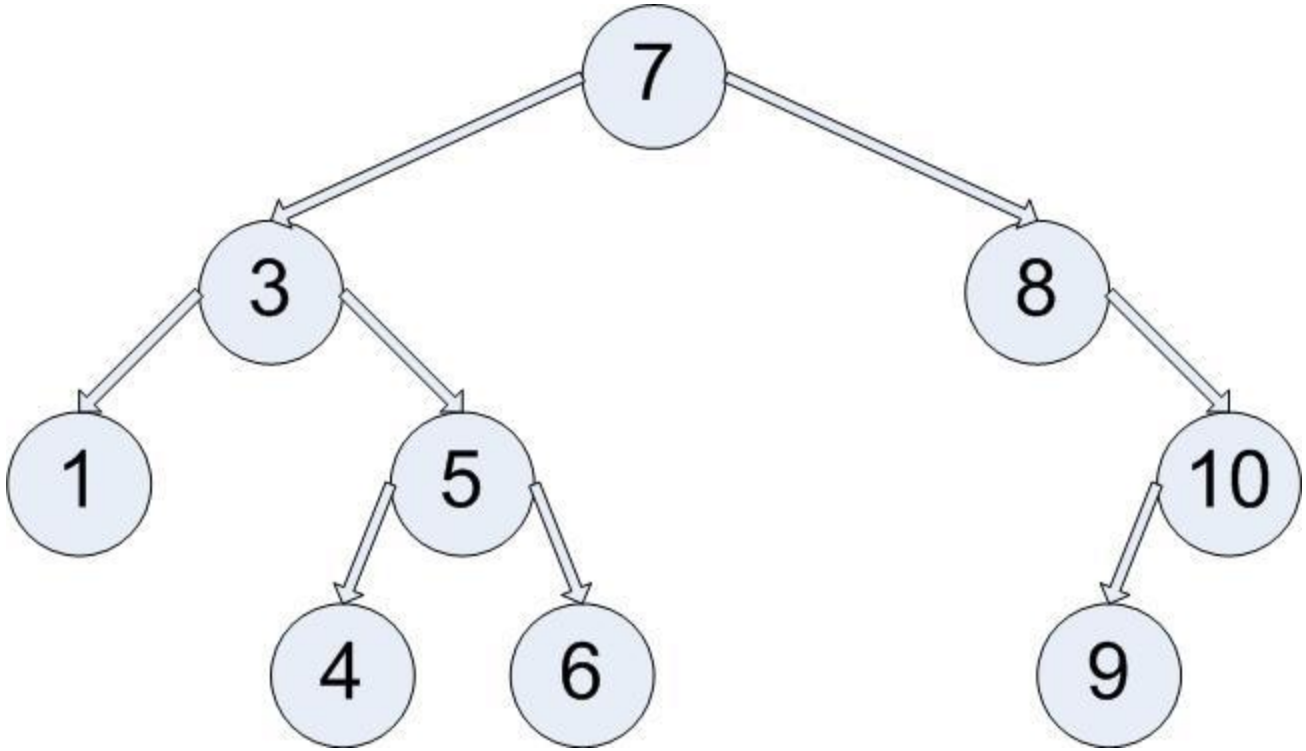
Skip Lists

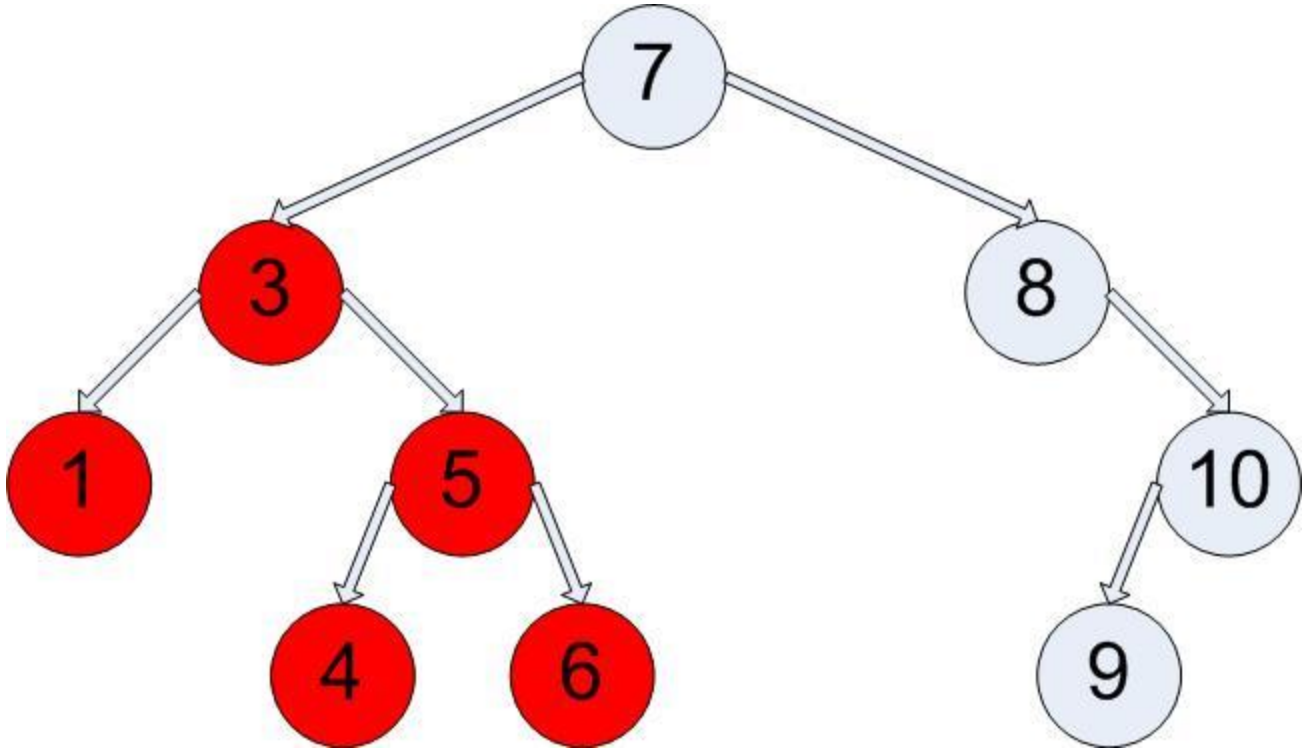
Outline

- **Trees**
- Skip Lists
  - Definition
  - Insertion
  - Search
  - Deletion
- Conclusion

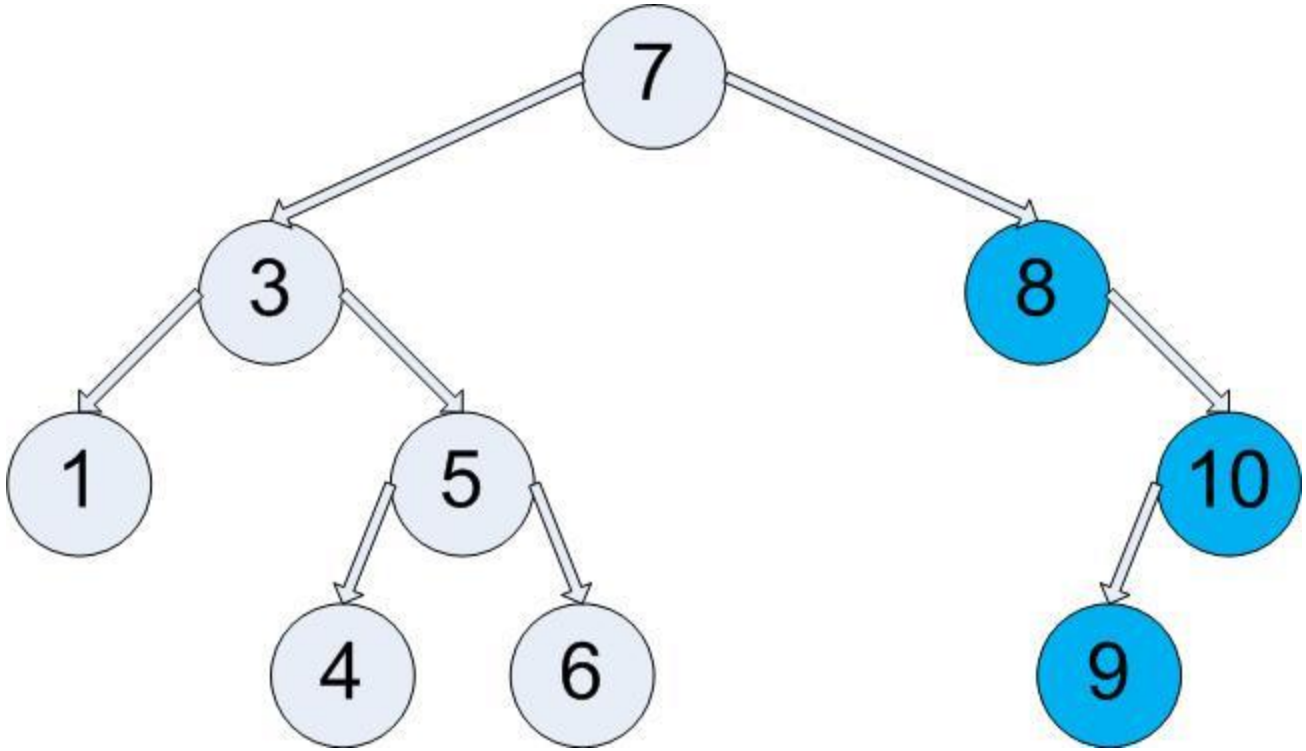
Binary Search Tree



Binary Search Tree



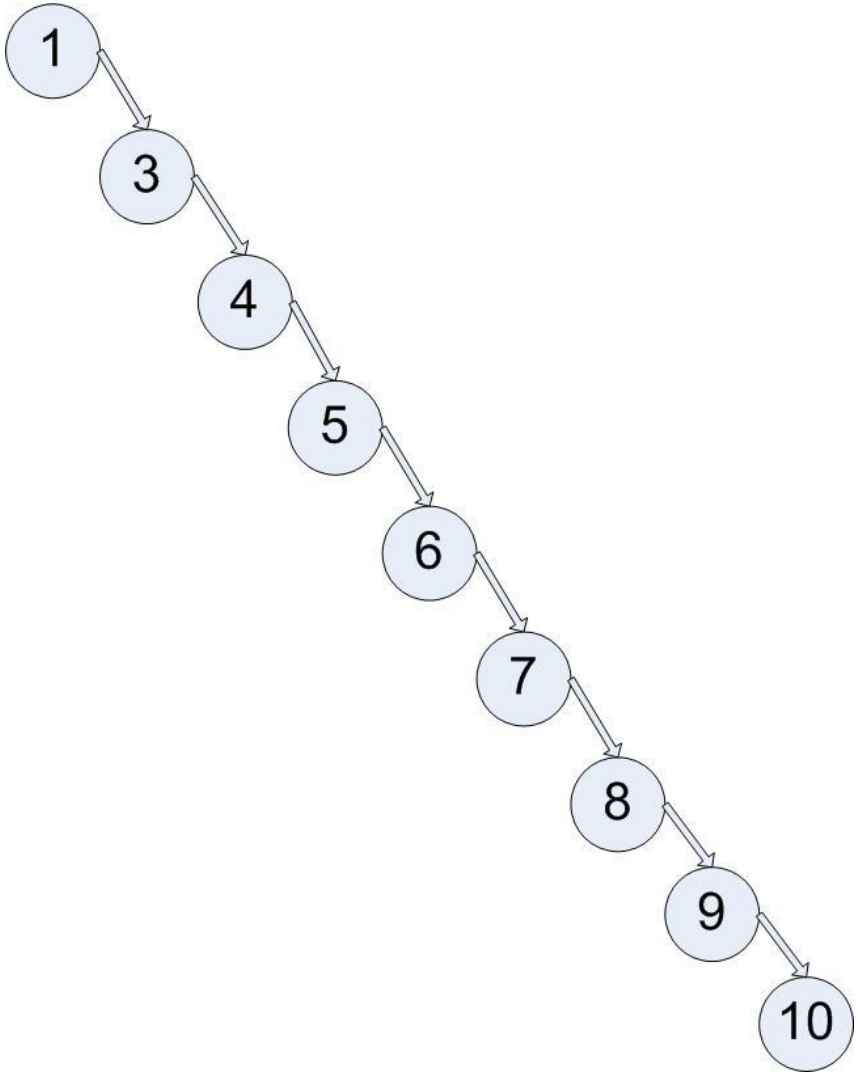
Binary Search Tree



Binary Search Tree Runtimes

- Runs in O(h) time:
  - Insert
  - Delete
  - Search
- h is the height of the tree.
- If tree is balanced and almost complete,
  h = log(n) where n is the total number of nodes.
- What is h if we insert in this order:
  - 1, 3, 4, 5, 6, 7, 8, 9, 10

Binary Search Tree



Self Balancing Trees

- AVL Trees
  - Height limited to 1.44*log(*n*)
  - O(log *n*) time for basic operations: insert, search, delete
  - Complicated rules for tree rotation: Right-right, Right-left…
- Red-Black Trees
  - Height limited to 2*log(*n*)
  - O(log *n*) time for basic operations: insert, delete, search
  - Properties to keep track of:
    - Root is black
    - All leaves are black
    - Both children of a red node are black
    - All path from a given node to its leaf nodes contains the same number of black nodes
  - Rotations

Outline

- Trees
- **Skip Lists**
  - **Definition**
  - Insertion
  - Search
  - Deletion
- Conclusion

Skip List Motivation

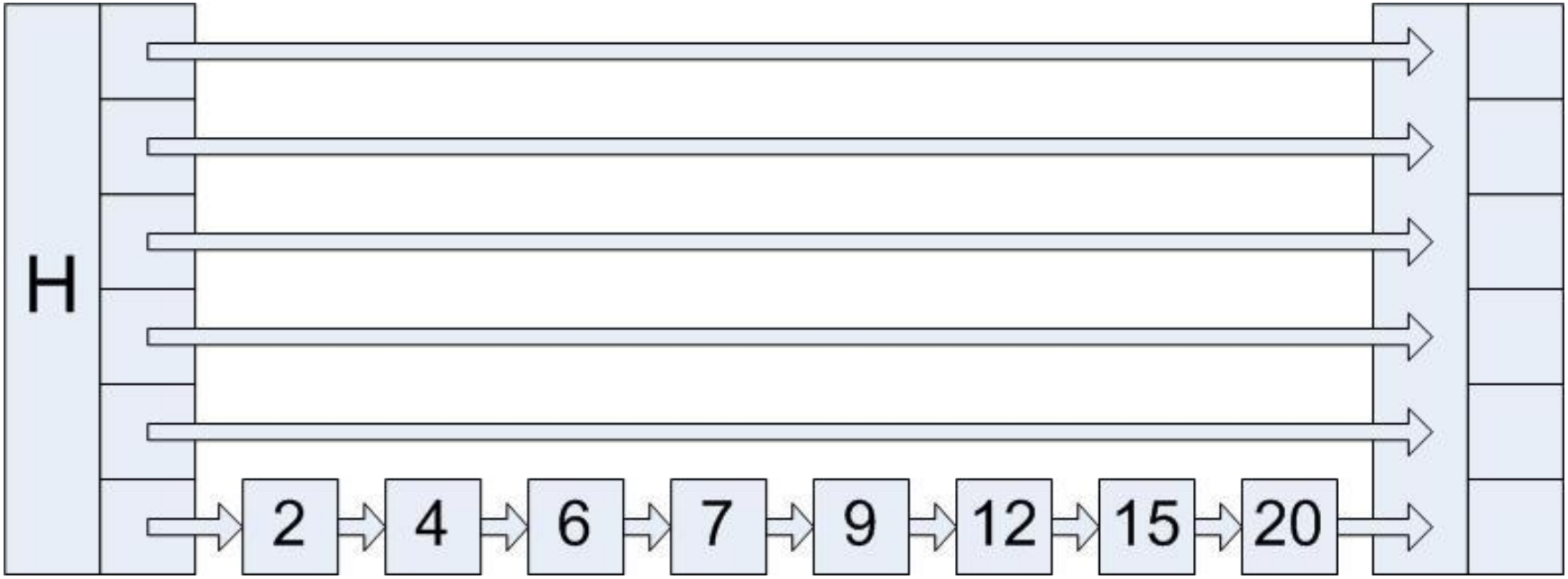
- Implementation is generally easier than a self-balancing tree.
- Skip lists can be used in place of trees for most applications.
- Skip lists have a balanced behavior thanks to randomization.
- As the creator William Pugh says: "Skip lists are about as fast as highly optimized balanced tree algorithms and are substantially faster than casually implemented balanced tree algorithms."

Probabilistic Algorithms

- Monte Carlo Algorithms
  - Guaranteed to be fast
  - Probably correct
- Las Vegas Algorithms
  - Guaranteed to be correct
  - Probably fast
- What are skip lists?

Skip List

- Lowest level of a skip list is a linked list.

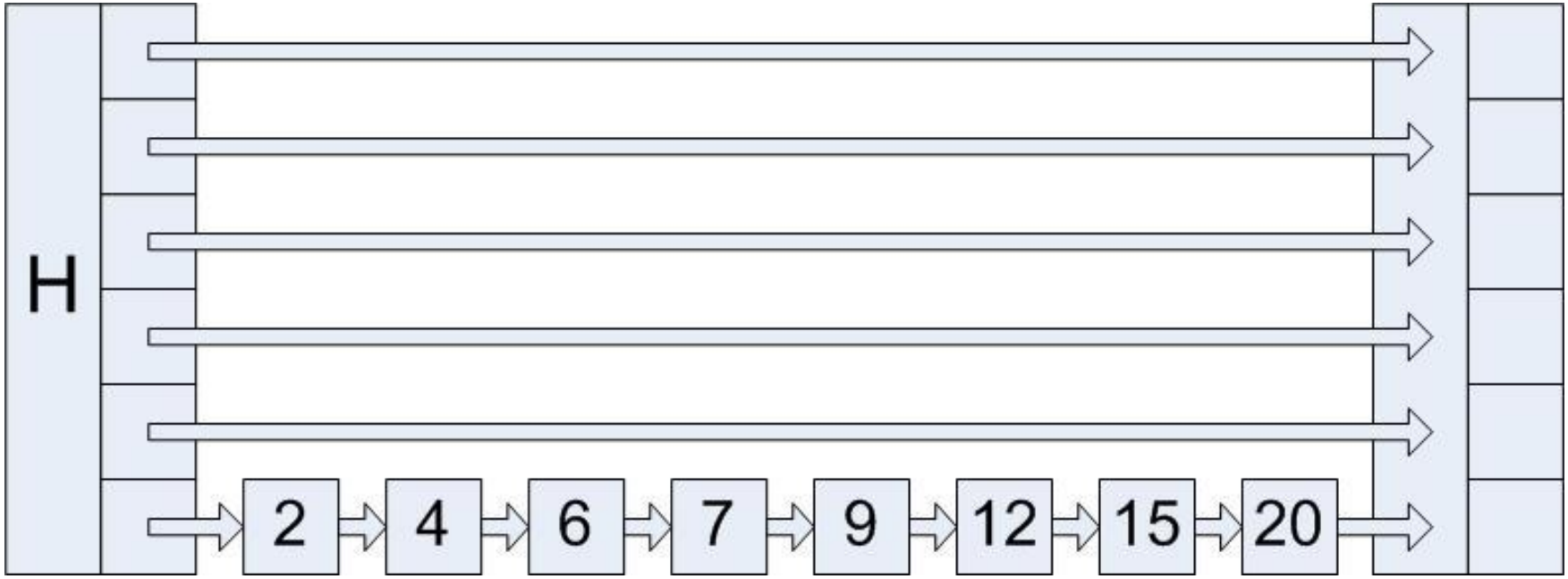


Skip List

- Ideally:
  - Level 0 contains all links to all nodes in the list.
  - Level 1 contains ½ of all of the elements in Level 0.
  - Level 2 contains ½ of all of the elements in Level 1.
  - …
- It is possible for all elements to be at Level 0 and have no elements at Level 1.

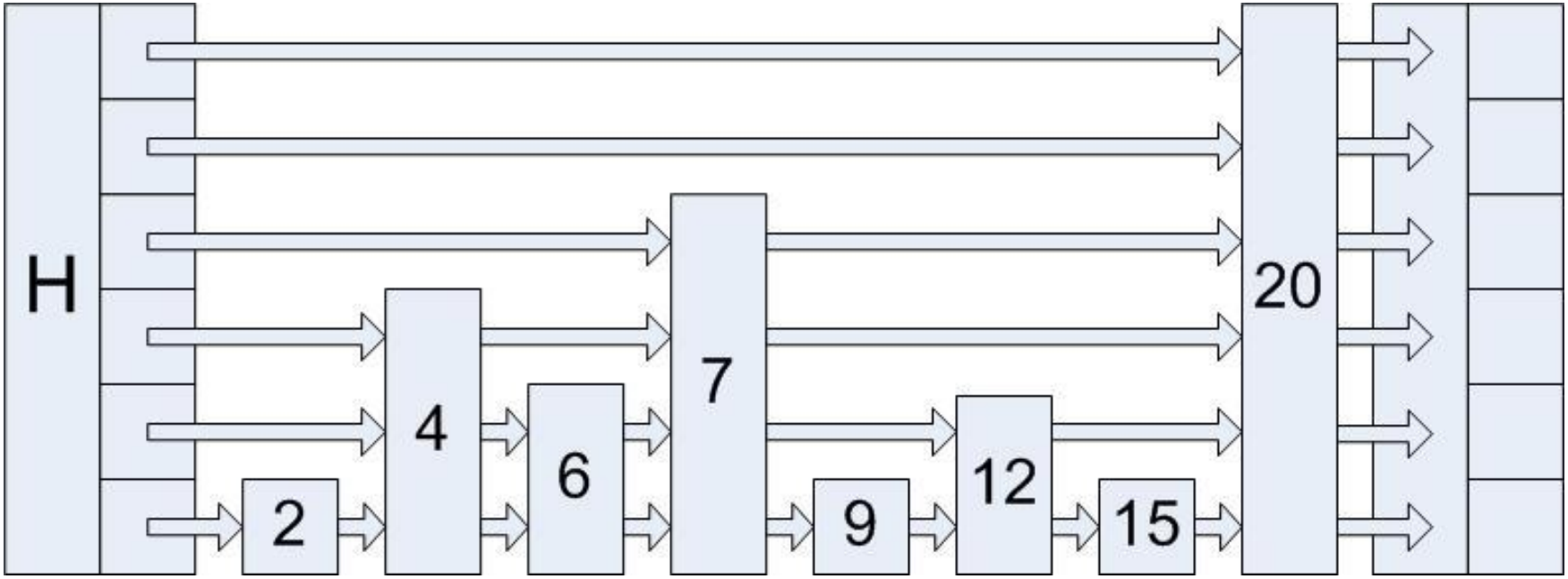
Skip List

- This is an unlucky skip list:



Skip List

- This is an normal skip list:



Outline

- Trees
- **Skip Lists**
  - Definition
  - **Insertion**
  - Search
  - Deletion
- Conclusion

Skip List

- Initial Skip List:

Example

- Insert 5, 3, 6, 4, 2, 1, 9, 7 into a skip list.
- Levels must be 0, 1 or 2.

Skip List Insertion

Insert *x* into skip list:

Run search to find the appropriate location to place *x*.

Find the greatest number less than *x*.

Find the smallest number greater than *x*.

level ← 0

while(coin flip is heads)

{

level ← level + 1

}

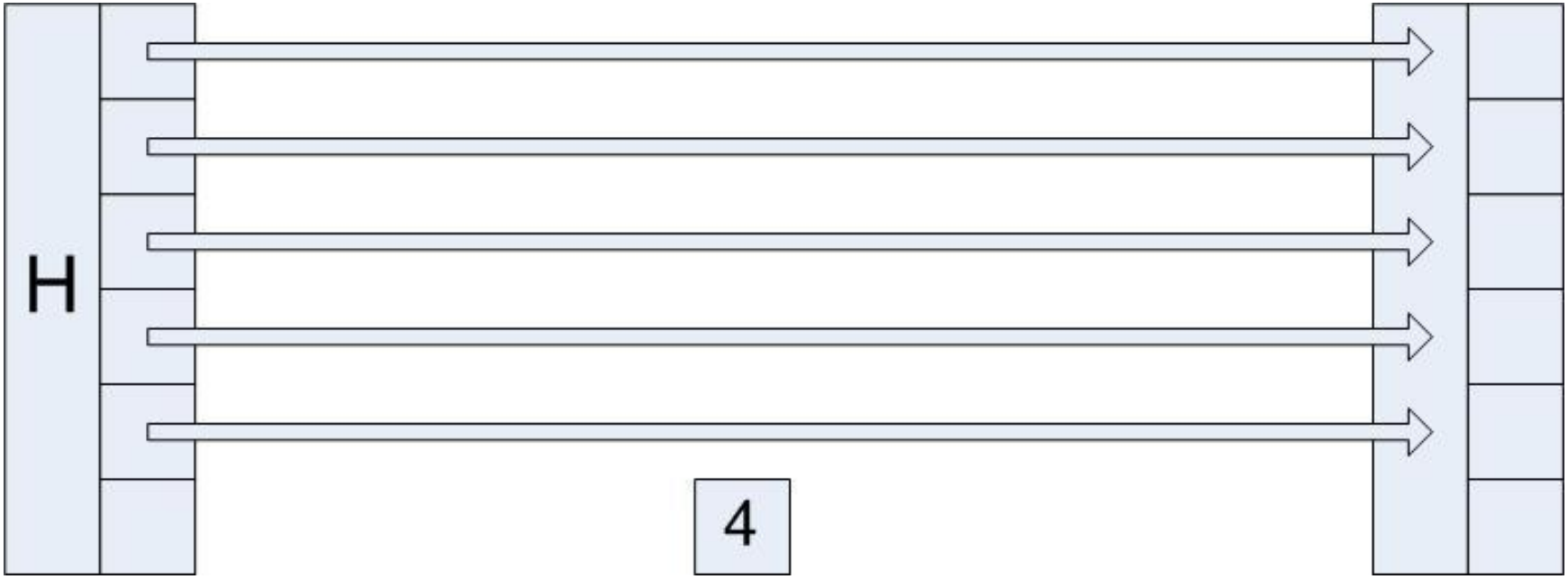
Place node *x* at level and reassign arrows as necessary.

Insertion

- Initial Skip List:

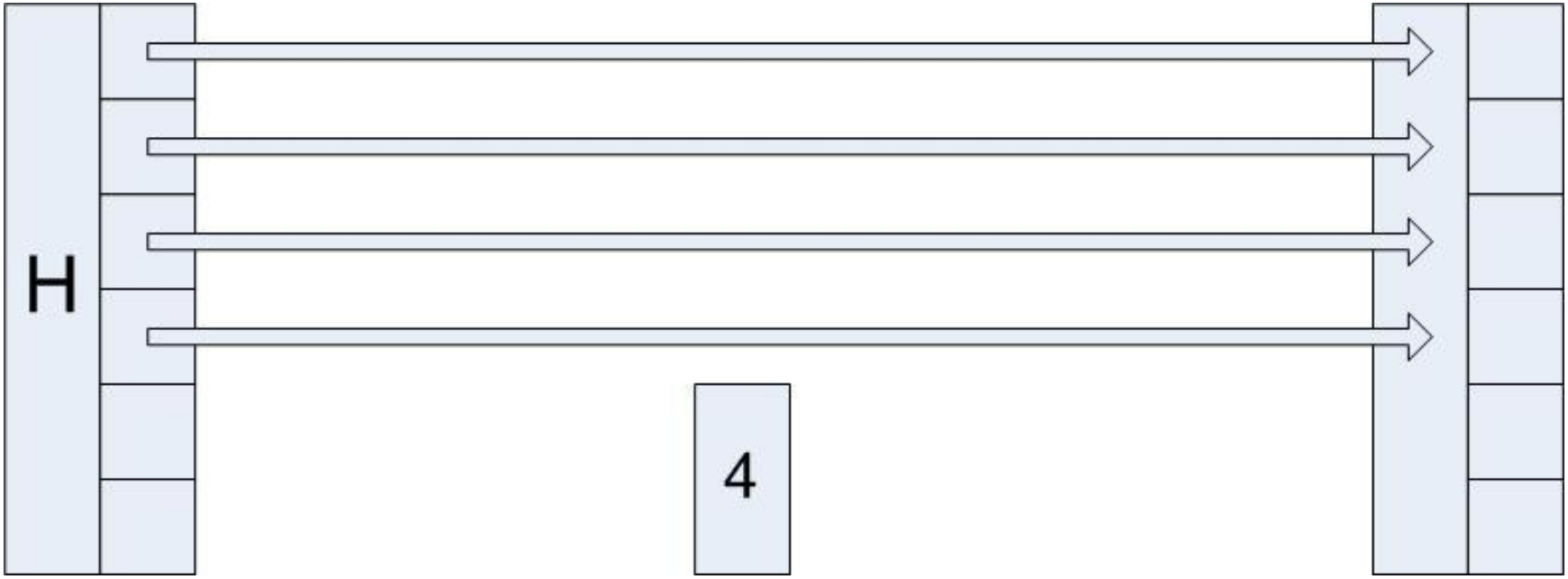
Insertion

- Insert 4:



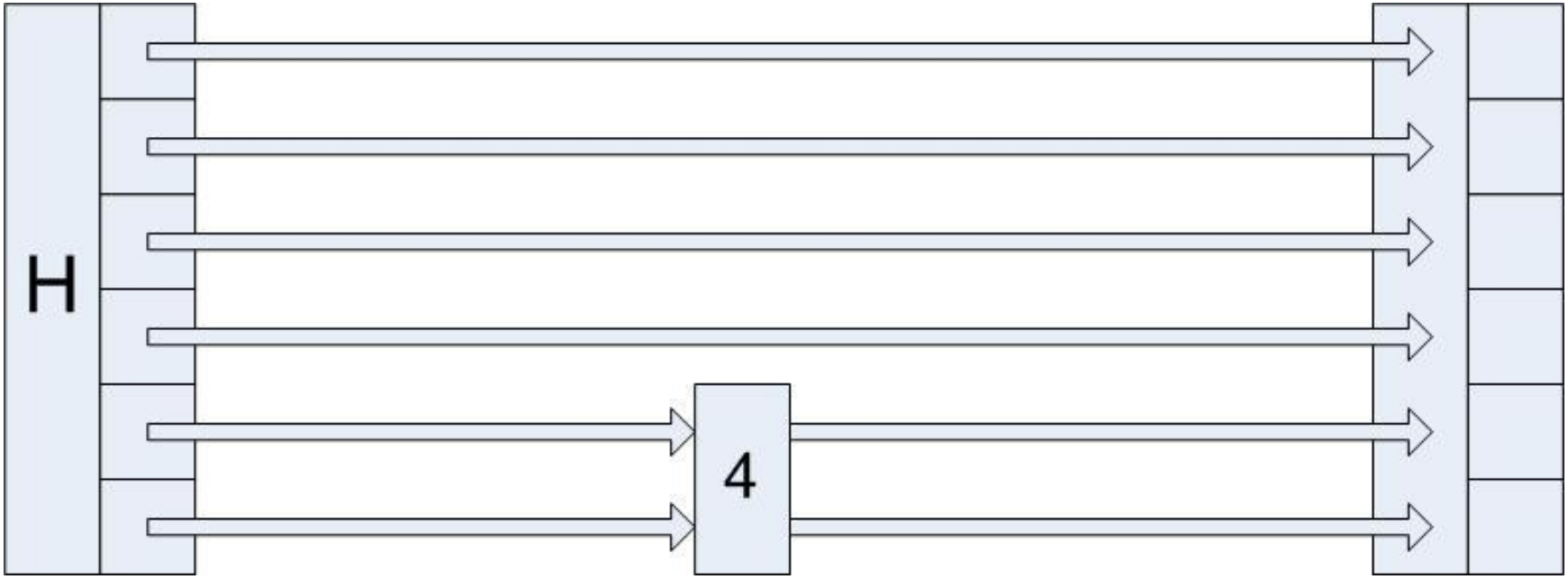
Insertion

- Insert 4:



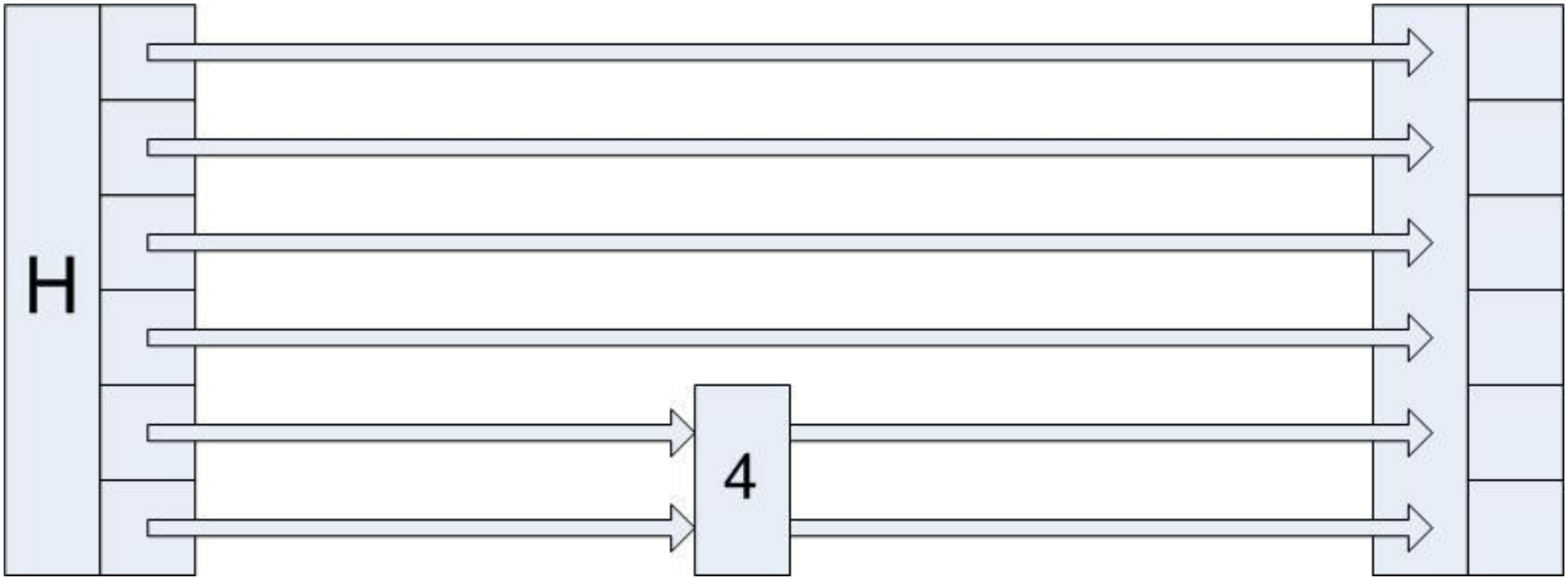
Insertion

- Insert 4:



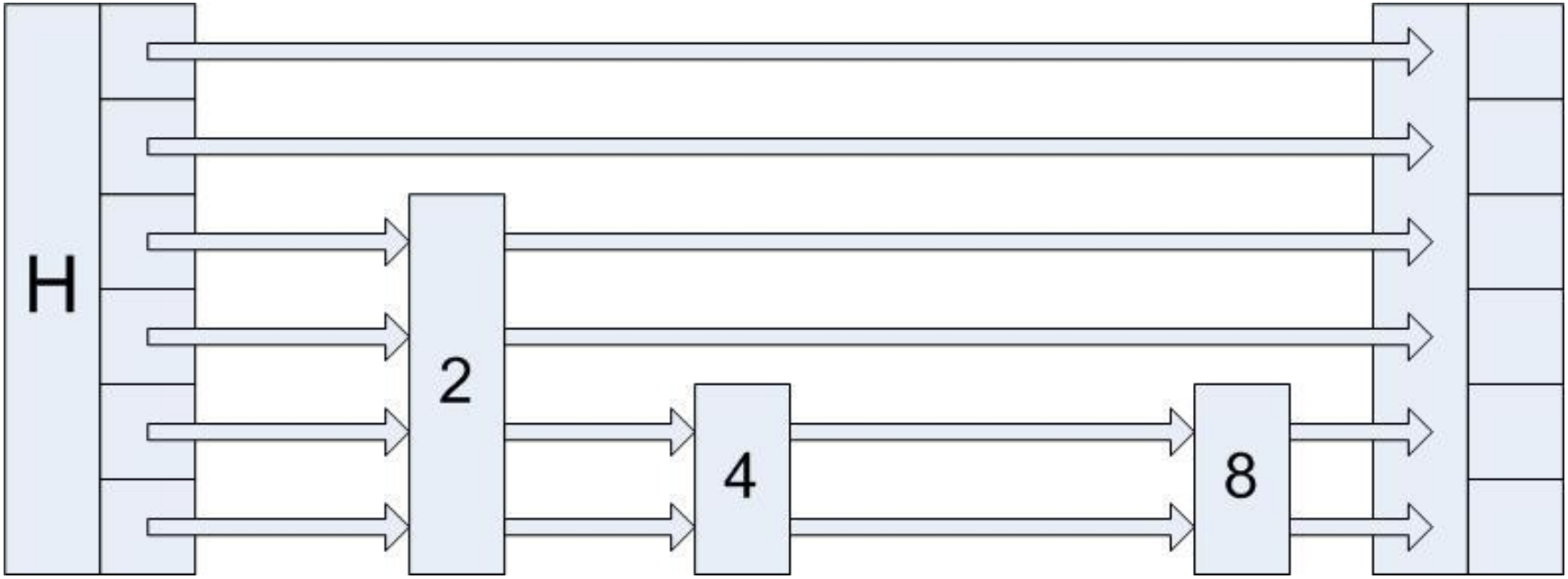
Quick Question

- Insert 8, has 2 levels.
- Insert 2, has 4 levels.



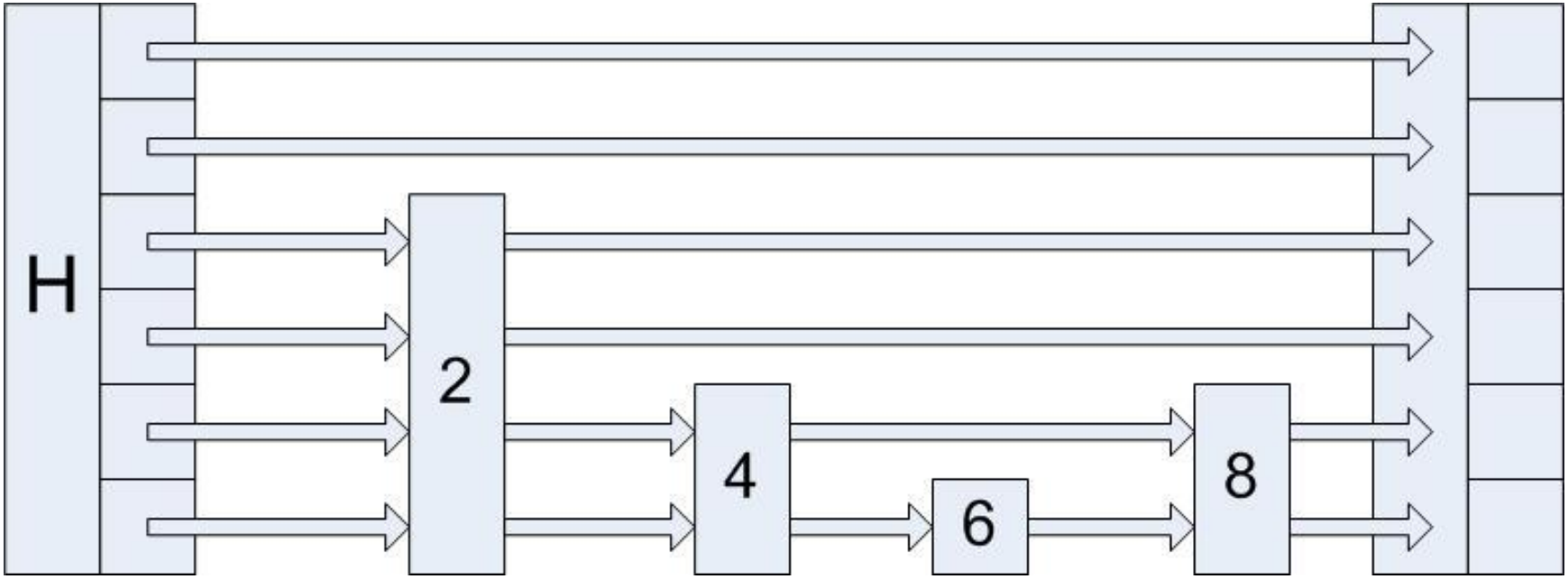
Insertion

- Insert 6 with 1 level:



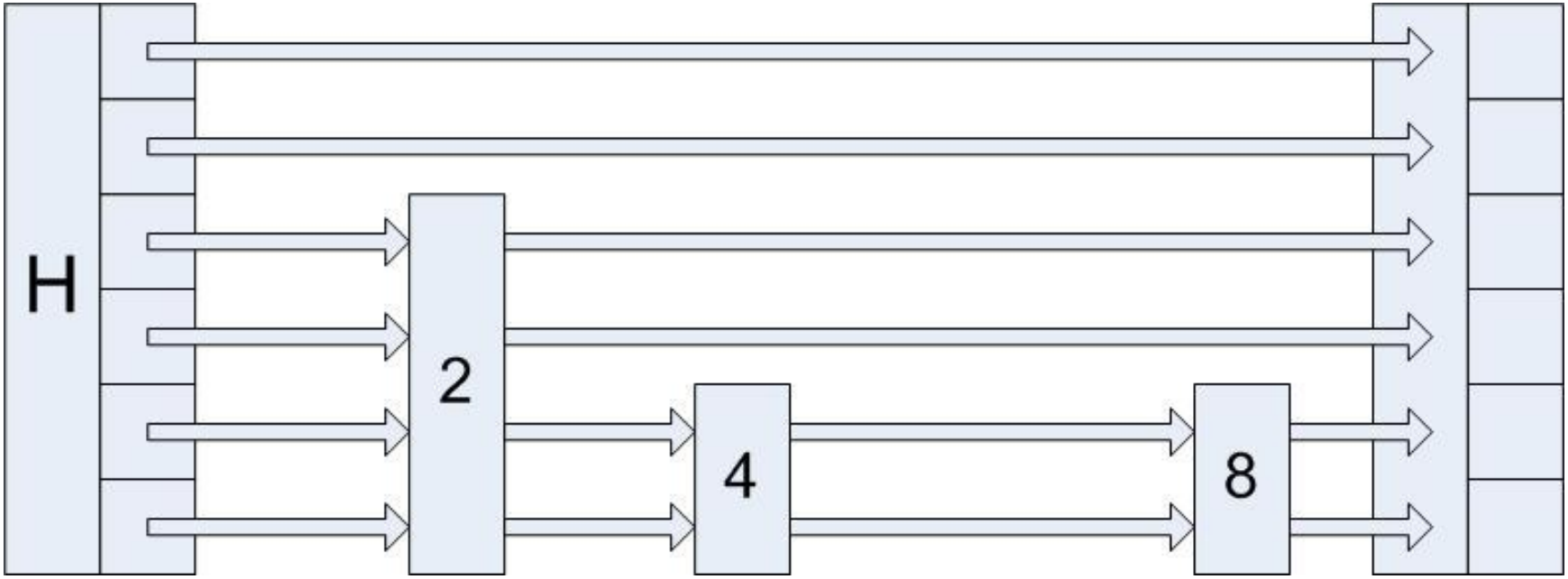
Insertion

- Insert 6 with 1 level:



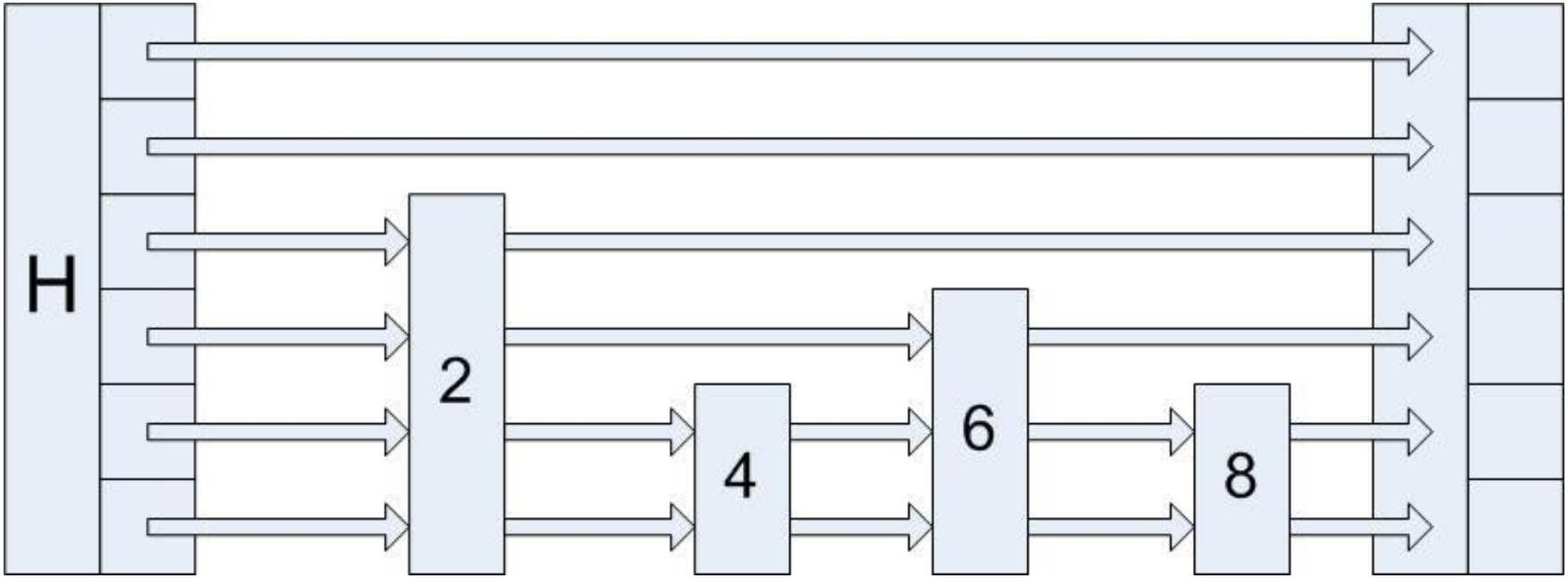
Insertion

- Insert 6 with level 3:



Insertion

- Insert 6 with level 3:

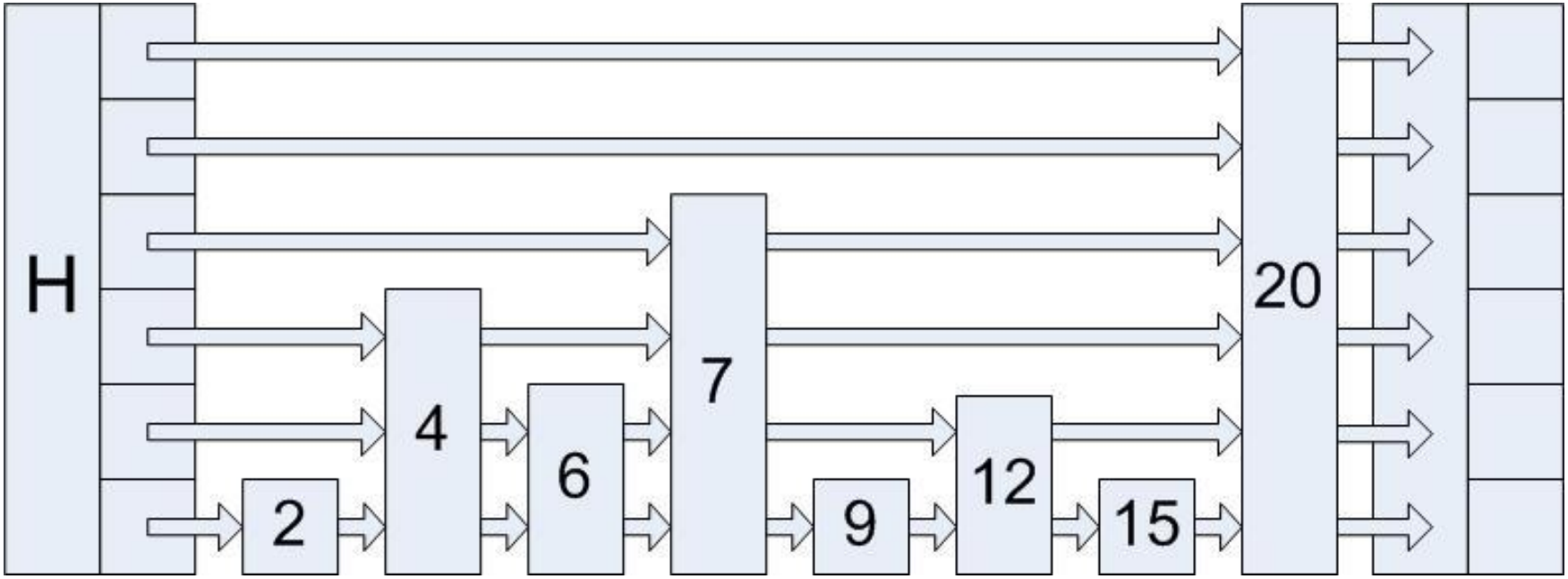


Outline

- Trees
- **Skip Lists**
  - Definition
  - Insertion
  - **Search**
  - Deletion
- Conclusion

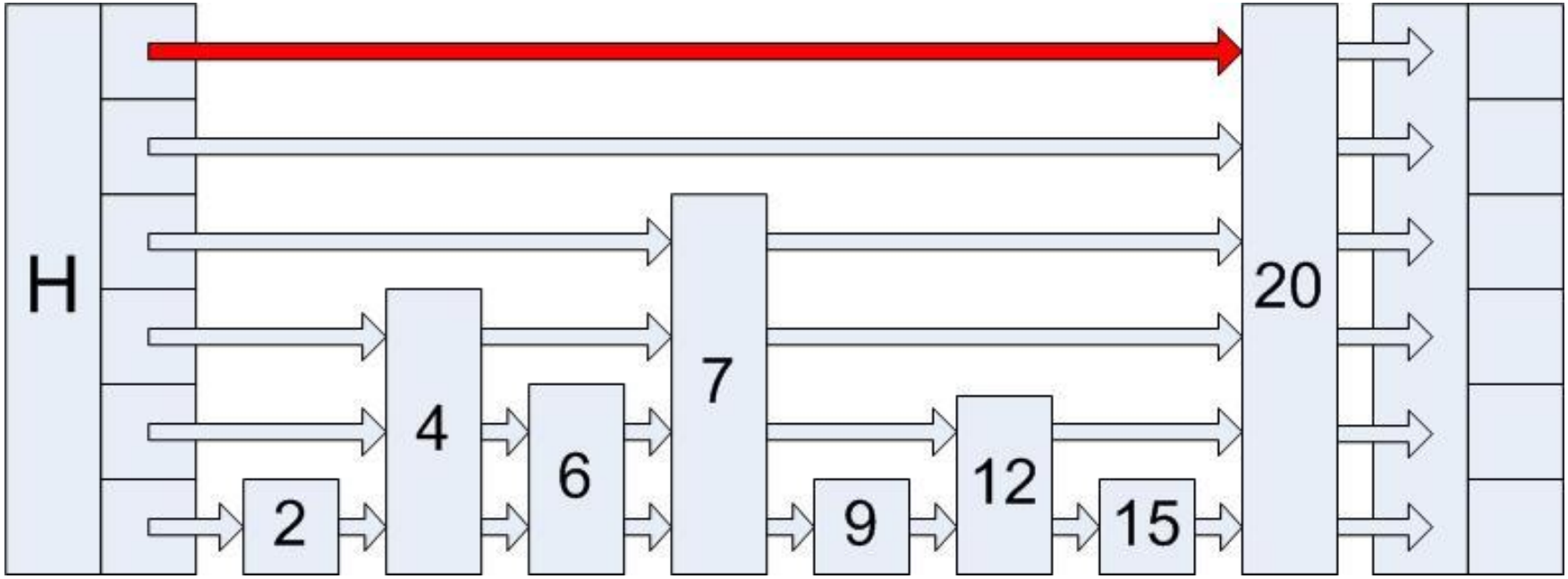
Skip List

- Search for the number 9:



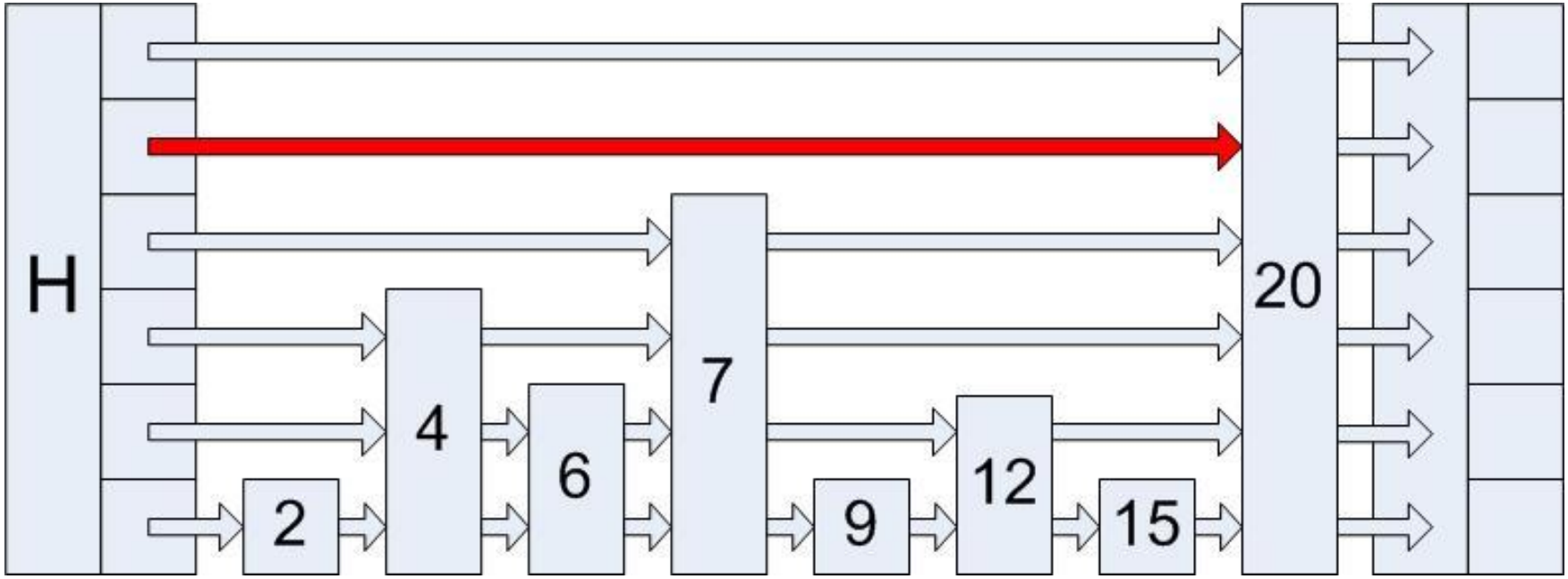
Skip List

- Search for the number 9:



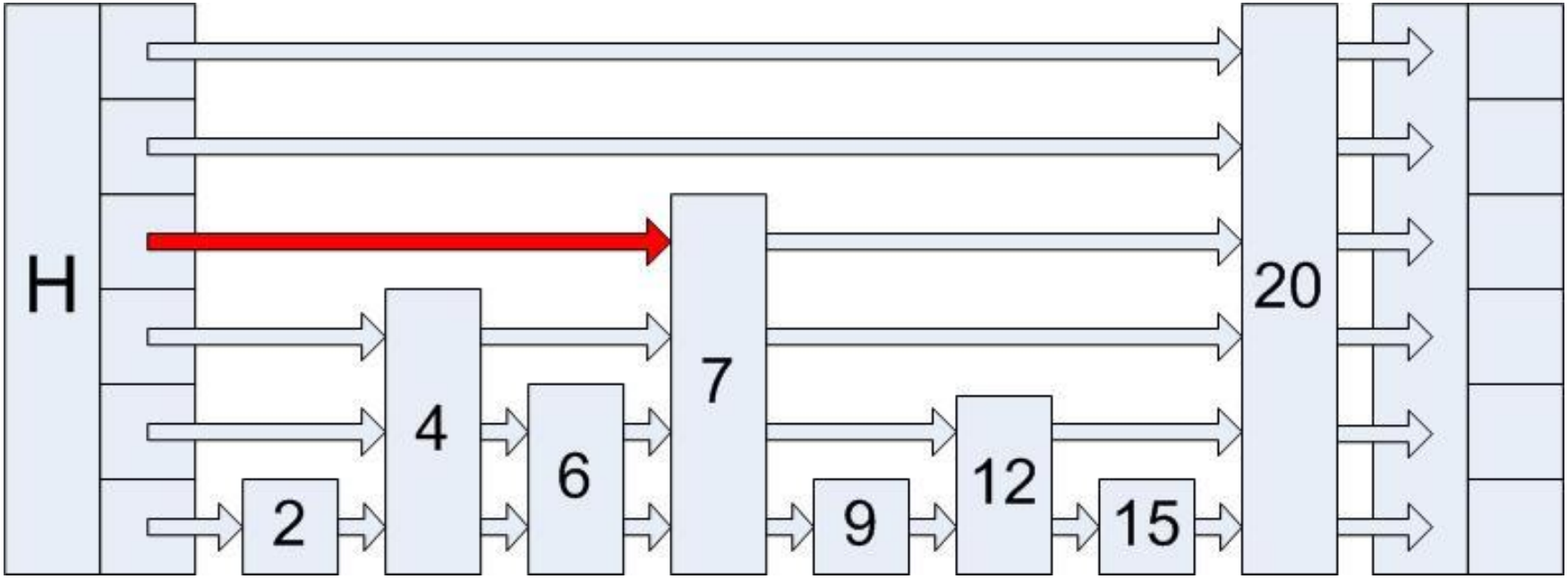
Skip List

- Search for the number 9:



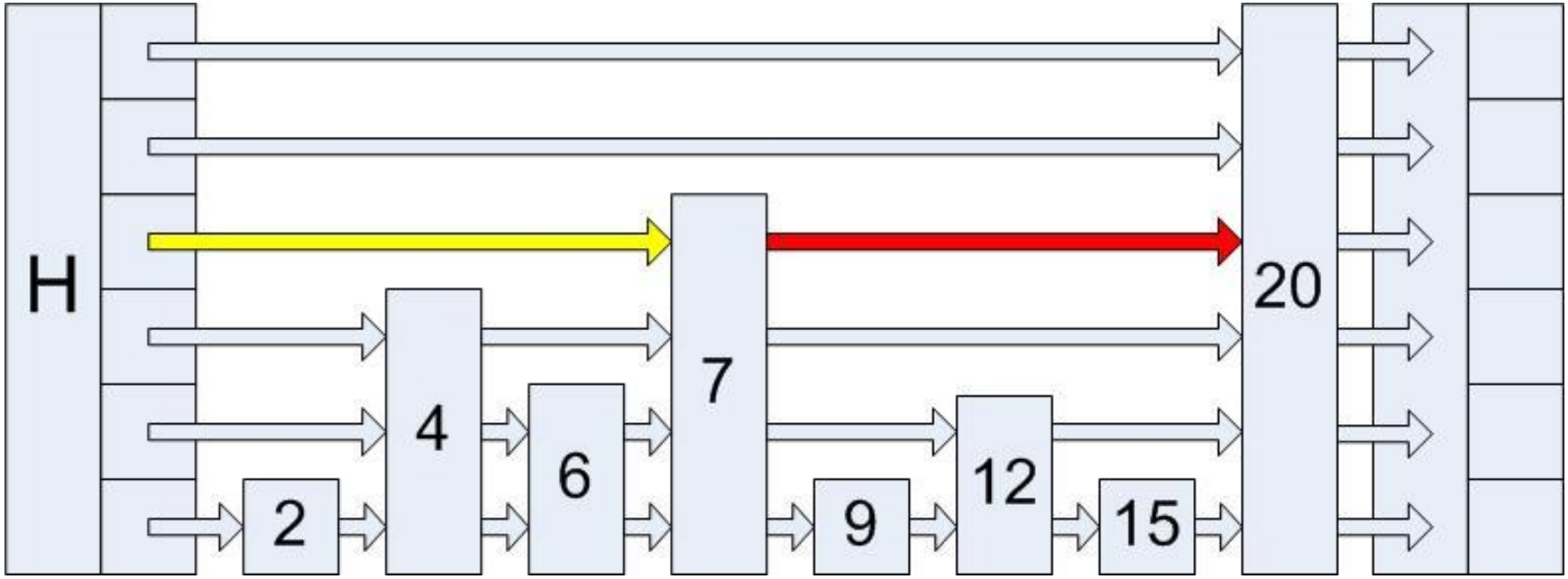
Skip List

- Search for the number 9:



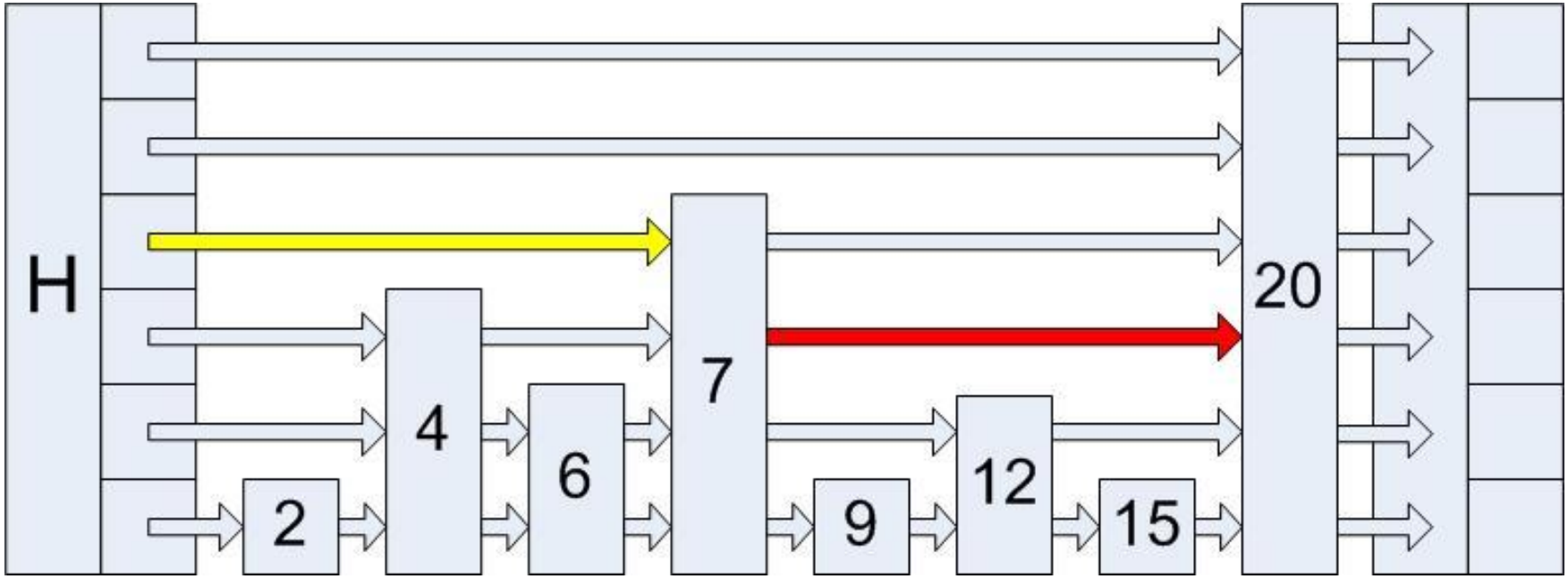
Skip List

- Search for the number 9:



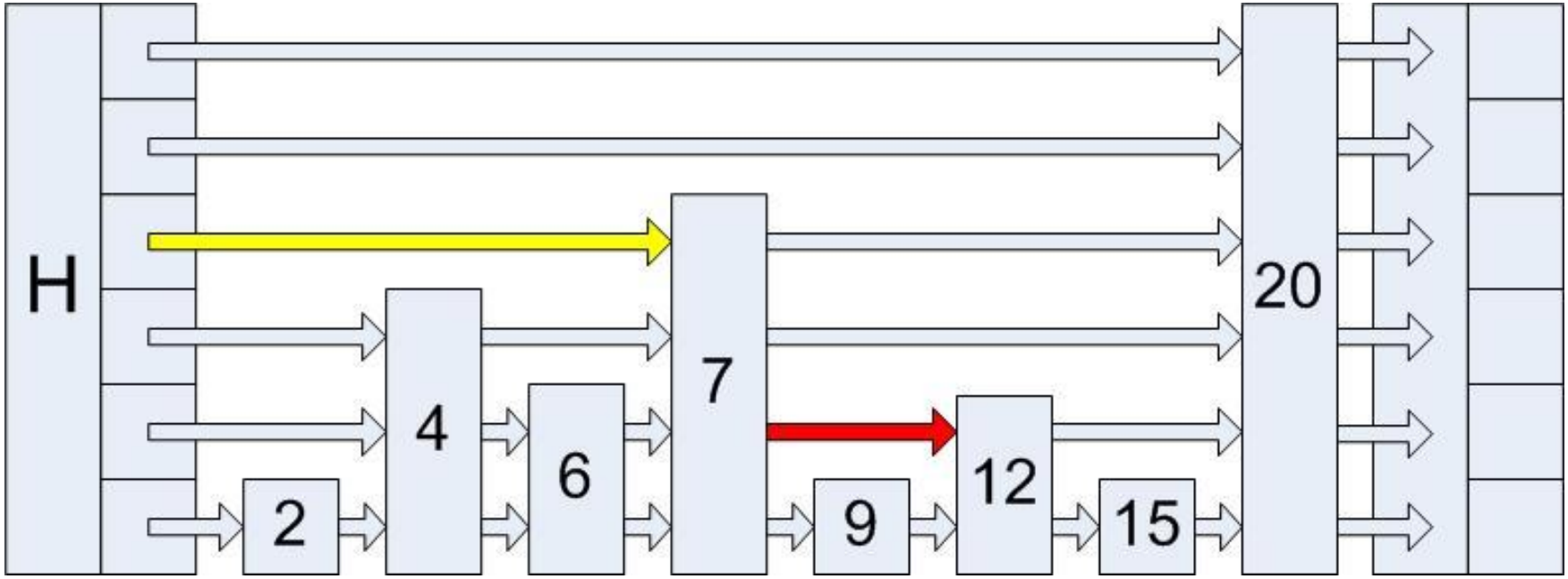
Skip List

- Search for the number 9:



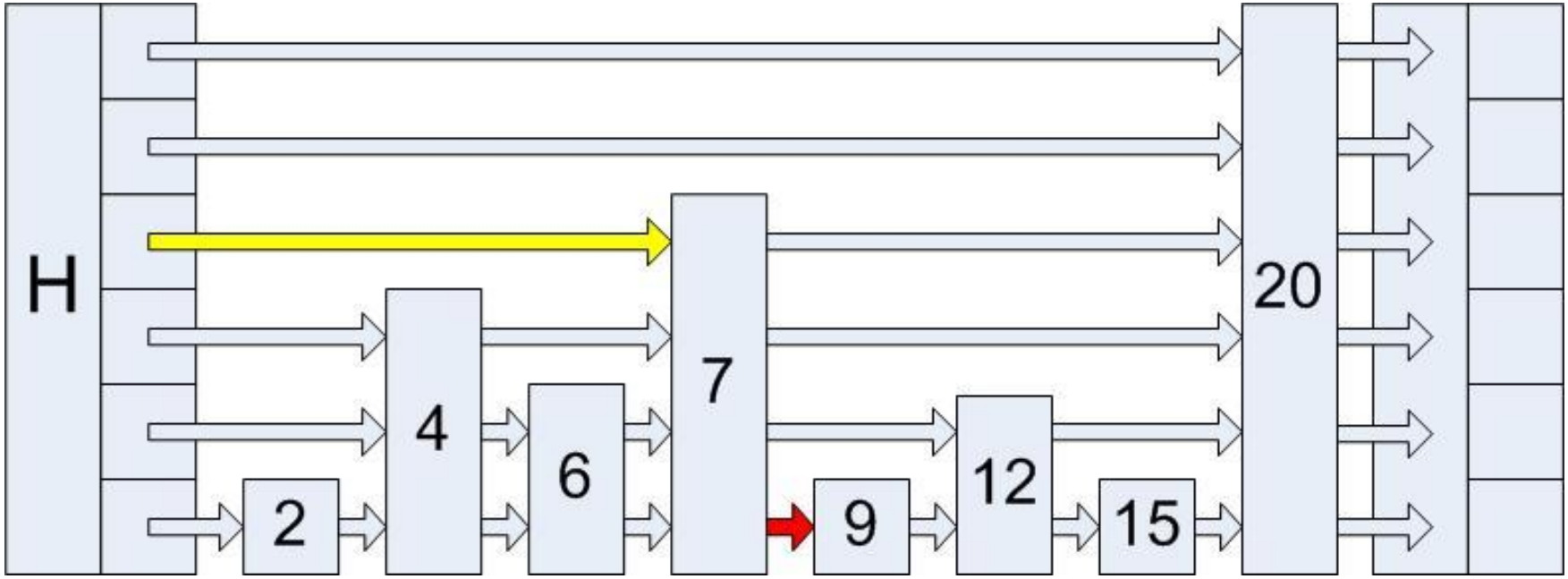
Skip List

- Search for the number 9:



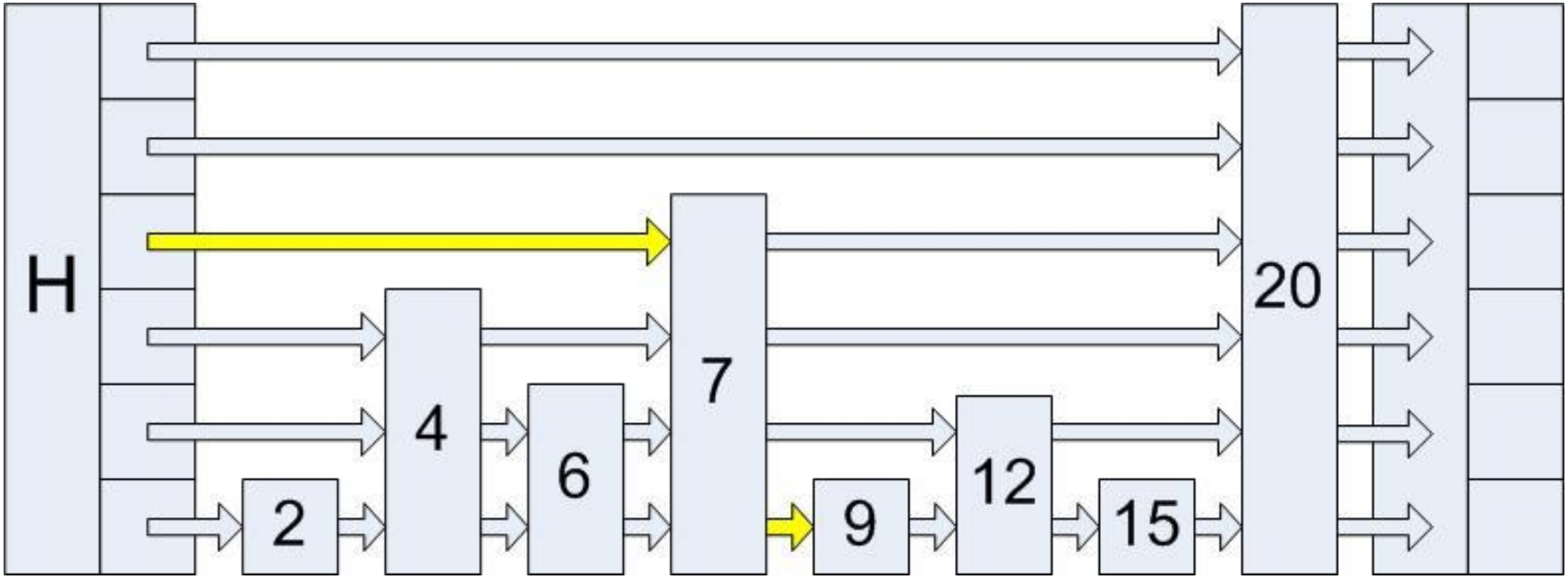
Skip List

- Search for the number 9:



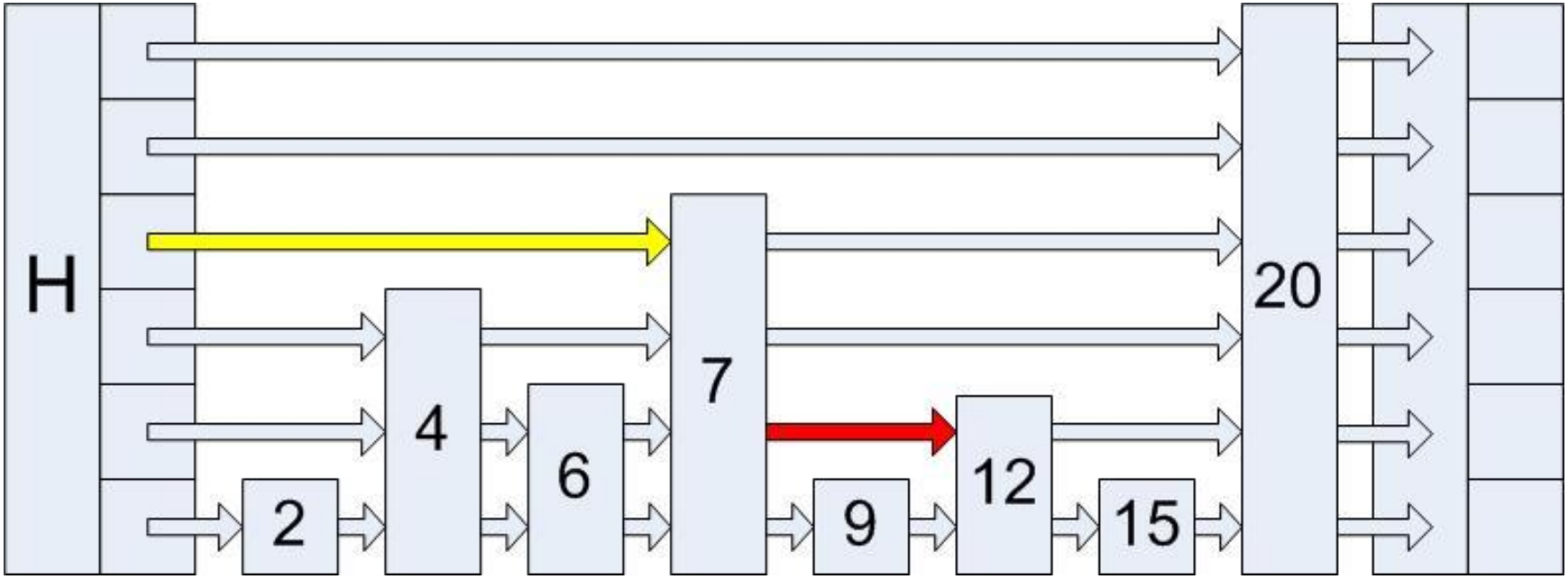
Skip List

- Search for the number 9:



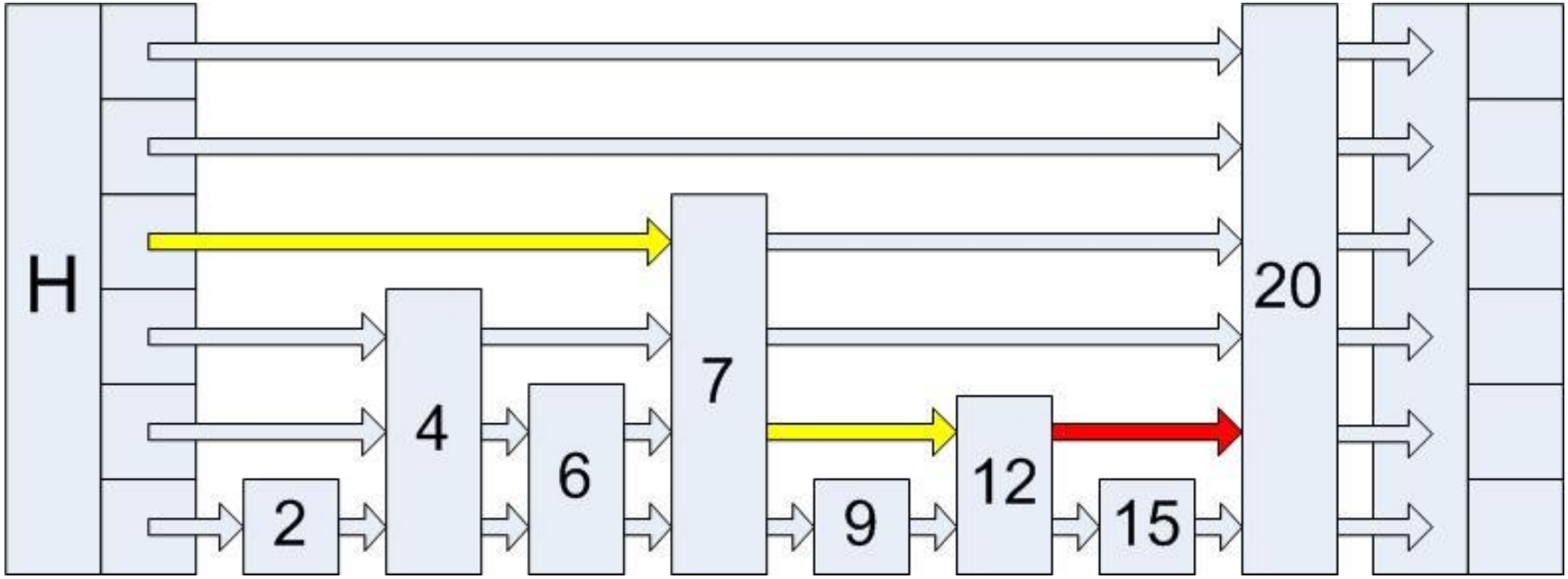
Skip List

- Search for the number 15:



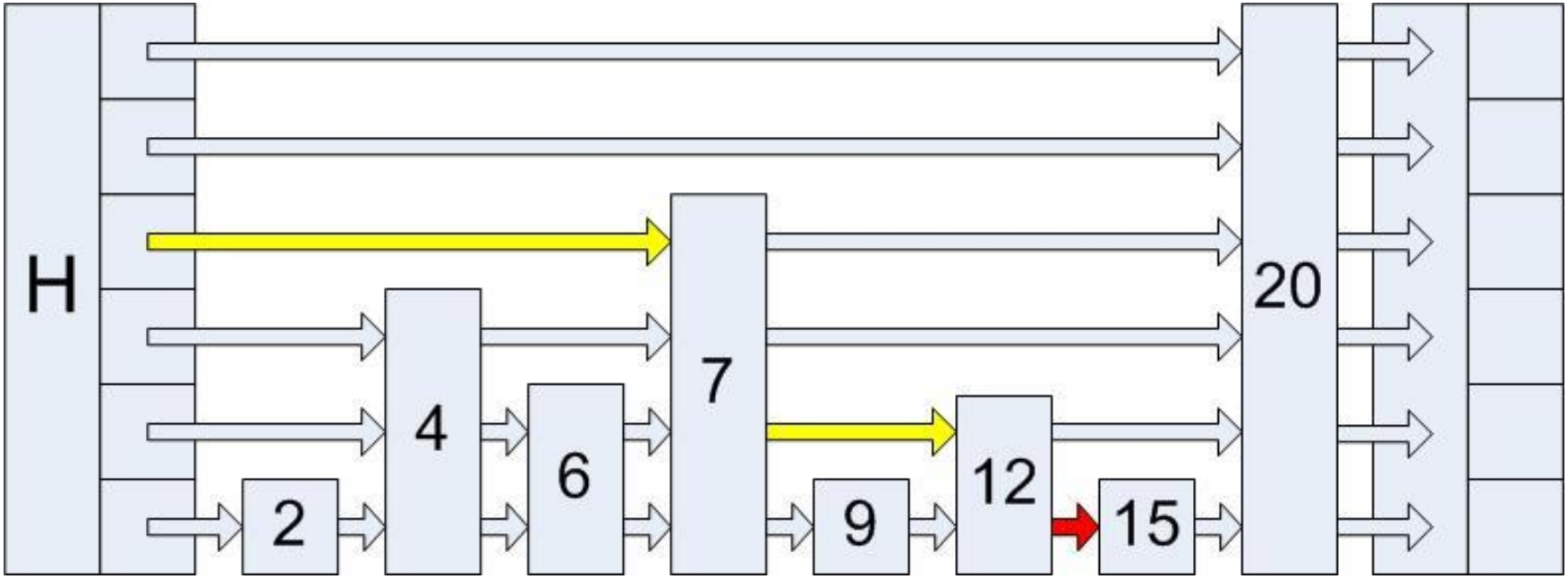
Skip List

- Search for the number 15:



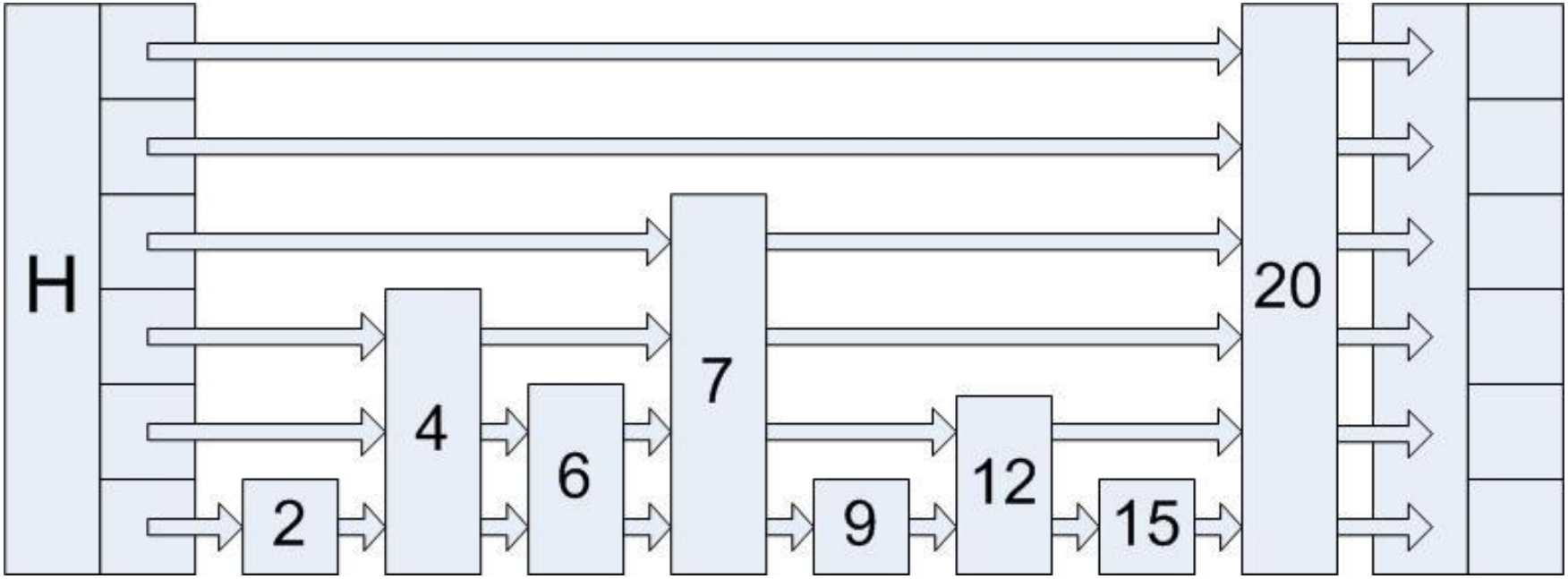
Skip List

- Search for the number 15:



Skip List

- How many comparisons do we make before we say the number 14 was not found?

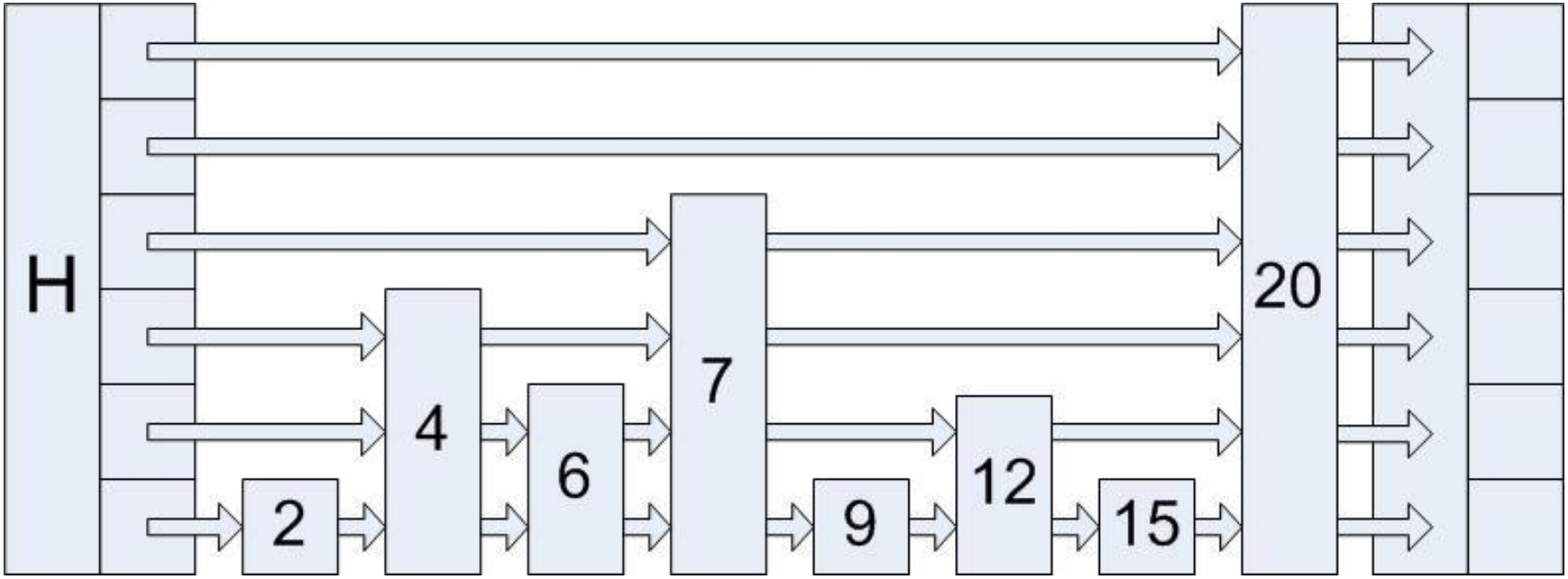


Outline

- Trees
- **Skip Lists**
  - Definition
  - Insertion
  - Search
  - **Deletion**
- Conclusion

Deletion

- Delete the number 7:



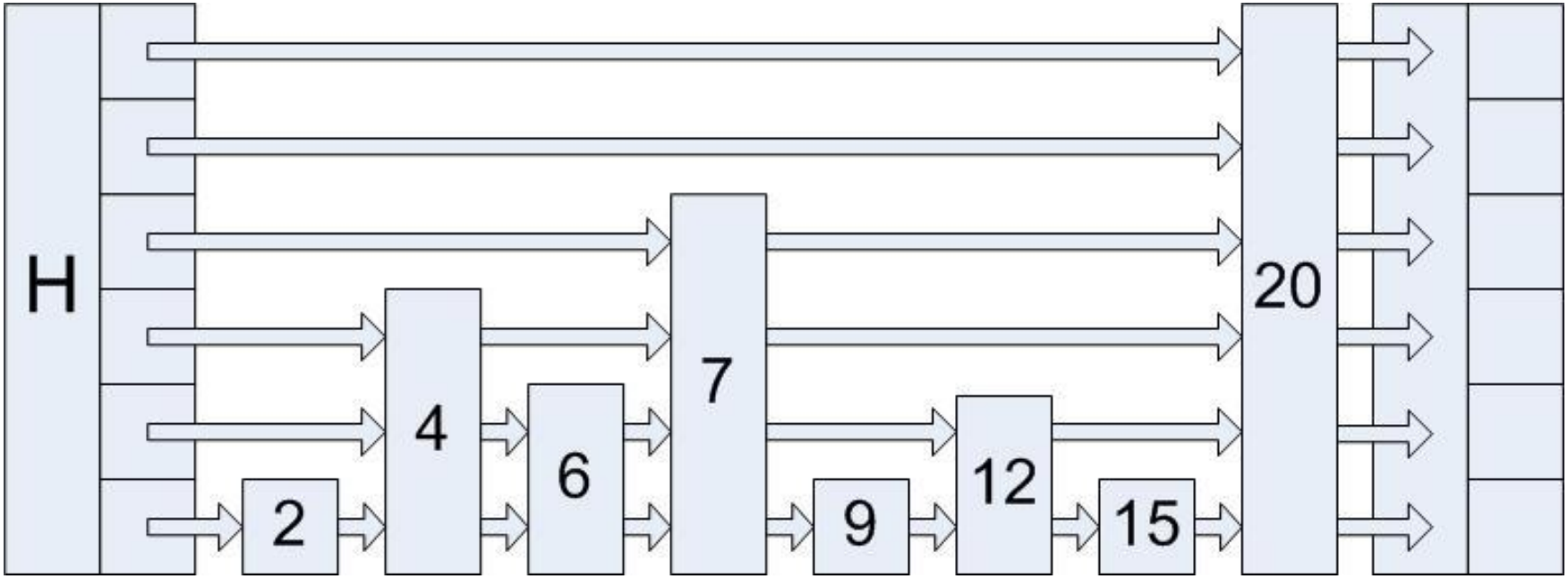
Deletion Algorithm

Delete *x* from skip list:

- Run search to find *x*.
- Delete node *x* and reassign arrows as necessary.

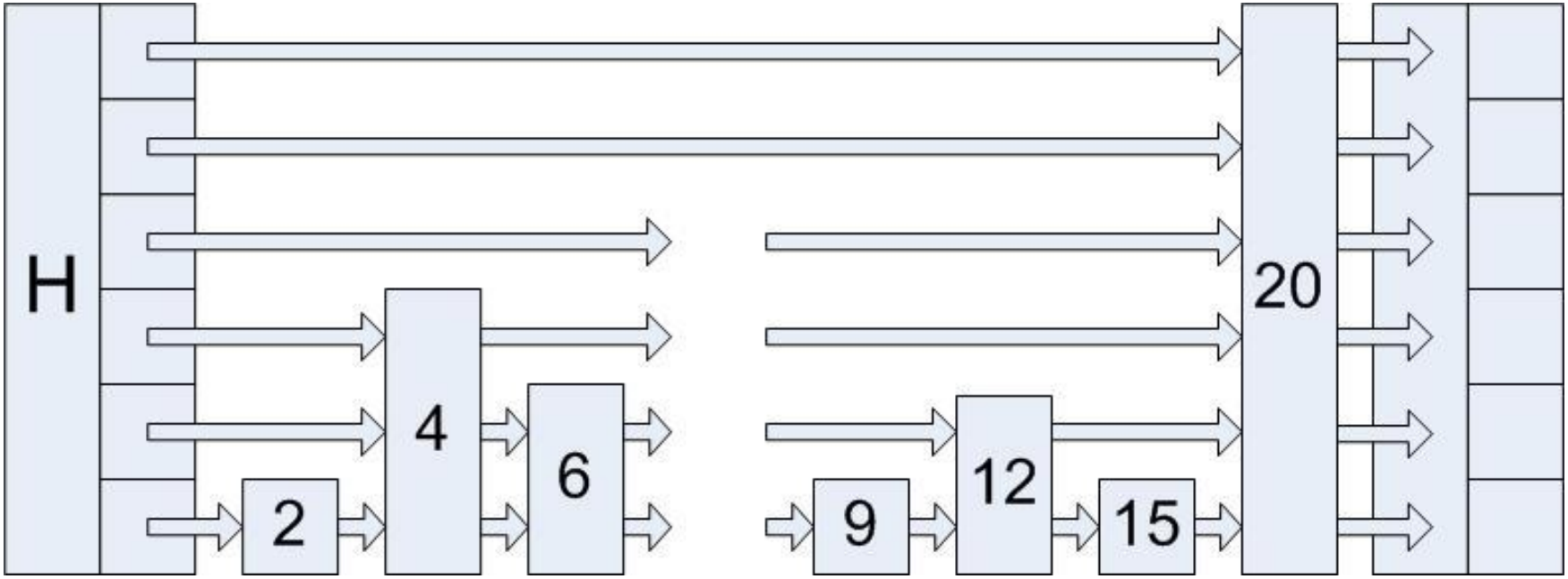
Deletion

- Delete the number 7:



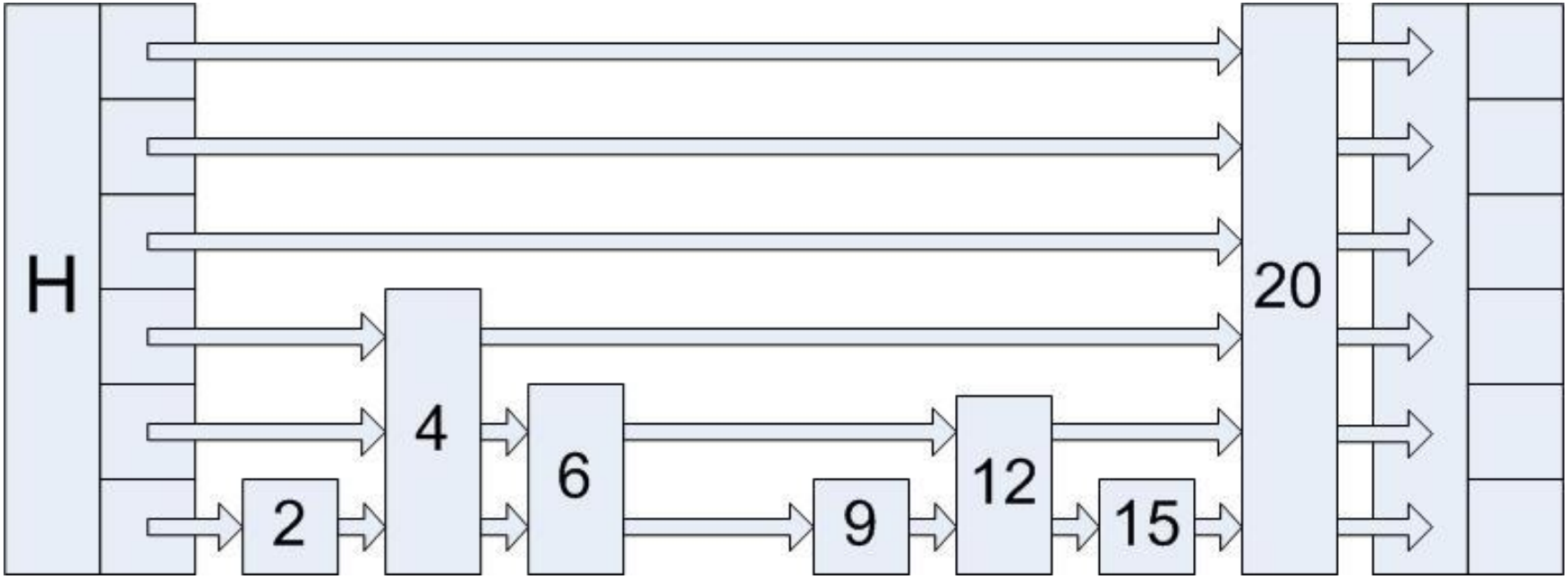
Deletion

- Delete the number 7:



Deletion

- Delete the number 7:



Outline

- Trees
- Skip Lists
  - Definition
  - Insertion
  - Search
  - Deletion
- **Conclusion**

Quick Questions

- Can the runtime for insert/delete/search be linear?
- What is a good level to set your maximum level to assuming you know roughly how many elements you will be inserting?
- If we have no maximum level, what level should we start our search at (assuming we know the number of nodes)?
- If I implement an InList(int x) function that returned a boolean, what is the last level I have to check to return false?