Data Structures and Algorithms
Skip List

CS 225
Brad Solomon

November 9, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

WCS Explorations Presents

dev ada

Dev Ada is WCS's annual project cycle! Teams will work with a project manager to create a project of their choice to present at EOH and the spring project showcase! Register or apply to be a project manager using the links below:

https://go.illinoiswcs.org/dev-ada-participant
https://go.illinoiswcs.org/dev-ada-pm

Final Project Reminders / Updates

Regrade late penalties accumulate weekly (Can see penalties on rubric)

"Weekly" is one week after you received feedback on your submission

Mid-project checkins will be available either Nov 16-18 or Nov 28-30

Your mentor will communicate their schedule to you directly

Final project final deadline extended to December 12th

Learning Objectives

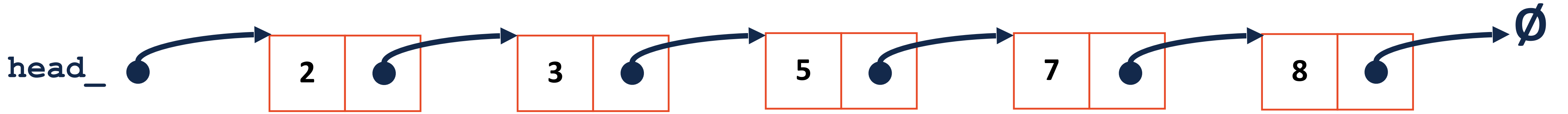
Motivate and introduce the skip list ADT

Conceptualize and code core functions

Analyze efficiency of skip list

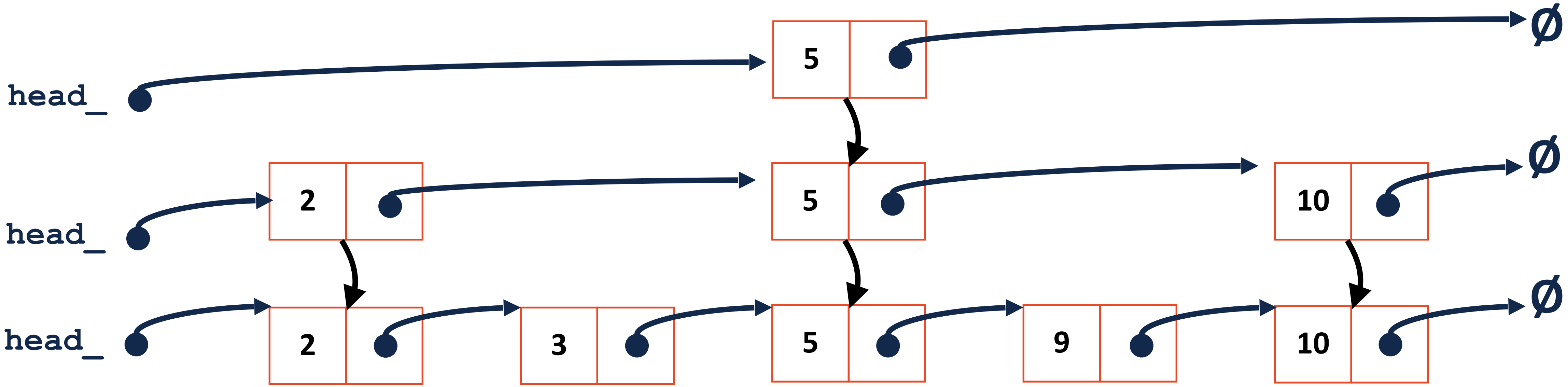
Linked List with ‘Checkpoints’

With some small overhead costs, we can store **checkpoints**.



Linked List with Perfect Checkpoints

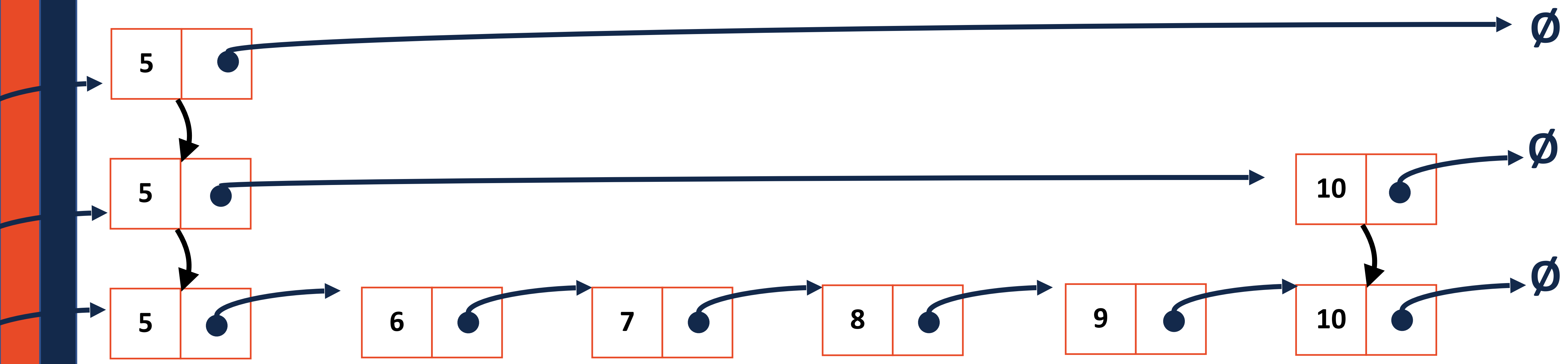
For optimal checkpoints, we want half the number of items at each level.



Linked List with Perfect Checkpoints

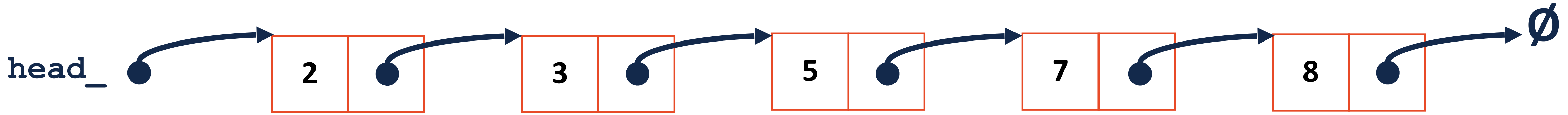
For optimal checkpoints, we want half the number of items at each level.

Maintaining this while inserting and deleting is too costly!



Linked List with Random Checkpoints

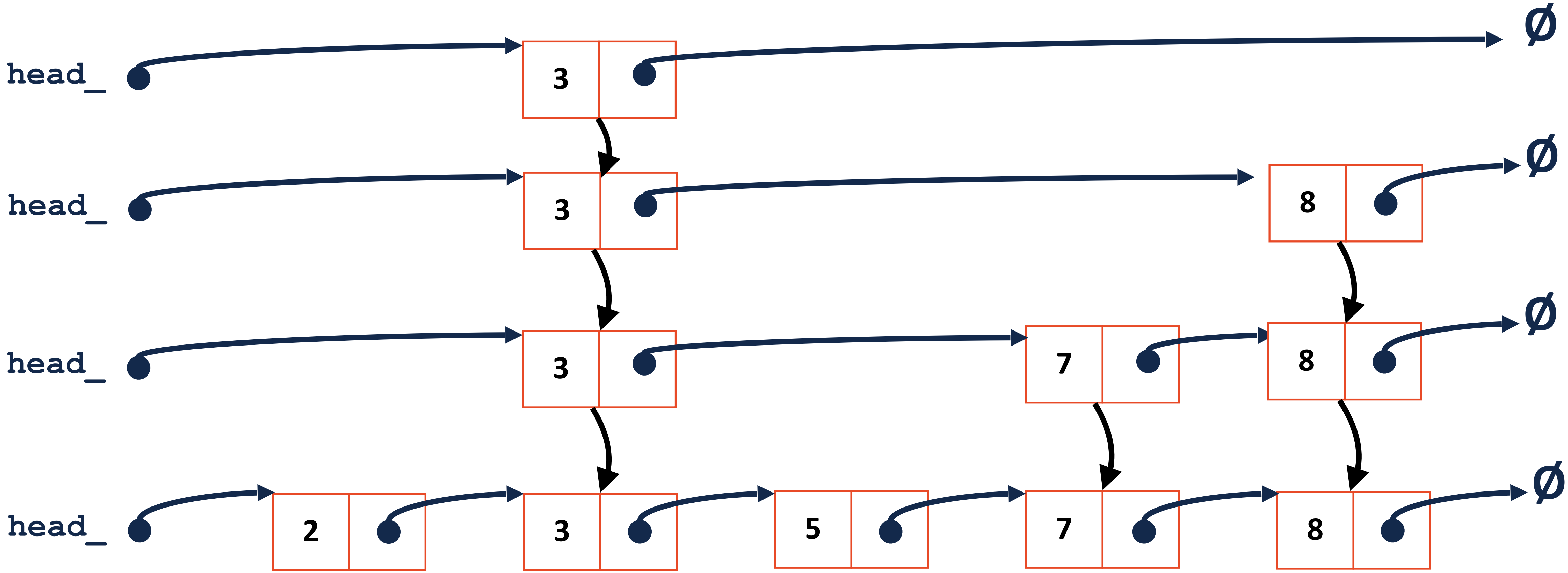
Instead of having **exactly** half each level, let's have **approximately** half!



Linked List with Random Checkpoints

Instead of having **exactly** half each level, let's have **approximately** half!

To analyze runtimes we use: ________________________



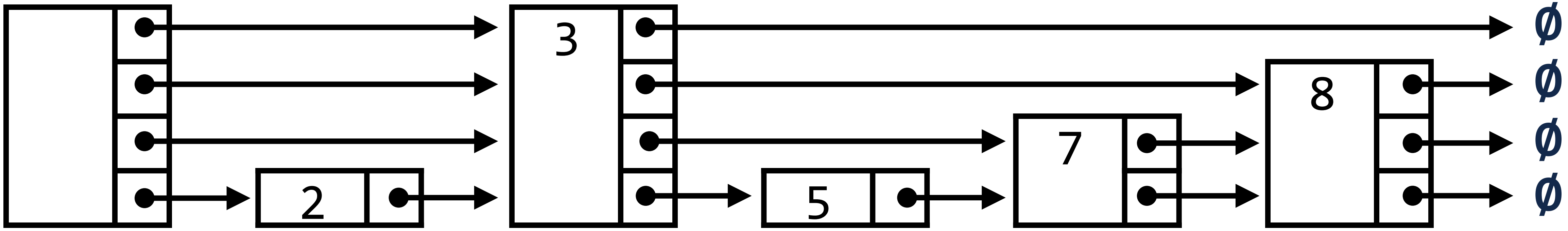
The Skip List

An ordered linked list where each node has variable size

Each node has at most one key but an arbitrary number of pointers

The decision for height is **randomized**

Claim: The **expected** time to insert, search, or delete is $O(log\ n)$

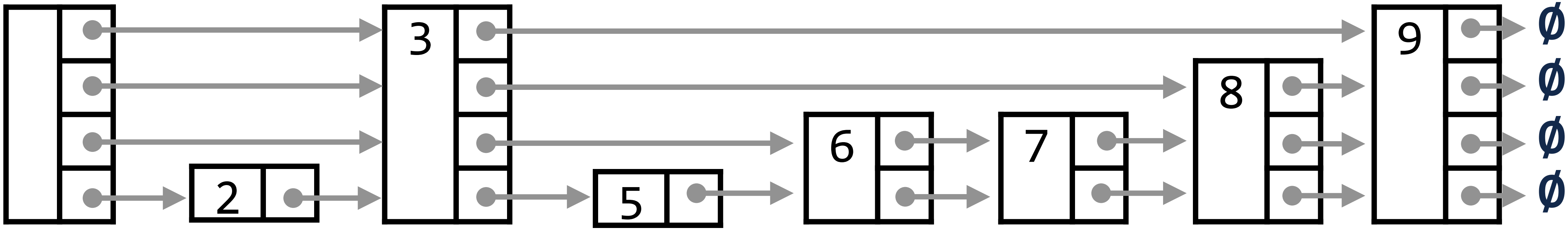


Skip List

```
template <class T>
class SkipList{
  public:
    class SkipNode{
      public:
        SkipNode(){
          next.push_back(nullptr);
        }

        SkipNode(int h, T & d){
          data = d;
          for(int i = 0; i <= h; i++){
            next.push_back(nullptr);
          }
        }
        T data;
        std::vector<SkipNode*> next;
    };

    int max; // max height
    float c; //update constant
    SkipNode* head;
    ...

```

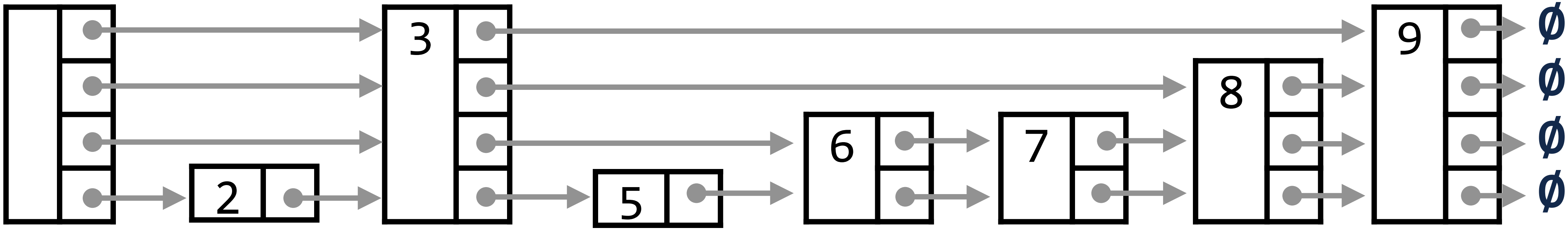
Skip List Find

`Find(9)`



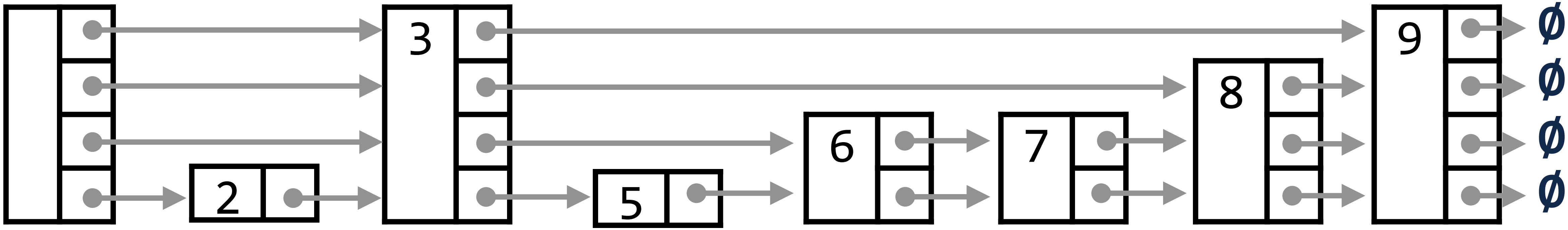
Skip List Find

Find(7)



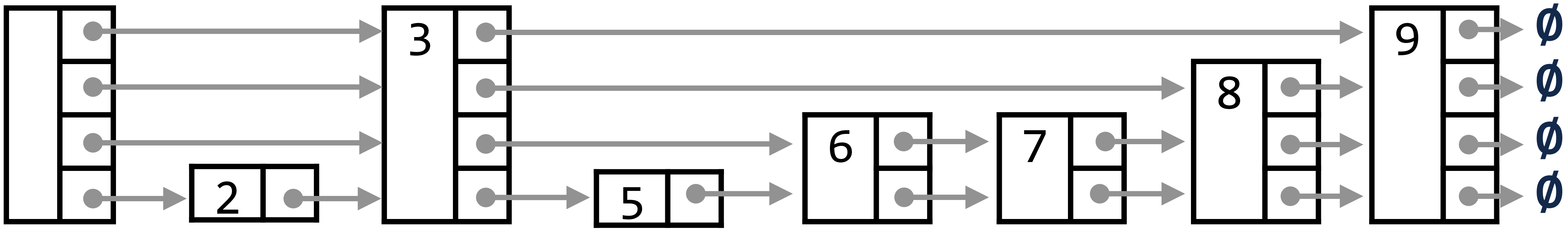
Skip List Find

`Find(1)`



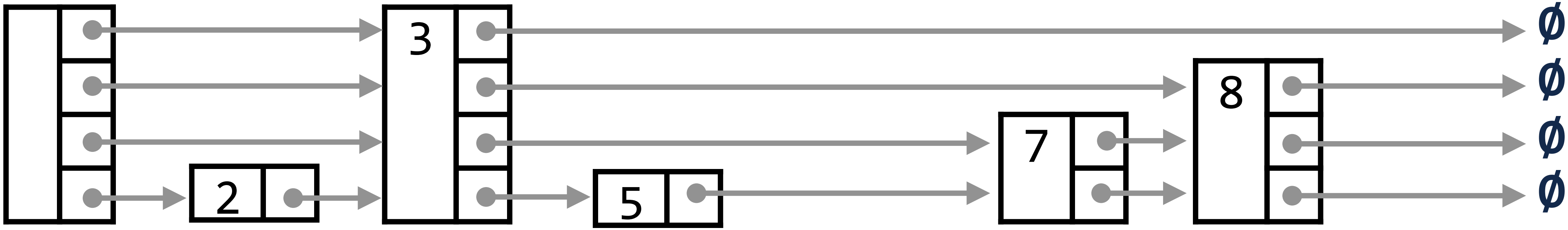
Skip List Find

```
template <class T>
bool SkipList<T>::find(T data){

  SkipNode* curr = head;

  for(int i = max; i >= 0; i--){
    while(curr->next[i]!= nullptr && curr->next[i]->data < data ){
      curr = curr->next[i];
    }
  }

  curr = curr->next[0];

  if (curr != nullptr && curr->data == data){
    return true;
  }
  return false;
}
```



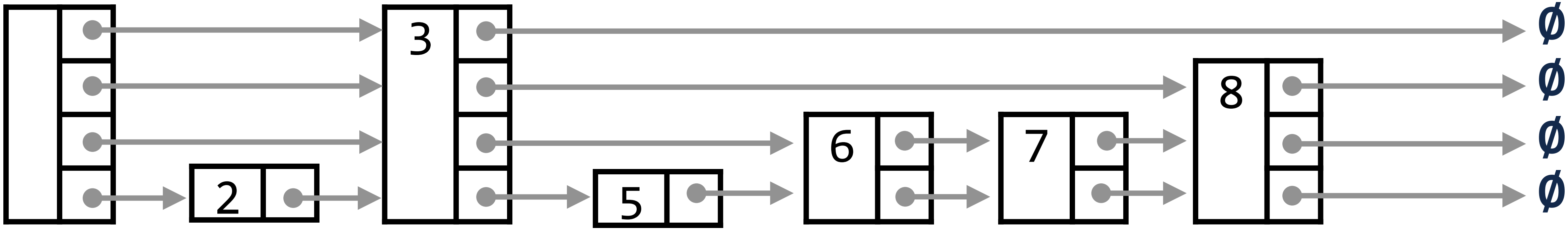
Skip List Insert

Insert(6)

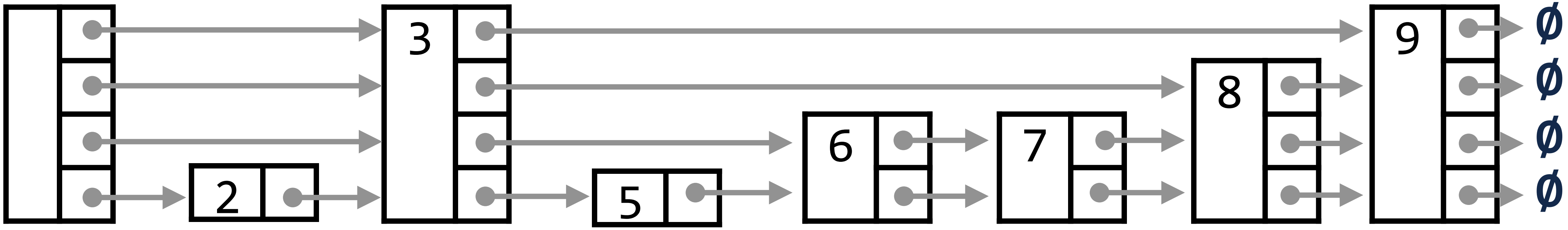


Skip List Insert

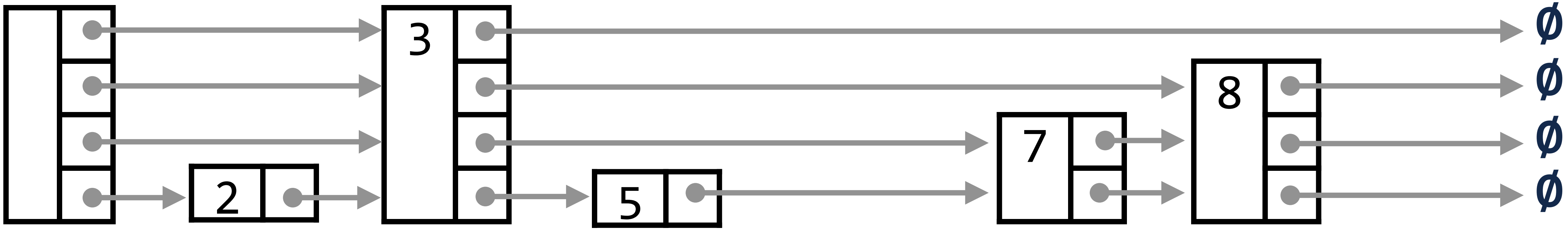
`Insert(9)`



Skip List Insert

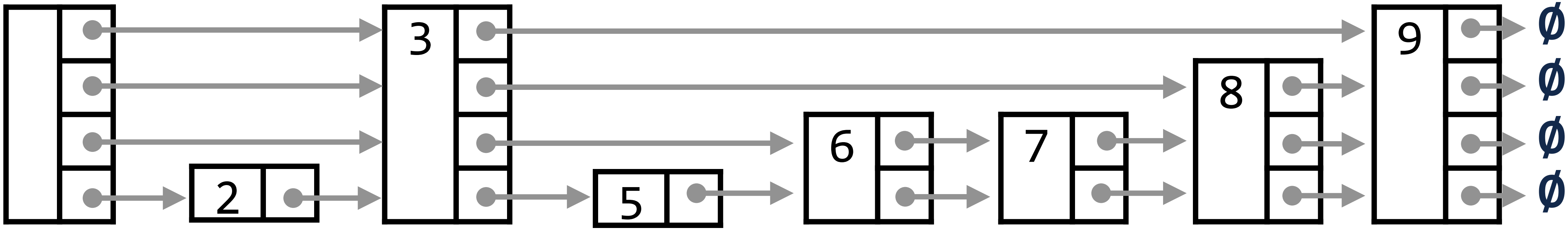


```
void SkipList<T>::insert(T data){
  int h = randHeight();
  SkipNode* n = new SkipNode(h, data);
  SkipNode* curr = head;

  for(int i = max; i >= 0; i--){
    while(curr->next[i]!= nullptr && curr->next[i]->data < data){
      curr = curr->next[i];
    }
    if (h >= i){
      curr->next[i]=n;
      n->next[i]=nextNode;
    }
  }
  if (h > max){
    int diff = h-max;
    for(int i = 0; i < diff; i++){
      (head->next).push_back(n);
    }
    max = h;
  }
}
```



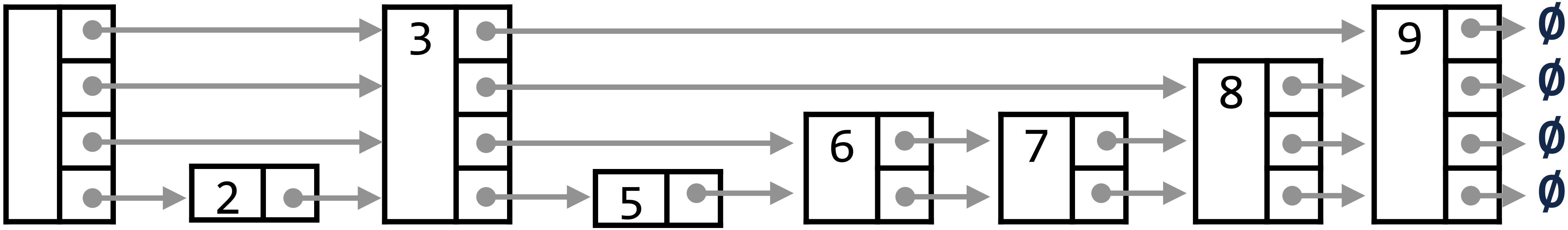
Skip List Remove

Remove(9)



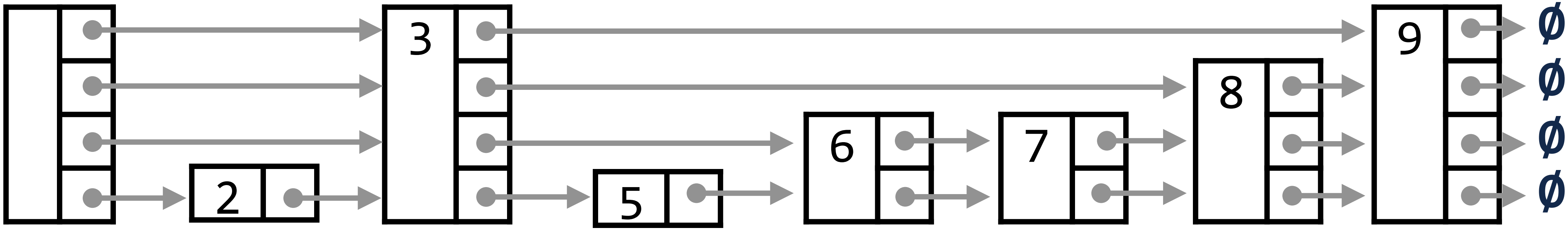
Skip List Remove

Remove(3)



Skip List Remove

Remove(5)



Skip List Expectation

Lets assume our skip list uses a coin flip for randomness `(c=0.5)`

Claim: Expected size of a node is 2.

Skip List Expectation

Lets assume our skip list uses a coin flip for randomness `(c=0.5)`

Claim: Expected size of skip list is 2n.

Skip List Expectation

Claim: Expected height of skip list is $O(log\ n)$

Skip List Expectation

Claim: Expected height of skip list is $O(log\ n)$

$$E[h] = \sum_{l=0}^{\lceil log\ n \rceil} E[I_l] + \sum_{l=\lceil log\ n \rceil + 1}^{\infty} E[I_l] \quad I_l = \begin{cases} 1 & l\text{th level contains a node} \\ 0 & l\text{th level empty} \end{cases}$$

Skip List Expectation

Claim: Expected length of search of skip list is $O(log\ n)$