CSCE-42603

Advanced Data Structures

Homework Review Assignment 4

Fall 2025

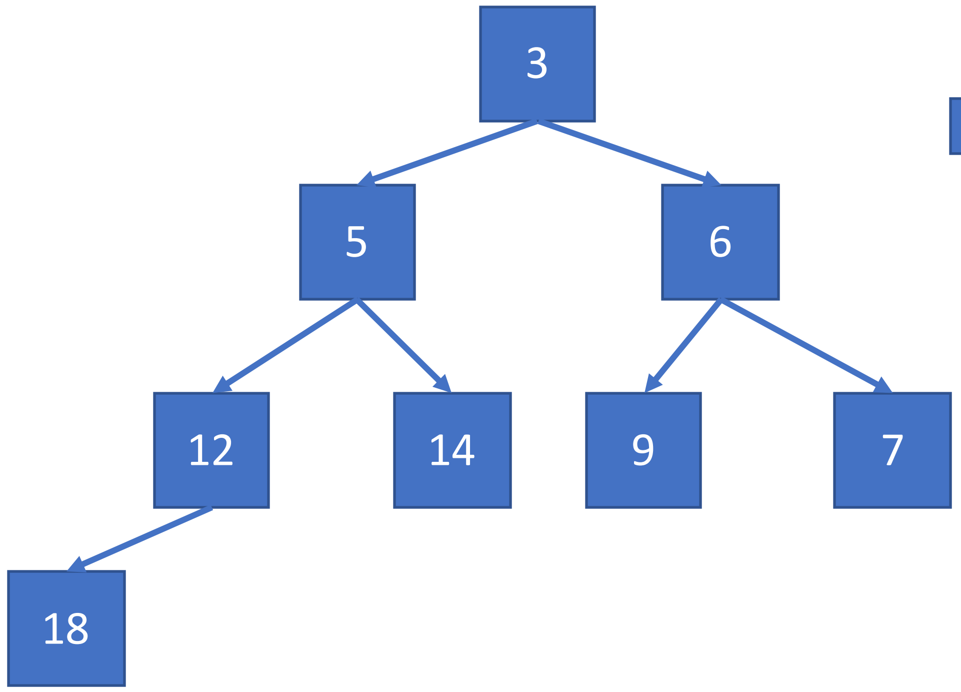
Prof. Khoa Luu

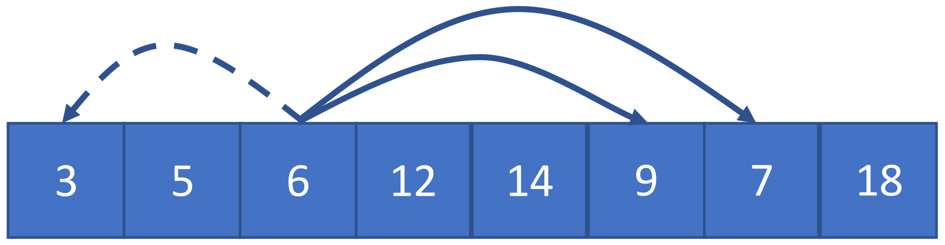
Dr. Thanh-Dat Truong

Disclaimer and Note

- The purpose of this review is to provide students with the approach to solving the homework, not giving the solution
- This review can be considered as suggestions/hints
- You are welcome to implement you own ideas as long as it is able to solve the homework correctly
- You can add/implement any addition functions if you think it is necessary

Heap Organization

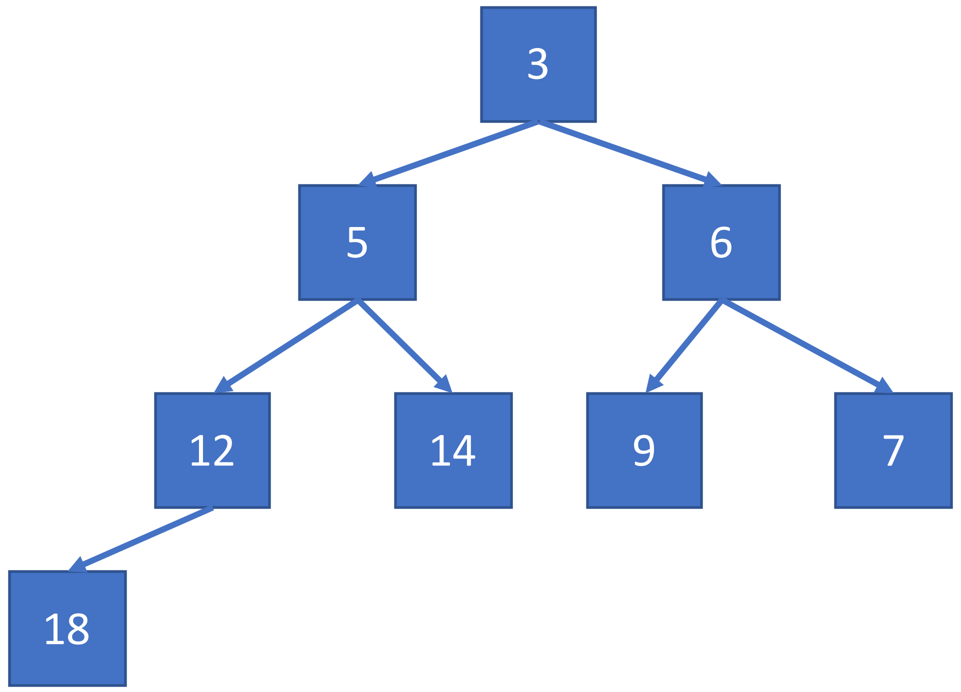




i

1-based Index	$\frac{i}{2}$	i	$2i \quad 2i+1$
0-based Index	$\frac{i+1}{2} - 1$	i	$2i+1 \quad 2i+2$

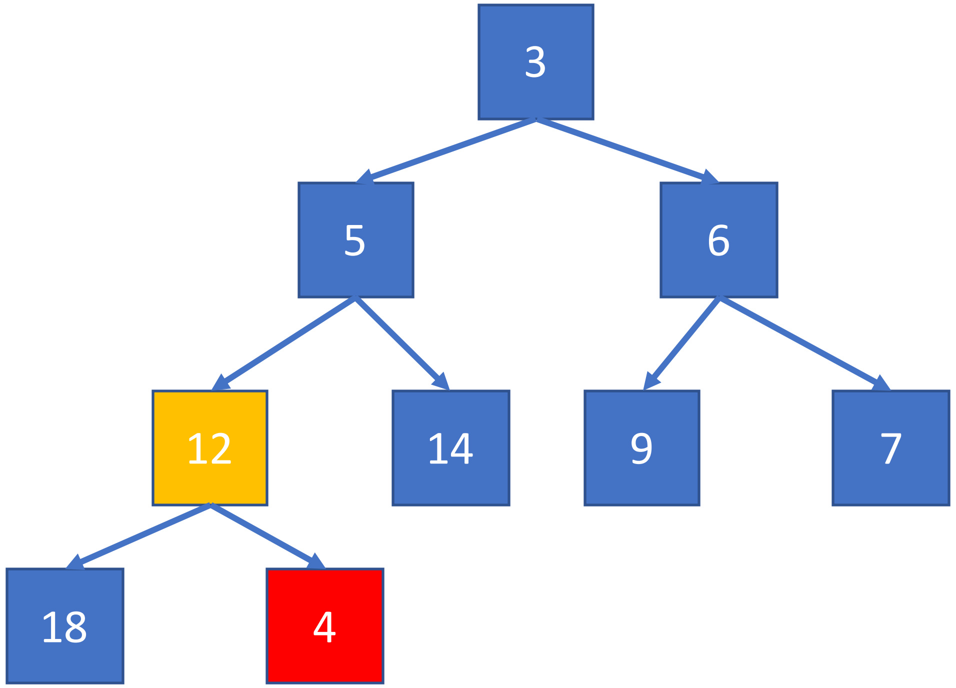
Top



- Simply return the top of the binary tree

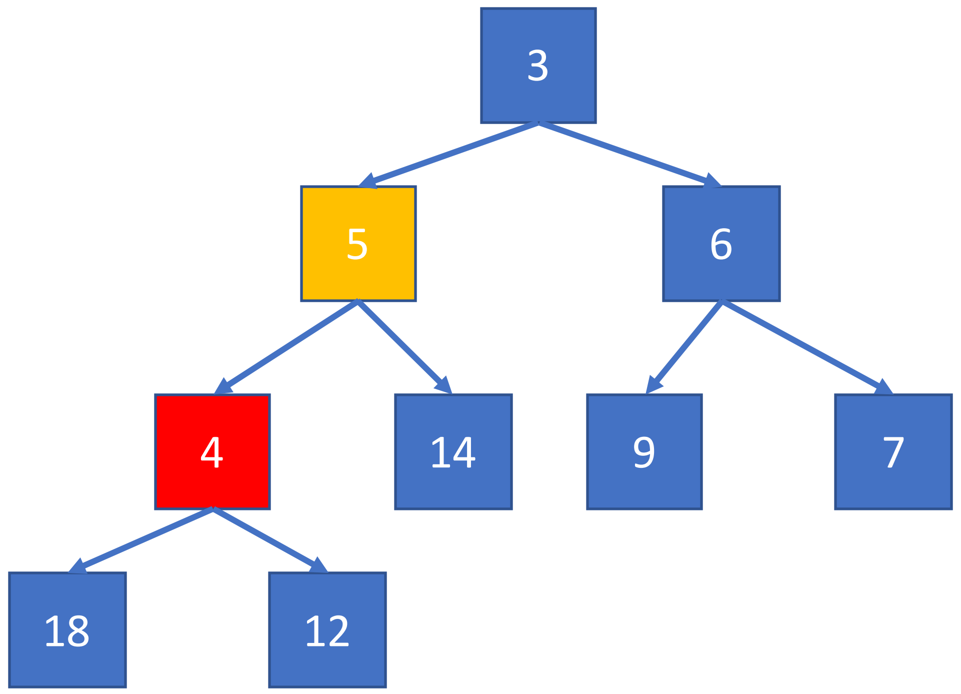
```
Node MinHeap::top() {
   return top of the heap
}
```

Push



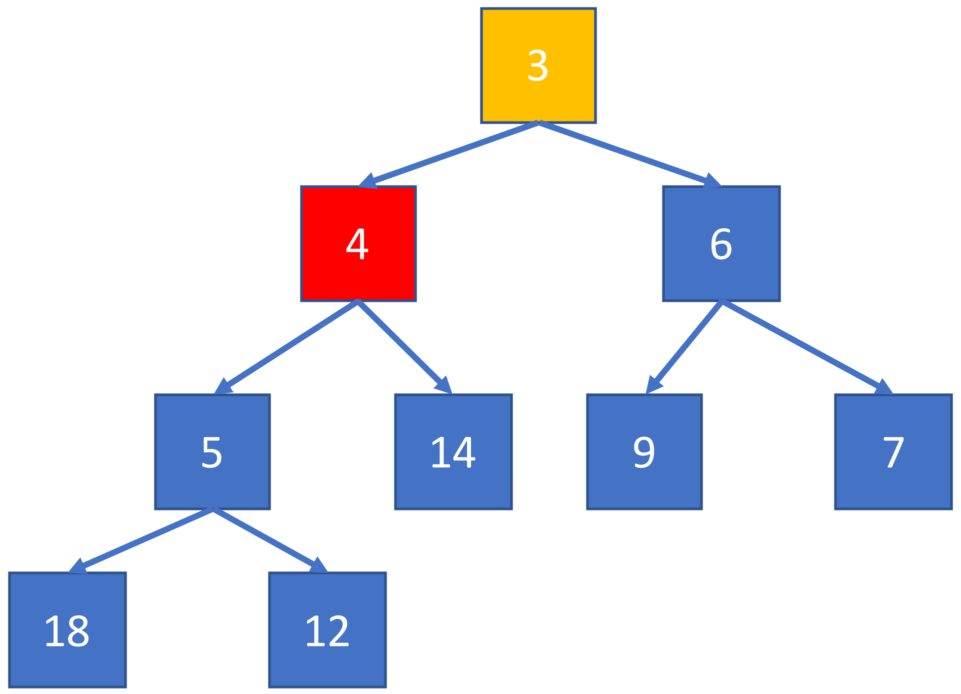
```
void MinHeap::push(int key, int vertex) {
  push the key and vertex of the end of heap
  children index = size of heap - 1;



}
```

Push



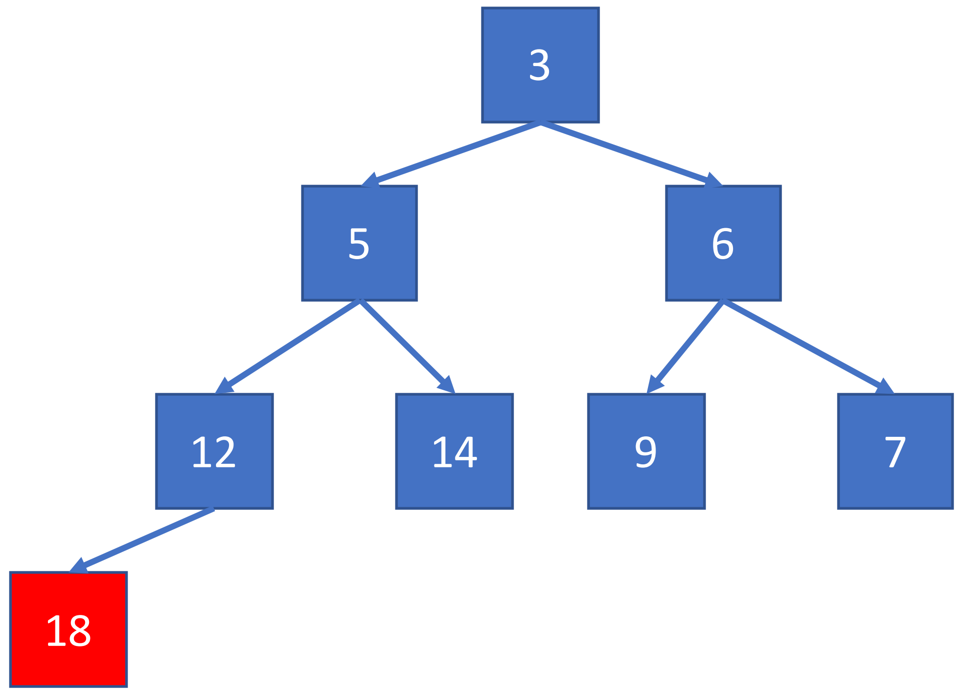
```
void MinHeap::push(int key, int vertex) {
  push the key and vertex of the end of heap
  children index = size of heap - 1;
  while (index > 0) {









  }
}
```

Push

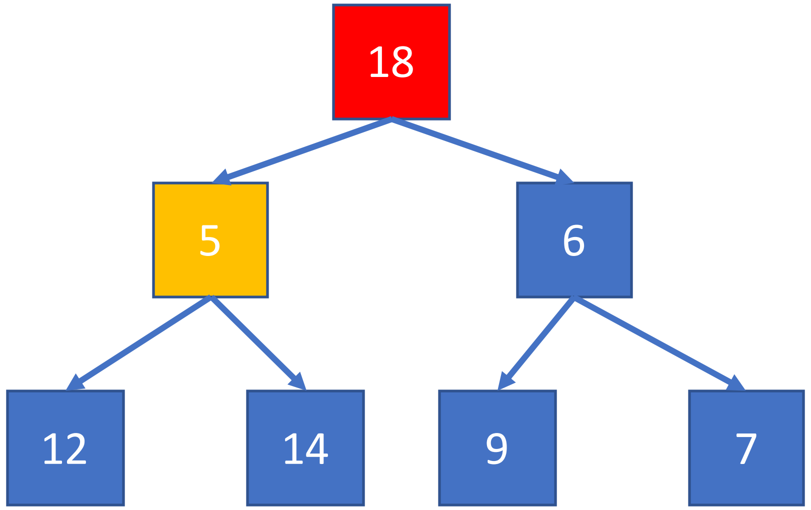


```
void MinHeap::push(int key, int vertex) {
  push the key and vertex of the end of heap
  children index = size of heap - 1;
  while (index > 0) {
    calculate parent index
    if (children key <  parent key) {
      swap parent and children
      chilren index = parent index;
    } else
        break;
  }
}
```

Pop

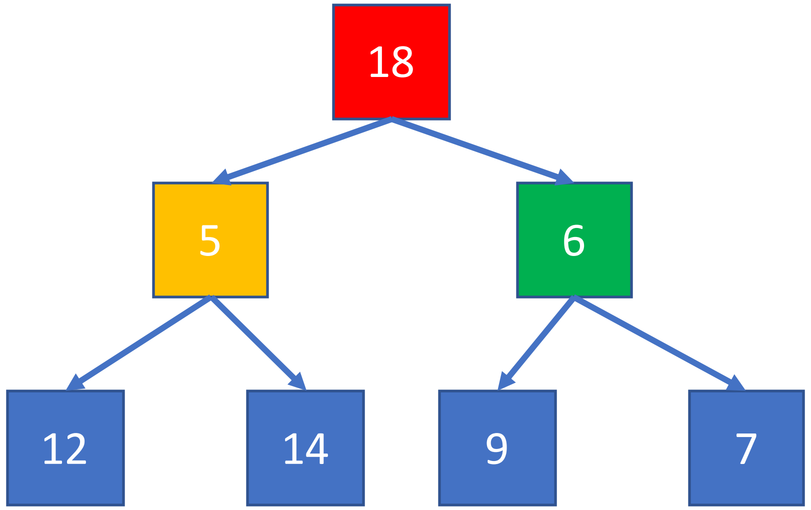


Pop



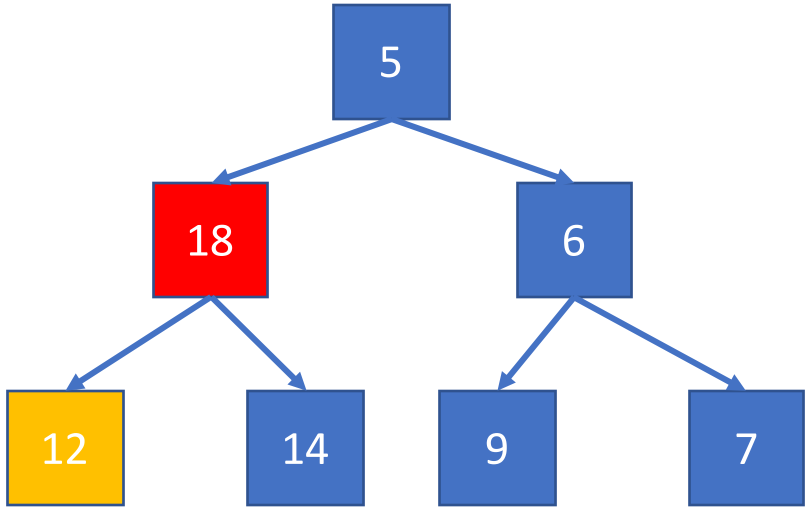
```
void MinHeap::pop() {
  N = size of heap





}
```

Pop



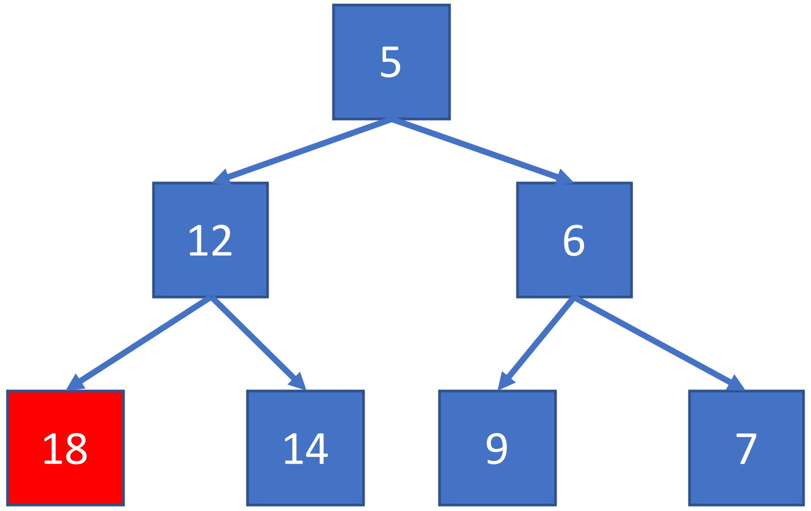
```
void MinHeap::pop() {
  N = size of heap
  swap top of heap and last of heap
  N = N - 1;
  remove the last element of heap
  parent index = 0;









}
```

Pop



```
void MinHeap::pop() {
  N = size of heap
  swap top of heap and last of heap
  N = N - 1;
  remove the last element of heap
  parent index = 0;
  while (child index < N) {







  }
}
```

Pop



```
void MinHeap::pop() {
 N = size of heap
 swap top of heap and last of heap
 N = N - 1;
 remove the last element of heap
 parent index = 0;
 while (child index < N) {
  current child = left child
  if (left child key > right child key)
    current child = right child
  if (current children key < parent key) {
   swap parent and current child
   parent = current child;
  } else
    break;
 }
}
```

Demo