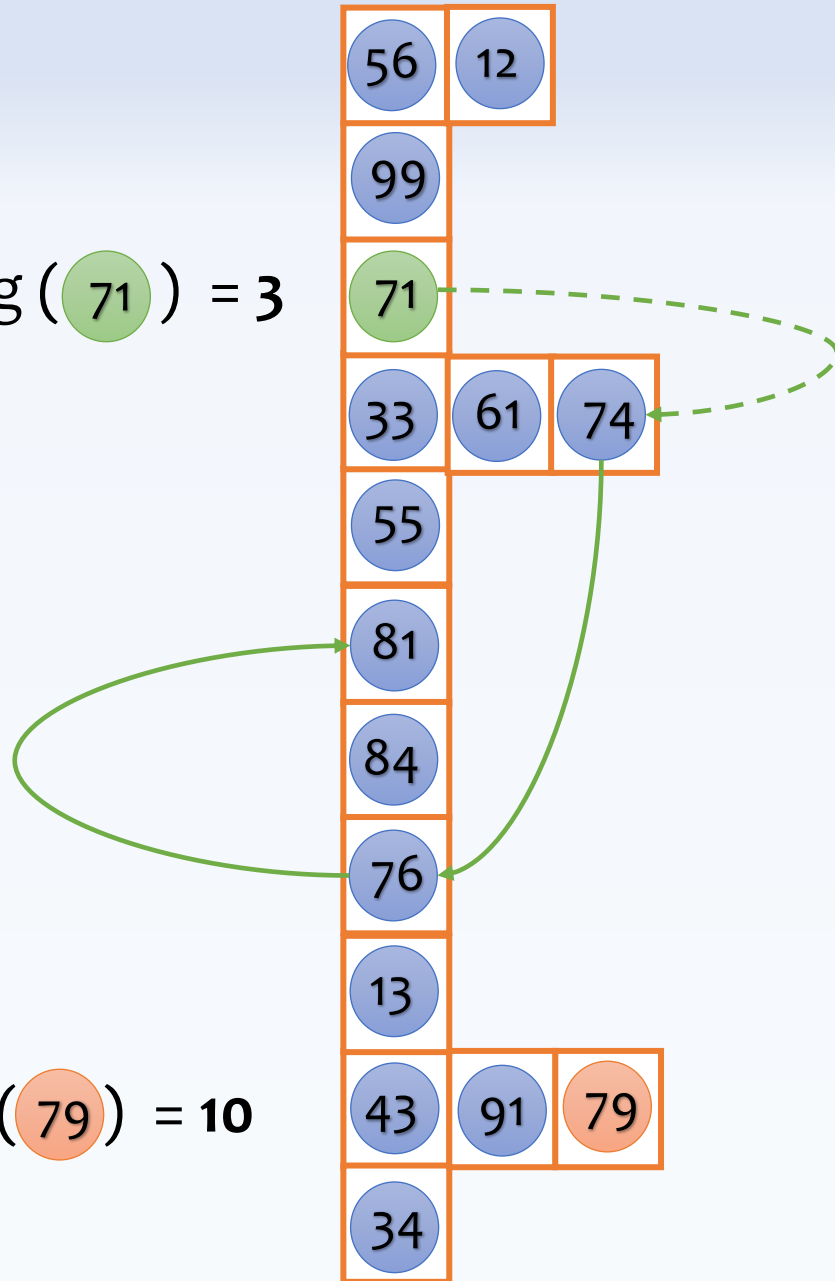
# Inserting 79



Hashing (79) = 10

# Inserting 79

Hashing (71) = 3



sorted seeds



Find the largest seed smaller than 79: 71

Hashing (79) = 10

# Inserting 79

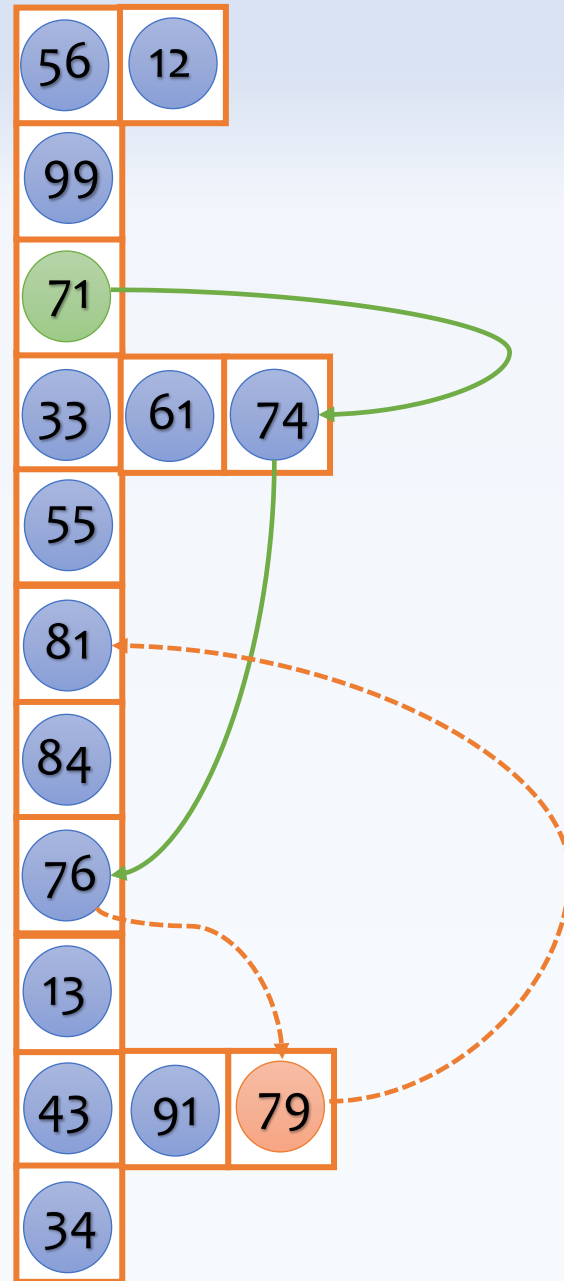
Hashing (71) = 3

sorted seeds



Find the largest seed smaller than 79: 71

adjust the pointers accordingly

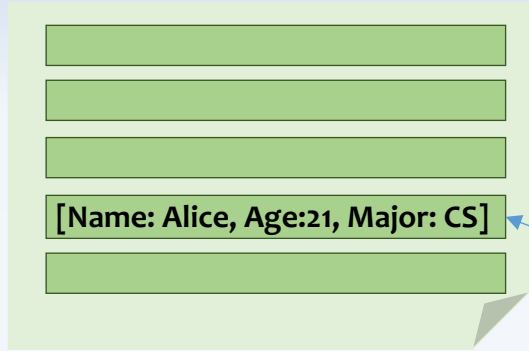


Hashing (79) = 10

# **Database Storage Layouts**

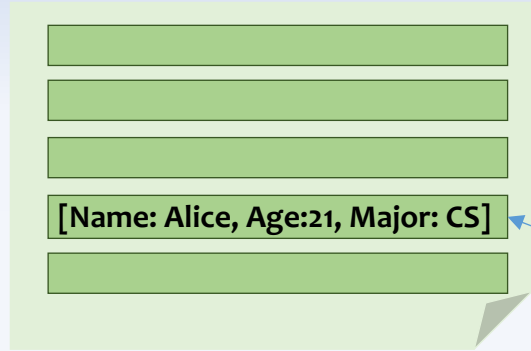
**(how often do we need an index for range queries?)**

database pages  
(containing a set of records)

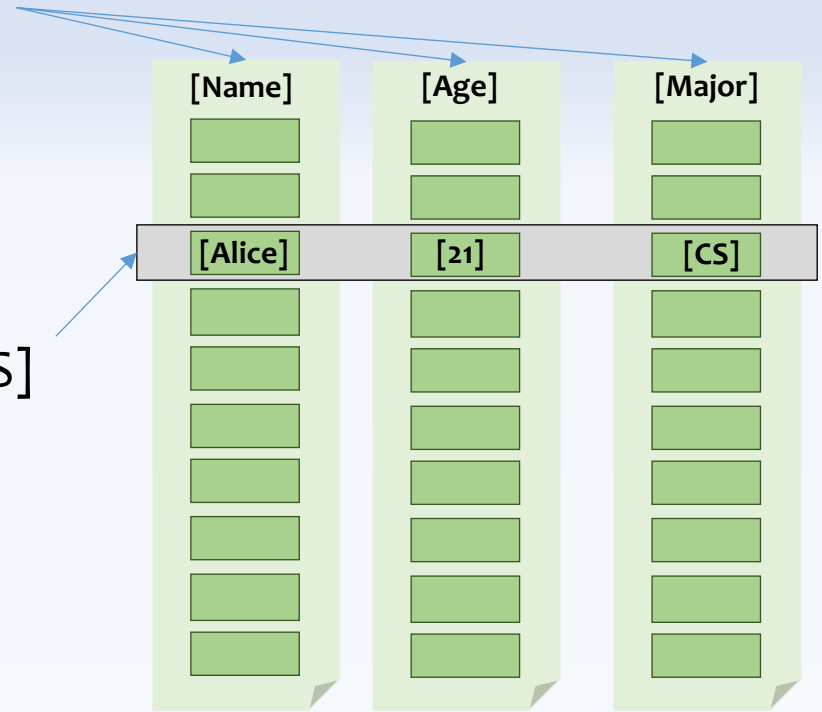


a database record, e.g.,  
[Name: Alice, Age:21, Major: CS]

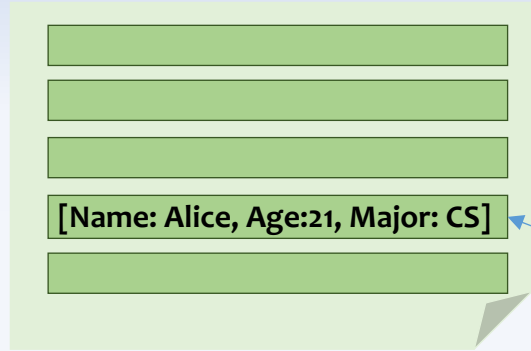
database pages  
(containing a set of records)



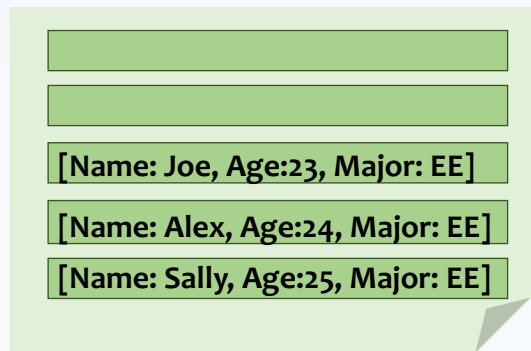
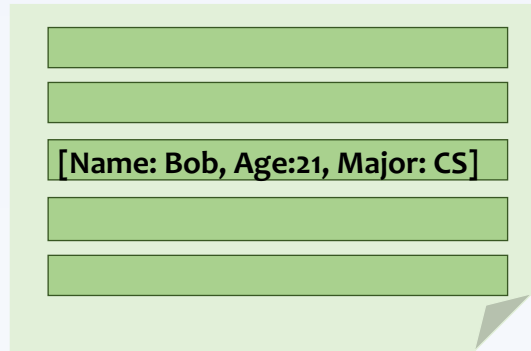
a database record, e.g.,  
[Name: Alice, Age:21, Major: CS]



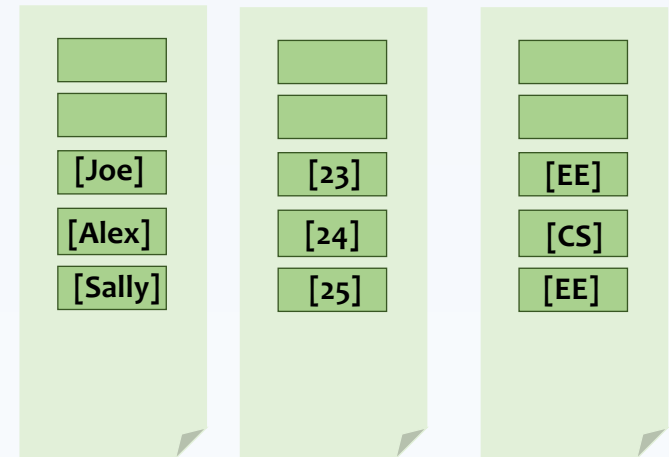
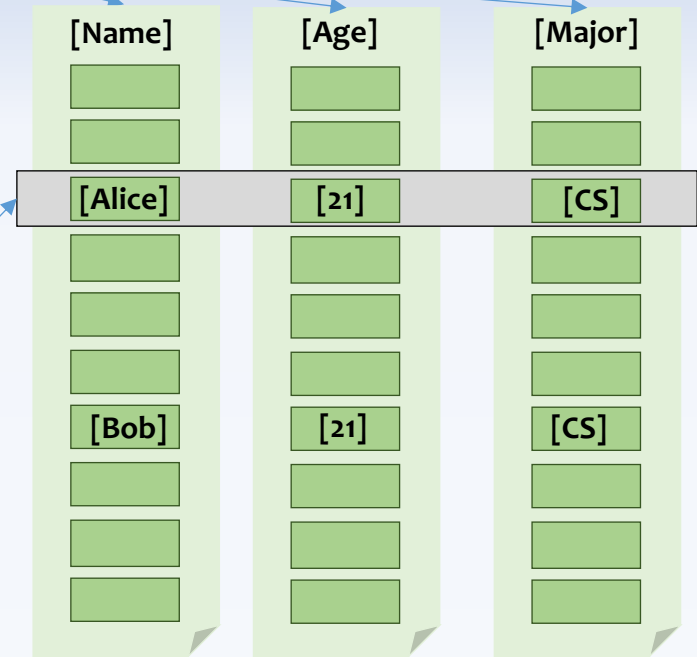
database pages  
(containing a set of records)



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



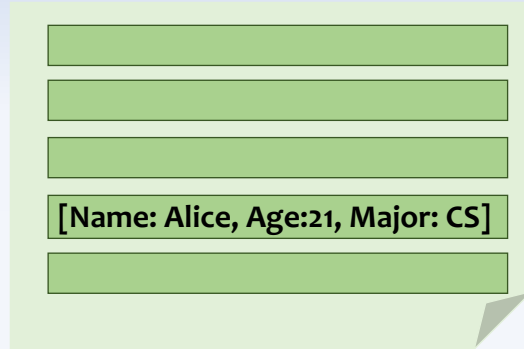
Row-based Layout



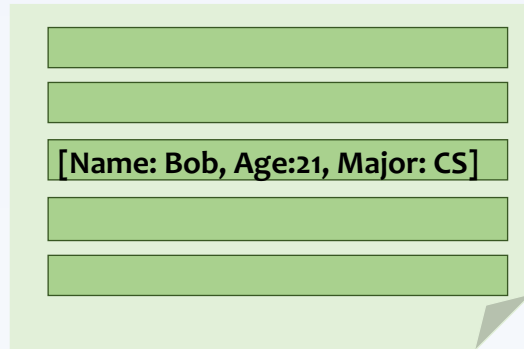
Column-based Layout



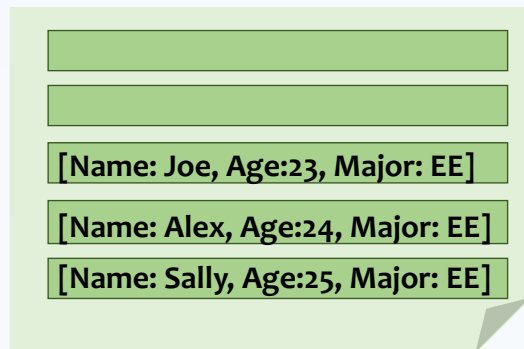
# Searching for all students between the age of 21 to 24 (returns many students)



A row-based layout for a single student record. It consists of five horizontal green boxes. The third box from the top contains the text "[Name: Alice, Age:21, Major: CS]".

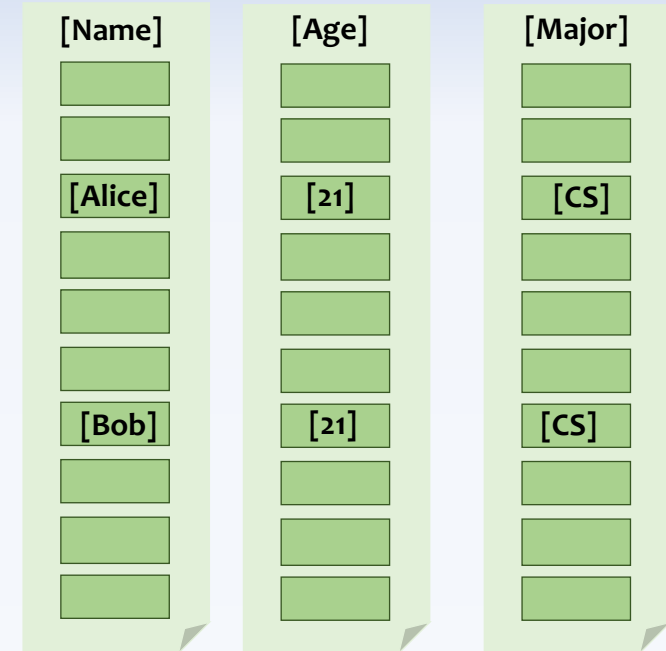


A row-based layout for a single student record. It consists of five horizontal green boxes. The third box from the top contains the text "[Name: Bob, Age:21, Major: CS]".

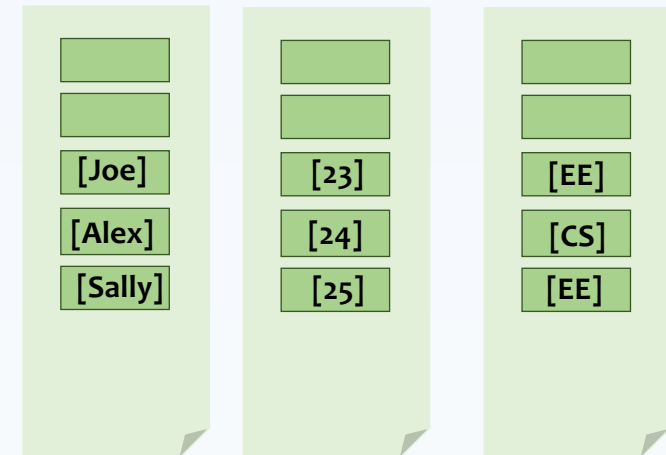


A row-based layout for three student records stacked vertically. Each record consists of five horizontal green boxes. The third box of each record contains the following text: "[Name: Joe, Age:23, Major: EE]", "[Name: Alex, Age:24, Major: EE]", and "[Name: Sally, Age:25, Major: EE]".

Row-based Layout



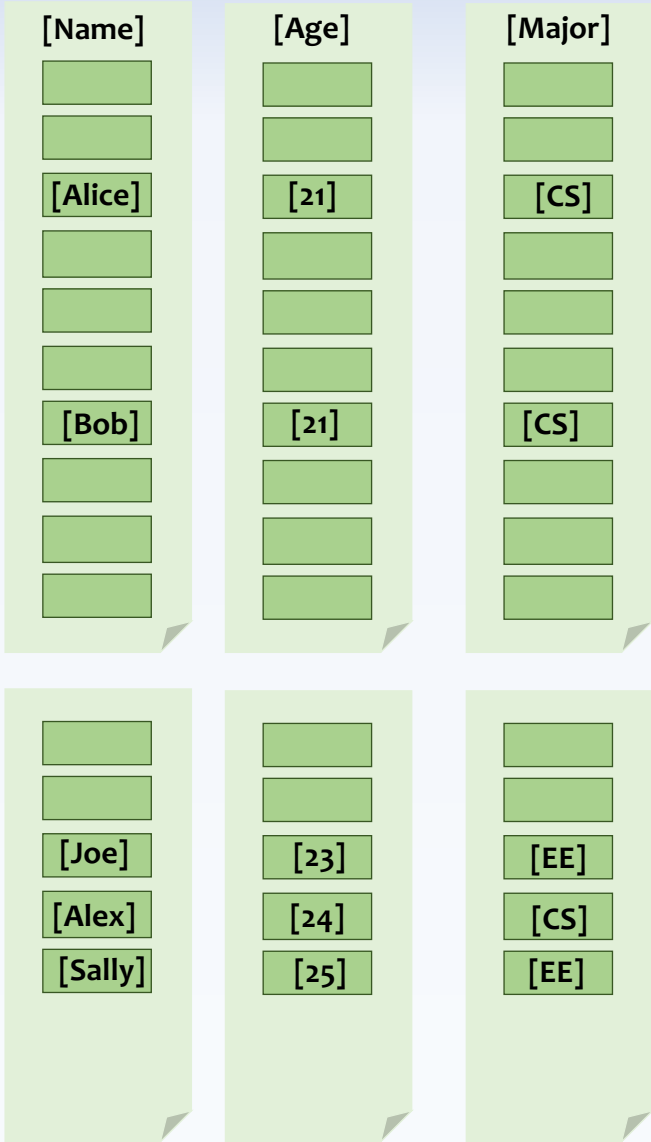
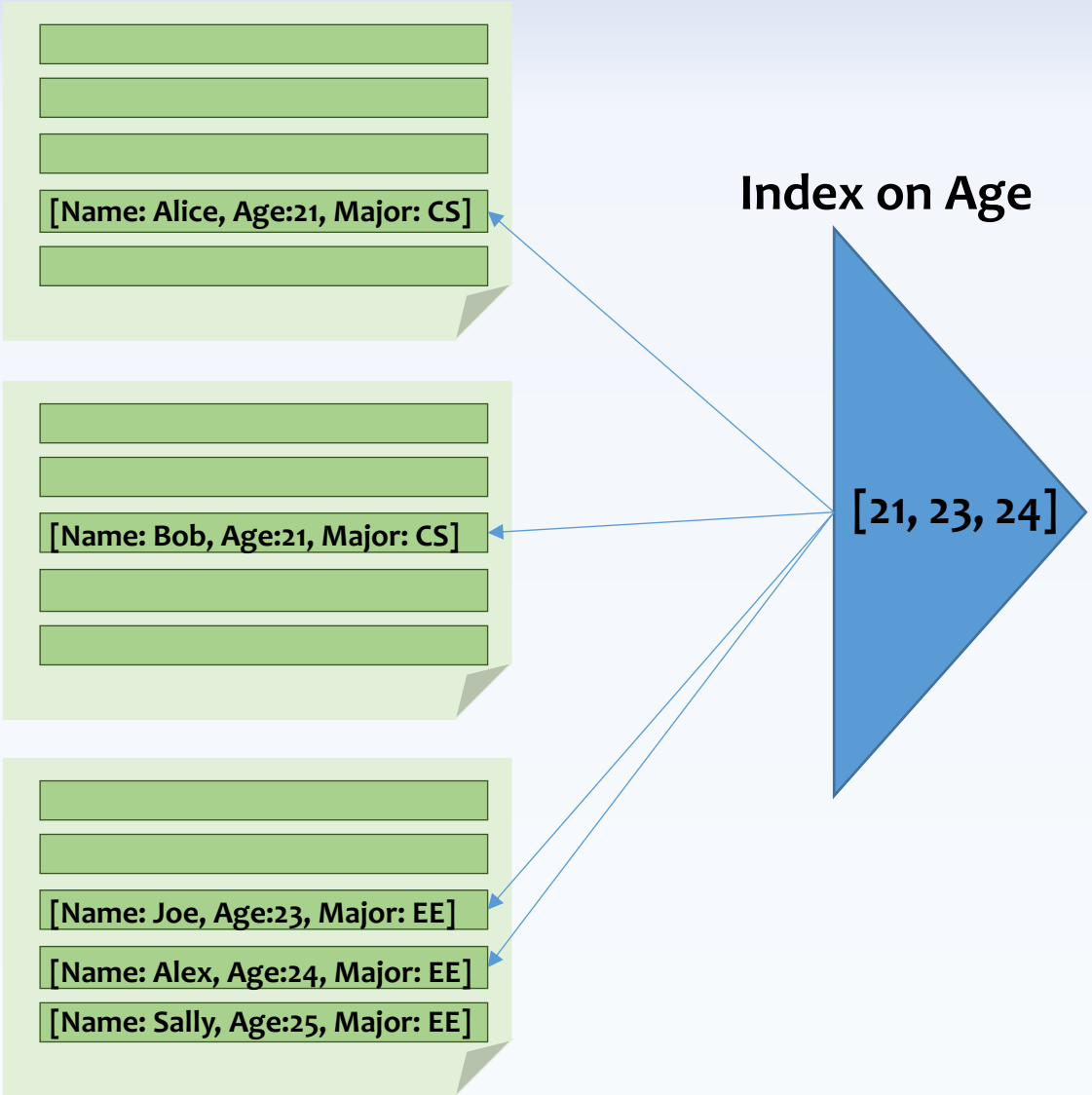
A column-based layout for two student records. It consists of three vertical green columns. The first column is labeled "[Name]", the second "[Age]", and the third "[Major]". The first record (Alice) has values "[Alice]", "[21]", and "[CS]" in the three columns respectively. The second record (Bob) has values "[Bob]", "[21]", and "[CS]" in the three columns respectively.



A column-based layout for three student records stacked vertically. It consists of three vertical green columns. The first column contains names: "[Joe]", "[Alex]", and "[Sally]". The second column contains ages: "[23]", "[24]", and "[25]". The third column contains majors: "[EE]", "[CS]", and "[EE]".

Column-based Layout

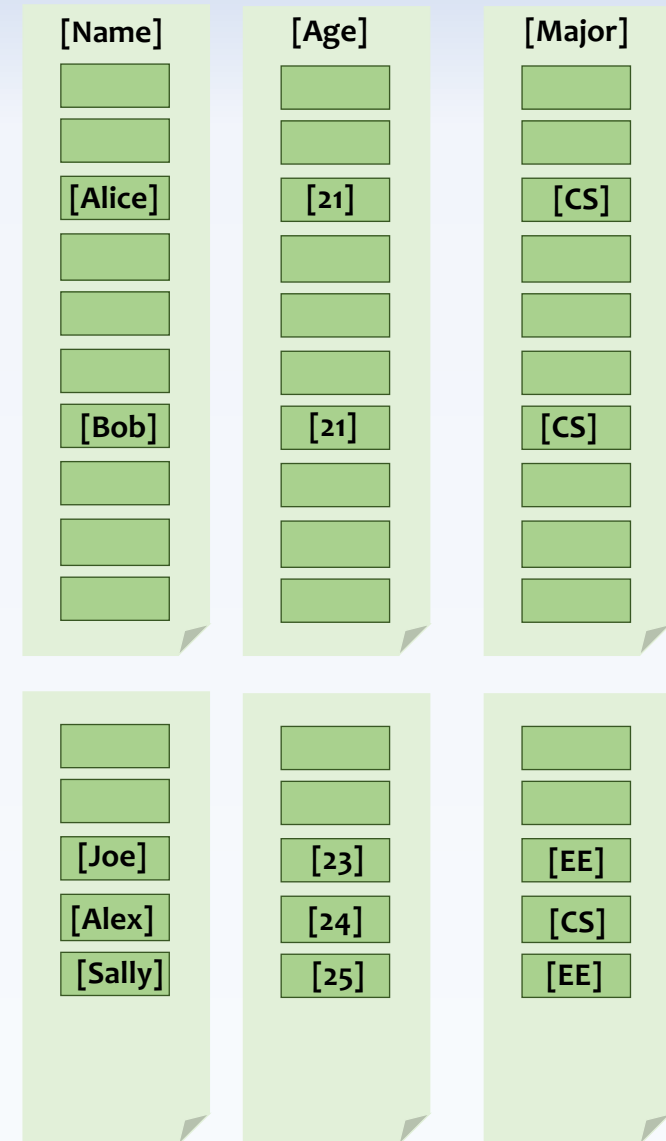
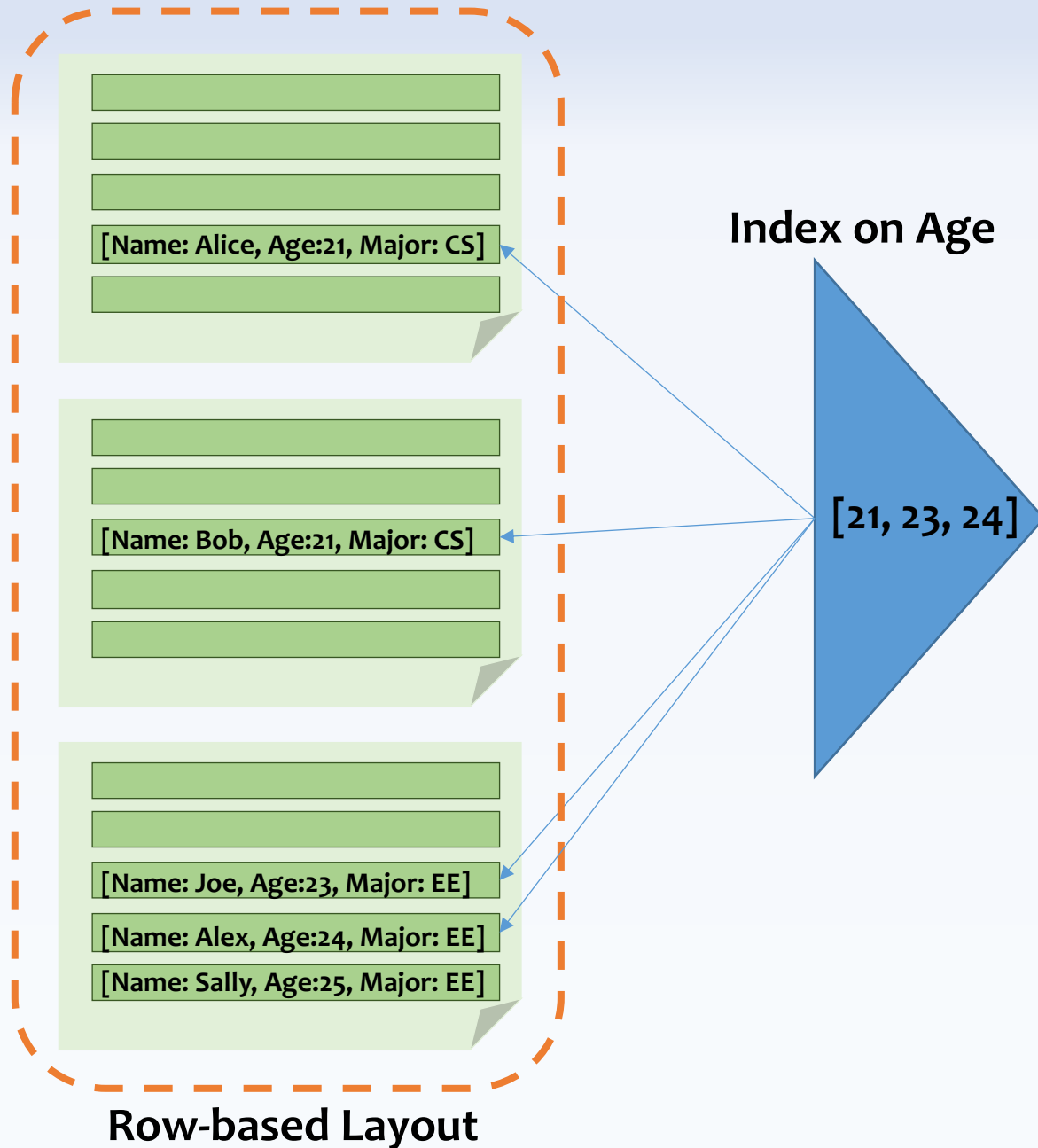
# Searching for all students between the age of 21 to 24 (returns many students)



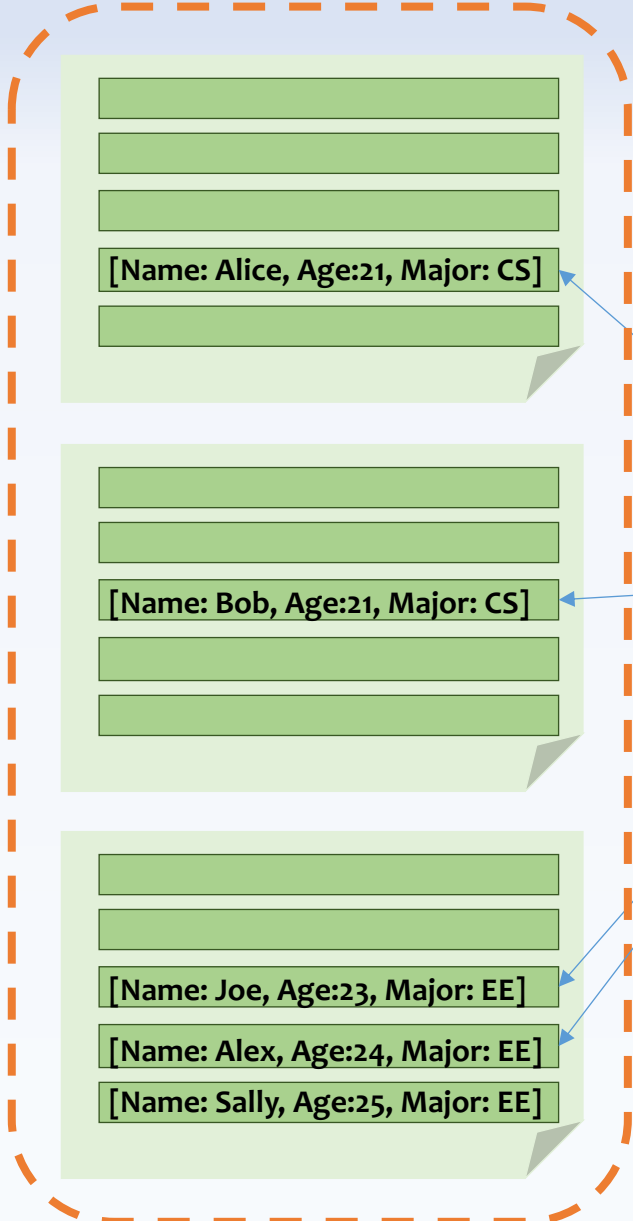
Row-based Layout

Column-based Layout

# Searching for all students between the age of 21 to 24 (returns many students)

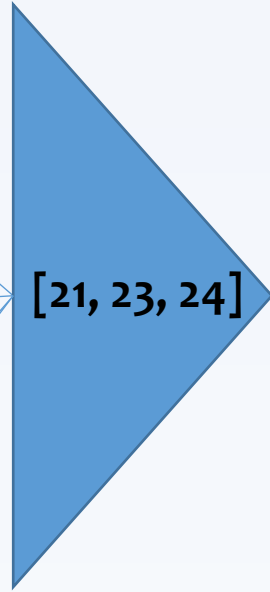


# Searching for all students between the age of 21 to 24 (returns many students)

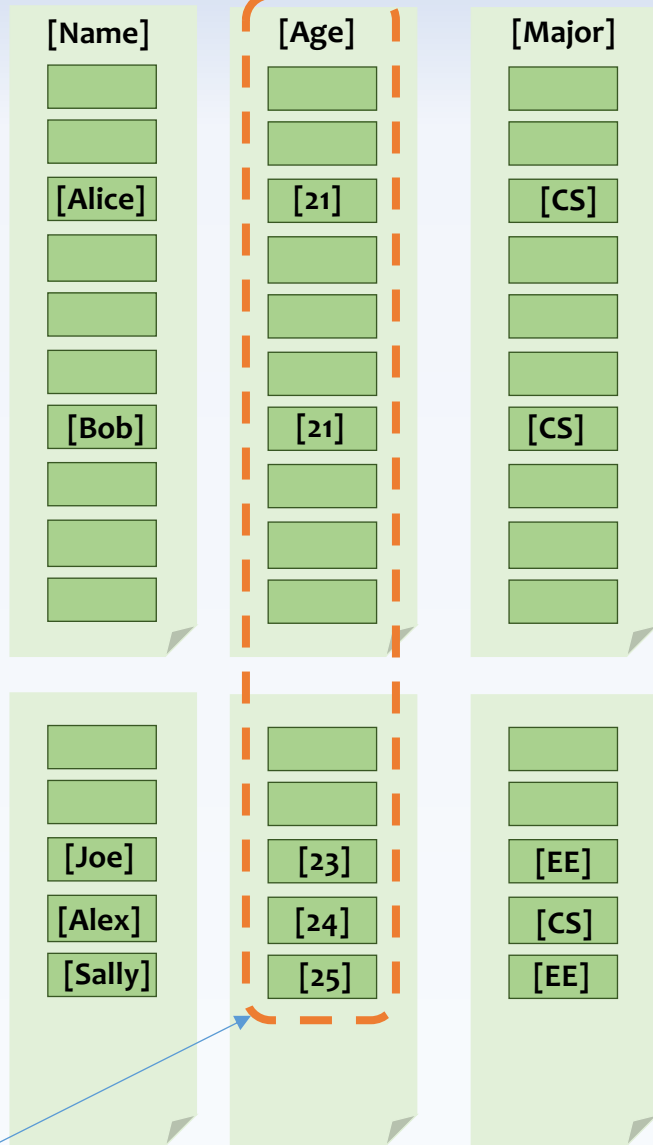


Row-based Layout

Index on Age

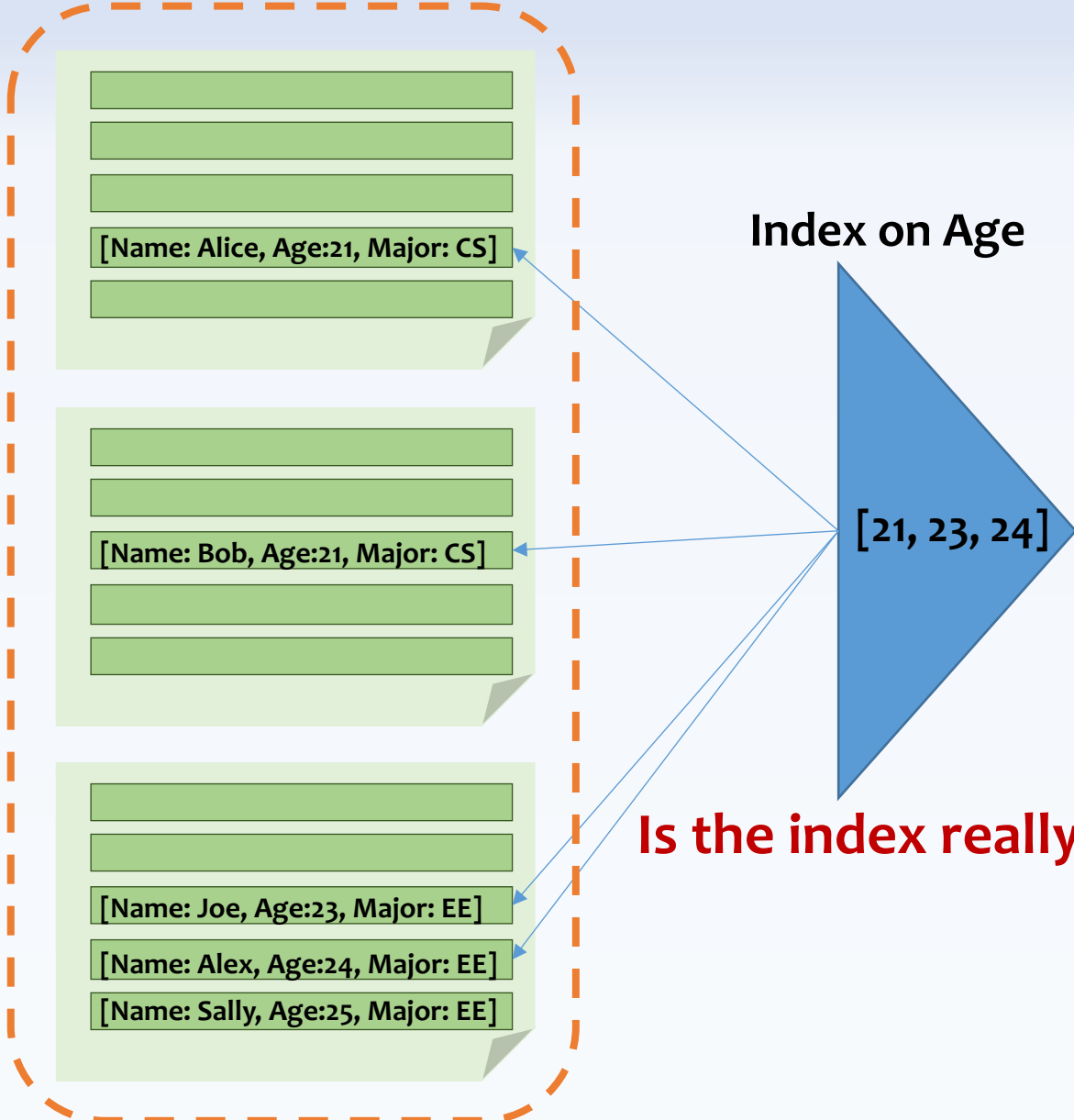


Alternatively read only the Age column to find the relevant values

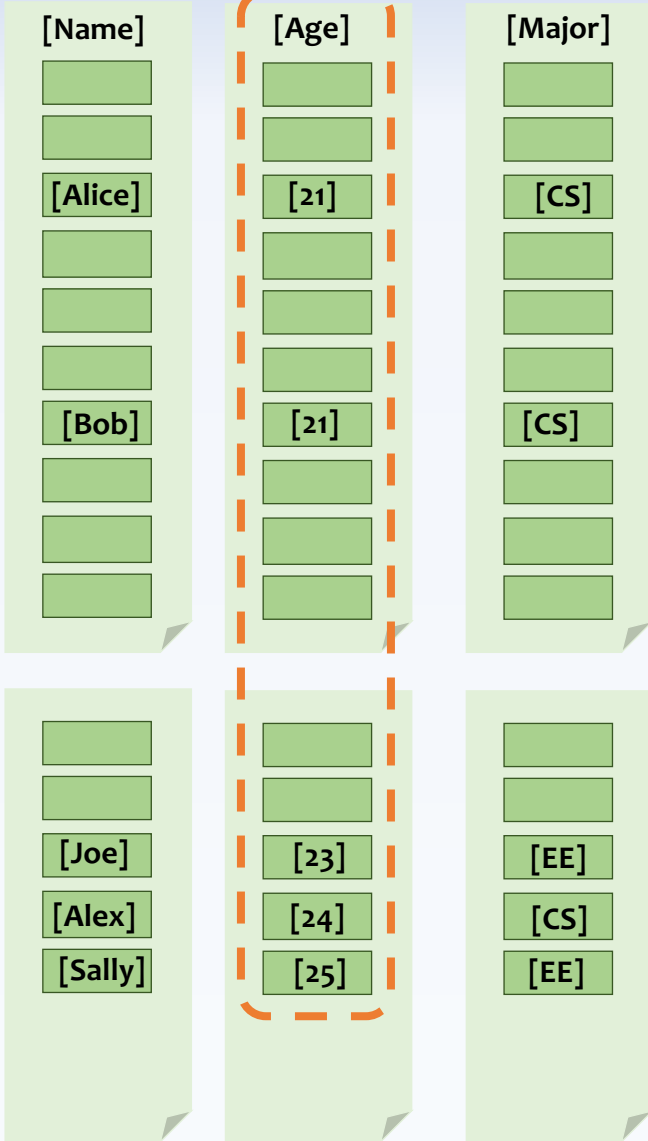


Column-based Layout

# Searching for all students between the age of 21 to 24 (returns many students)



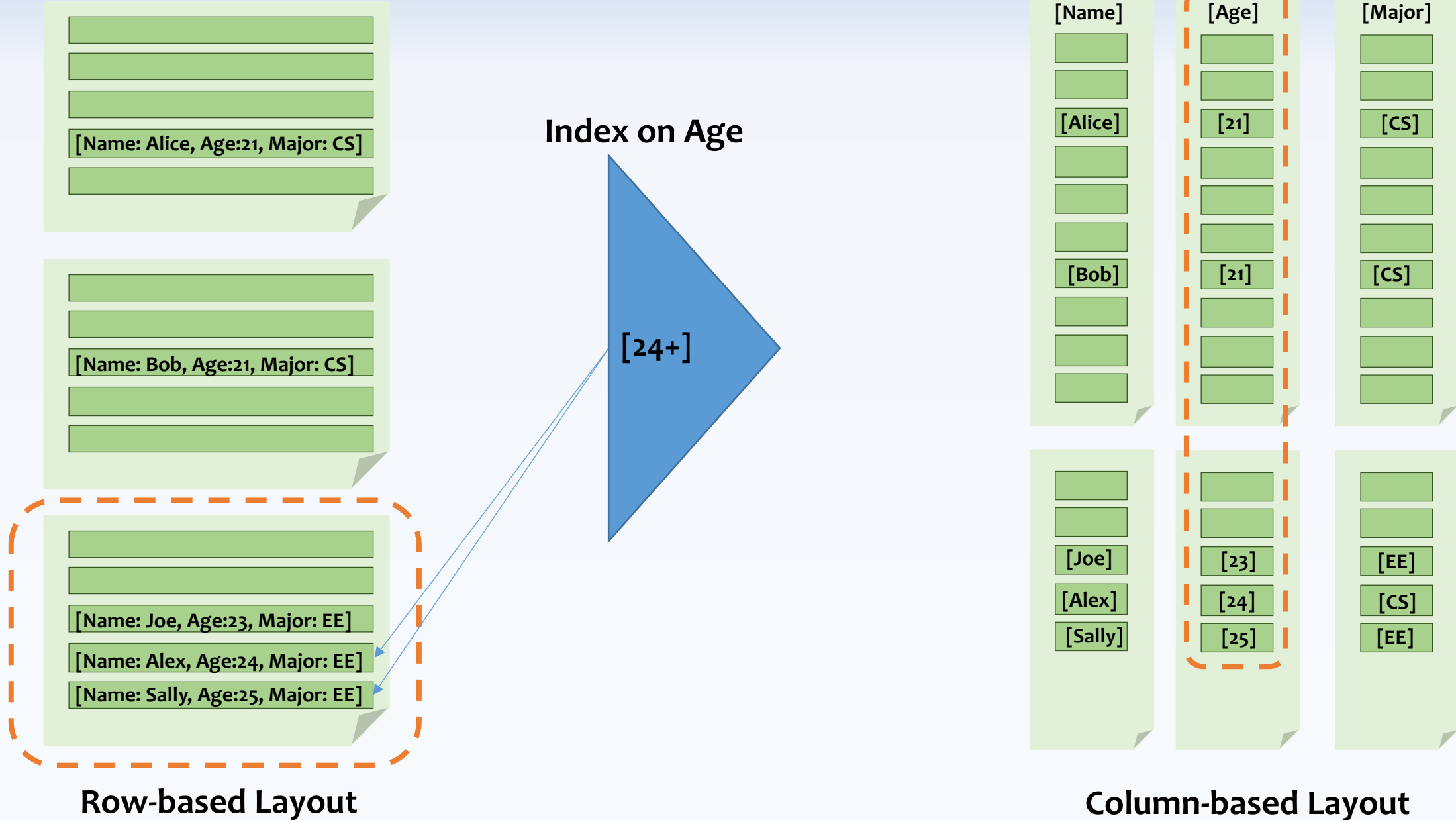
Row-based Layout



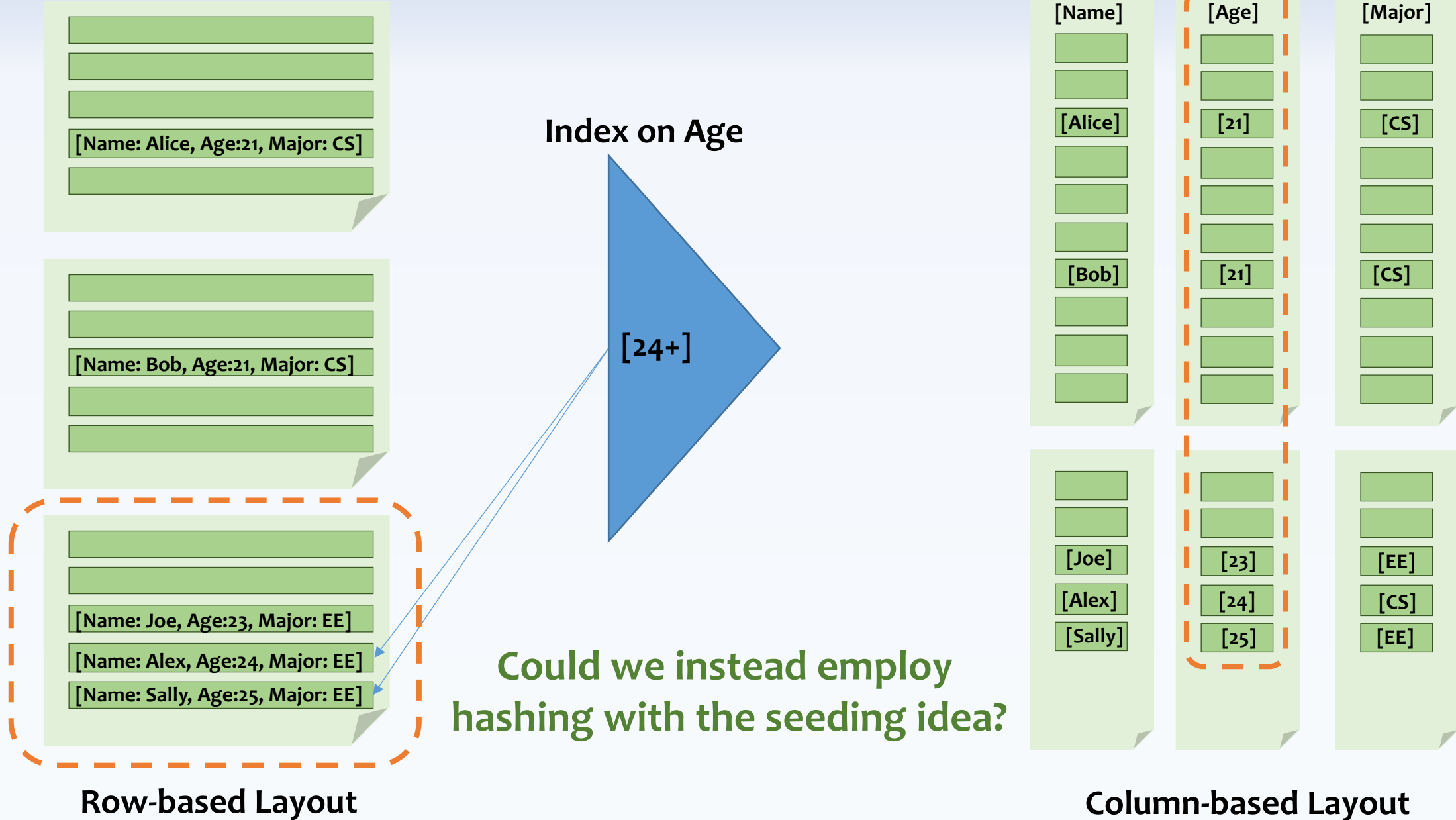
Column-based Layout

Is the index really useful here?

# Searching for all students over the age of 24 (returns only few students)



# Searching for all students over the age of 24 (returns only few students)



**Thank You  
Questions?**