Data structures and algorithms

Tutorial 3

Amr Keleg

Faculty of Engineering, Ain Shams University

March 8, 2019

Contact: amr_mohamed@live.com

Outline

Outline

How to insert an element at the beginning of an array?

```
void insert_at_beginning(int element, int arr[],
int arr_len, int max_arr_len){

  // INSERT the element

}
```

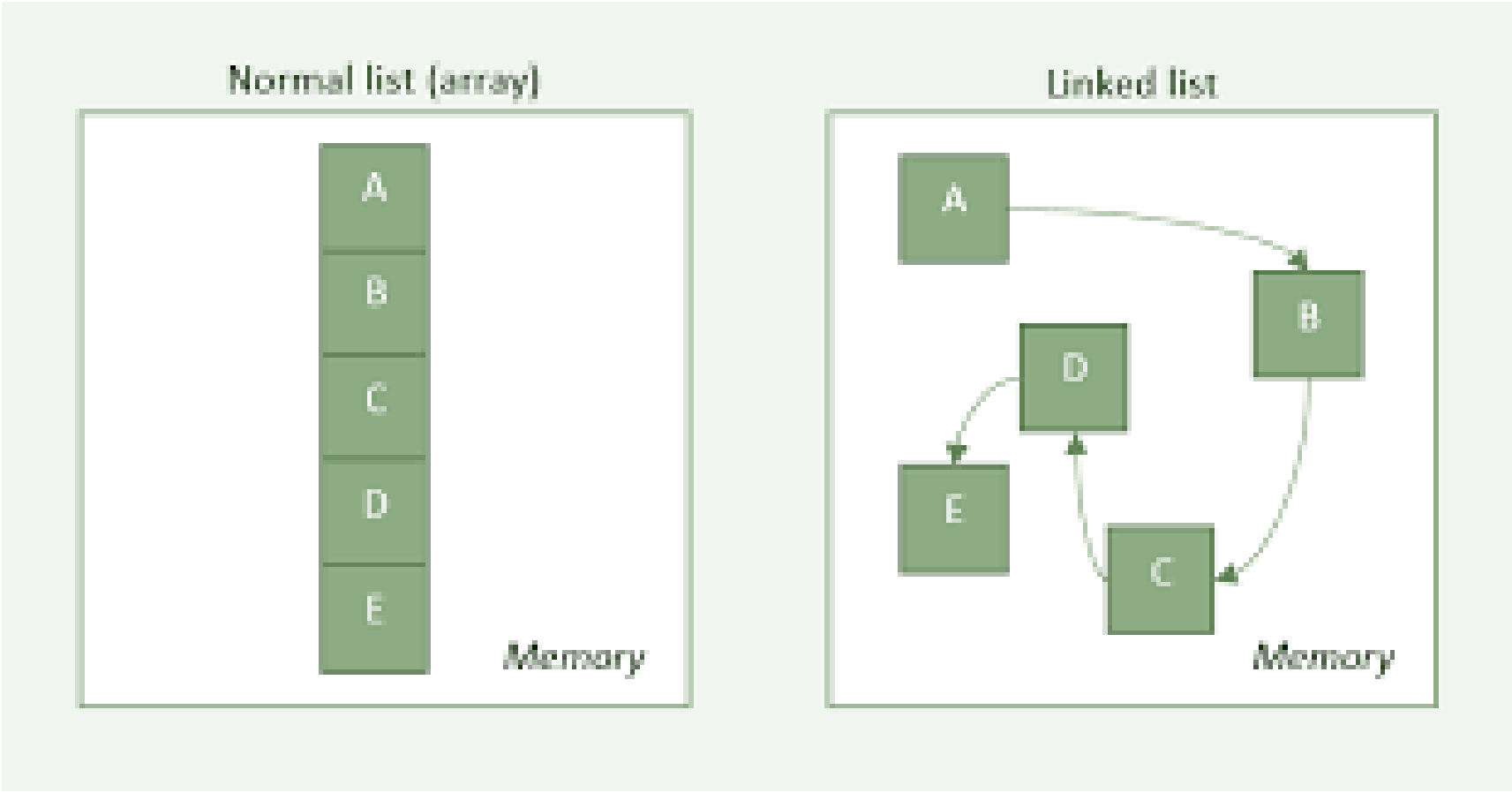
```
void insert_at_beginning(int element, int arr[],
int arr_len, int max_arr_len){

   for (int index=arr_len-1; index>=0; index--){
      arr[index + 1] = arr[index];
   }

   arr[0] = element;
}
```

The complexity is O(n).
What happens if the array has reached its maximum size?

1. Create a new array of bigger size.
2. Copy the data from the old array to the new one.
3. Add the new element at the beginning of the array.

(The same applies for appending an element to the end of the array).

Outline


Normal list (array)
A
B
C
D
E
Memory
Linked list
A
B
D
E
C
Memory

Outline

```
class Node{
  // carries data and pointer to the following element
};
```

```
class Node{
  private:
    int data;
    Node * next;

  public:
    Node(int d){
      this->data = d;
      this->next = nullptr;
    }

    void set_next(Node * next){
      this->next = next;
    }
    Node * get_next(){
      return next;
    }
    // Same for data
    friend class LinkedList;
};
```

Outline

```
class LinkedList{
  private:
    Node * head;

  public:
    LinkedList();
    bool empty();
    int length();
    void push_front(int d);
    void pop_front();
    void push_back(int d);
    void pop_back();
    void print();
    bool contains(int d);
    void clear();
    void bubble_sort();
    ~LinkedList();
};
```

```
LinkedList(){
  head = nullptr;
}
```

```
bool empty(){
  return head == nullptr;
}
```

```
void push_front(int d){
  Node * new_head = new Node(d);
  new_head->next = head;
  head = new_head;
}
```

```
int length(){
  Node * cur_node = head;
  int length = 0;
  while(cur_node != nullptr){
    length++;
    cur_node = cur_node->next;
  }

  return length;
}
```

```
void pop_front(){
  if (this->empty()){
    return;
  }
  Node * head_to_be_deleted;
  head_to_be_deleted = head;
  head = head->next;
  delete head_to_be_deleted;
}
```

```
void push_back(int d){
  if(empty()){
    push_front(d);
  }
  else{
    Node * last_node = head;
    while(last_node->next != nullptr){
      last_node = last_node->next;
    }
    last_node->next = new Node(d);
  }
}
```

```
void pop_back(){
  if(empty()){
    return;
  }
  Node * pre_last_node = nullptr;
  Node * last_node = head;
  while(last_node->next !=nullptr){
    pre_last_node = last_node;
    last_node = last_node->next;
  }

  delete last_node;

  if (pre_last_node!=nullptr){
    pre_last_node->next = nullptr;
  }
  else{
    head = nullptr;
  }
```

```
void clear(){
  while(!empty()){
    pop_front();
  }
}
```

```
void bubble_sort(){
  if(empty() || head->next == nullptr)
    return;
  for(int it=0; it<length() - 1; it++){
    Node * pre_last_node = head;
    Node * last_node = head->next;
    while(last_node != nullptr){
      if (pre_last_node->data > last_node->data){
        swap(pre_last_node->data, last_node->data);
      }
      pre_last_node = last_node;
      last_node = last_node->next;
    }
  }
}
```

```
~LinkedList(){
   clear();
}
```

Outline

```
// Implement print recursively
void print_node(Node * n){
  if (n==nullptr)
  {
    cout<<"\n";
    return;
  }
  cout<<n->data<<" ";
  print_node(n->next);
}

    void print(){
  print_node(head);
}
```

```
bool contains(int d){
  if (empty())
    return false;
  Node * cur_node = head;
  while(cur_node != nullptr){
    if (cur_node->data == d)
      return true;
    cur_node = cur_node->next;
  }
  return false;
}
```

```
// Try implementing these functions
// Delete the element at index
void delete_at(int index);

// Insert a new element after the node at index
void insert_at(int index, int d);
```

Visualizations of linked list: `https://visualgo.net/bn/list`

Outline

STL (Standard Template Library):

- Vectors:
  https://syntaxdb.com/ref/cpp/vectors
  List of functions:
  http://www.cplusplus.com/reference/vector/vector/
- Lists:
  http://www.cplusplus.com/reference/list/list/

Reference:
Data Structure and Algorithm Analysis in C++, 3rd edition, Mark Allen Weiss.

- 1.5 C++ Details
- 1.6 Template

Feedback form: `https://goo.gl/forms/Y3LtLnAL9FAi2LDp1`