Data structures

Algorithms eBook part 2

by Adam Higherstein

Code School

Dynamical Data Structures

Linked list

Heap memory is used.

Can grow and shrink
(basic arrays have fixes size)

You can add new elements to the beginning of linked list, or to the end
or to the middle.
Element can be removed from the beginning, from the end and from the middle.

Very flexible data structures!

Singly linked list

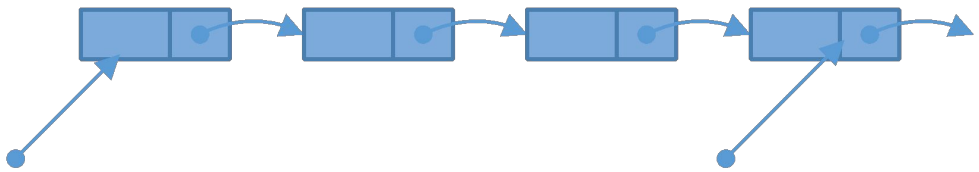
One direction only

Code School

Dynamical Data Structures

Example: let's create this

Linked list



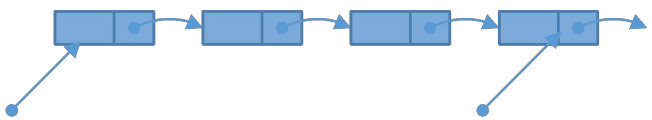
Code School

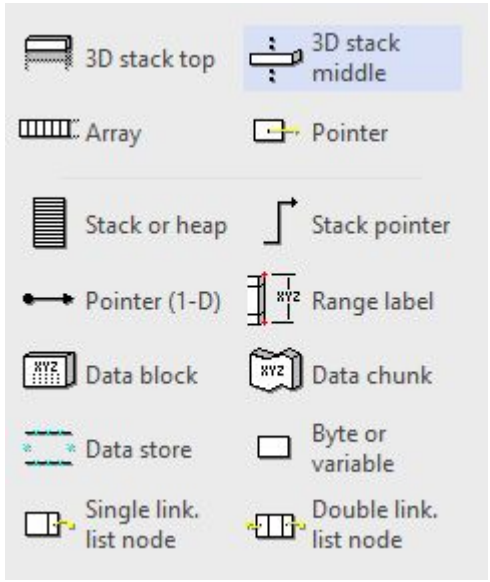
Dynamical Data Structures

Linked list

What tool to use?
E.g. Ms Visio has symbols for memory objects

Example: let's create this





Code School

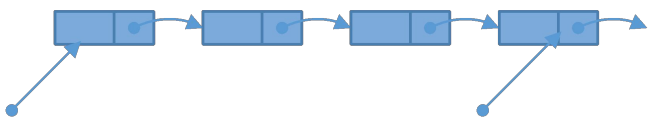
Dynamical Data Structures

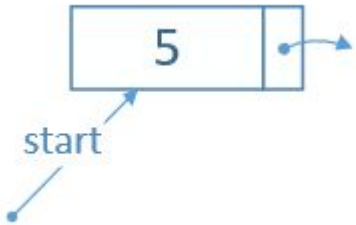
Linked list

Let's create our example list step by step

First element is created and reference start point to it

Example: let's create this





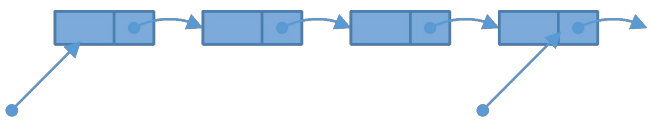
Code School

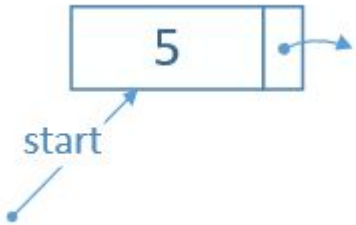
Dynamical Data Structures

Linked list

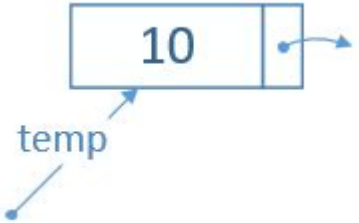
Let's create our example list step by step

Example: let's create this





First element is created and reference start point to it



Second element is created by using an extra reference named temp

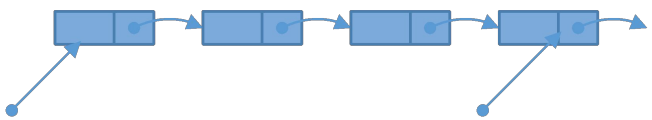
Code School

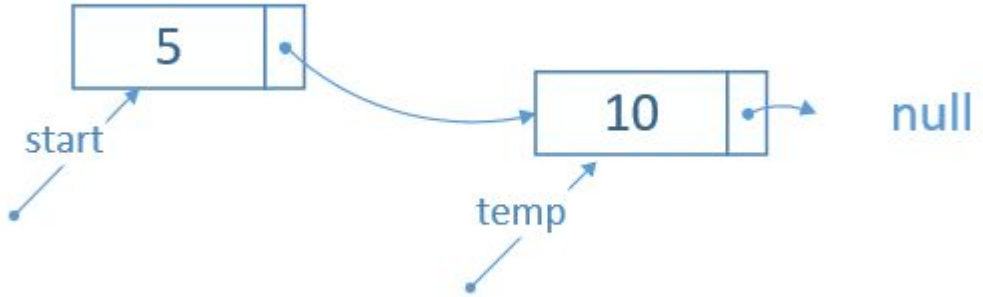
Dynamical Data Structures

Linked list

Let's create our example list step by step

Example: let's create this





Let's connect new element to our list.
And if we do not know if there are coming new elements, we add there null_ last element points to null

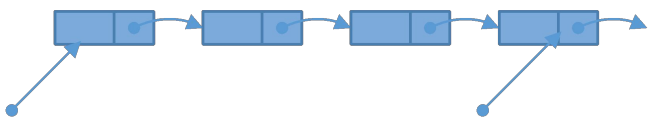
Code School

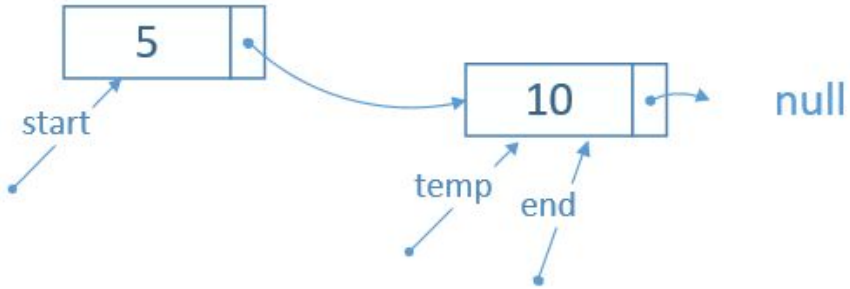
Dynamical Data Structures

Linked list

Let's create our example list step by step

Example: let's create this





We put there also reference to mark the Last element

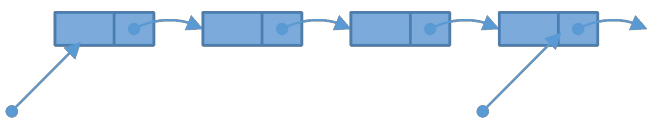
Code School

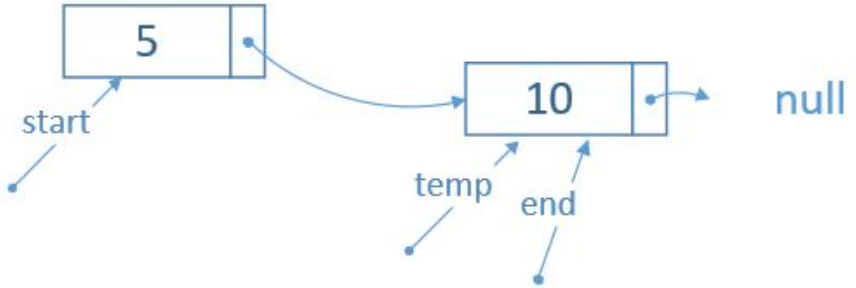
Dynamical Data Structures

Linked list

Let's create our example list step by step
next element, value is 400

Example: let's create this





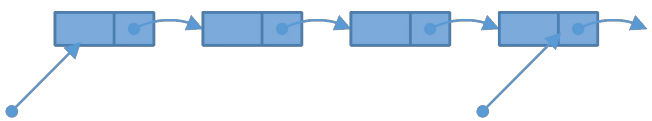
Code School

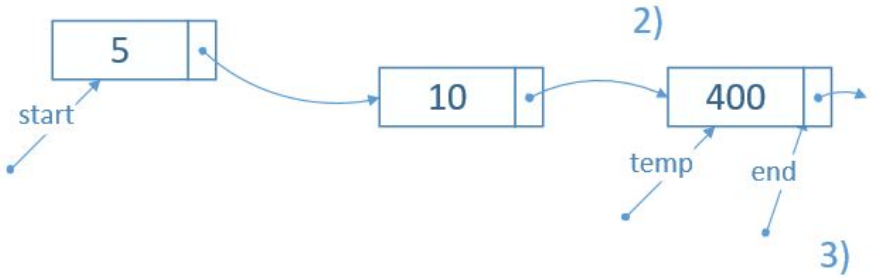
Dynamical Data Structures

Linked list

Let's create our example list step by step
next element, value is 400

Example: let's create this





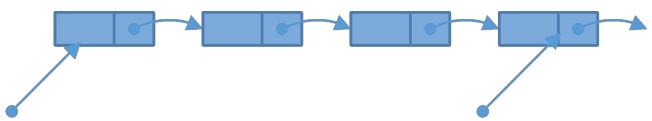
Code School

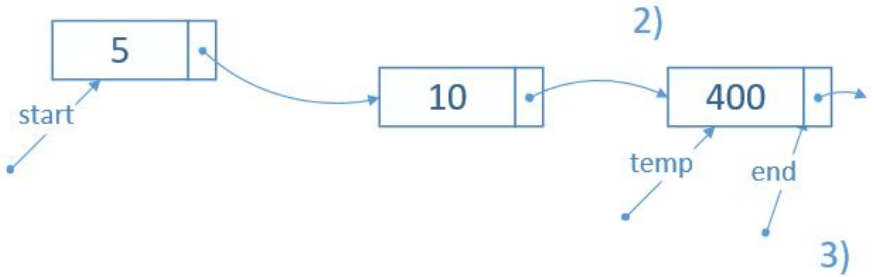
Dynamical Data Structures

Linked list

Let's create our example list step by step
next element, value is 400

Example: let's create this





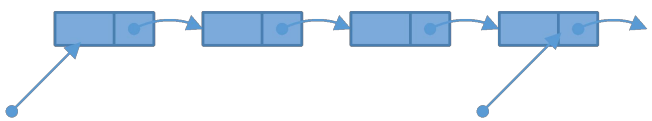
Code School

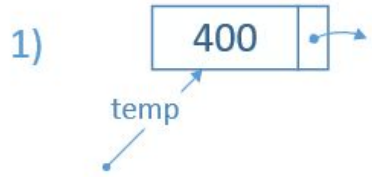
Dynamical Data Structures

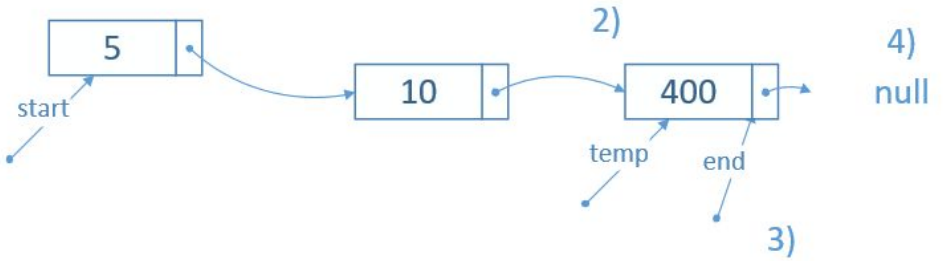
Linked list

Let's create our example list step by step
next element, value is 400

Example: let's create this







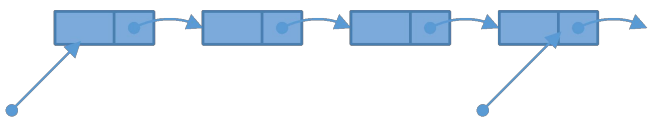
Code School

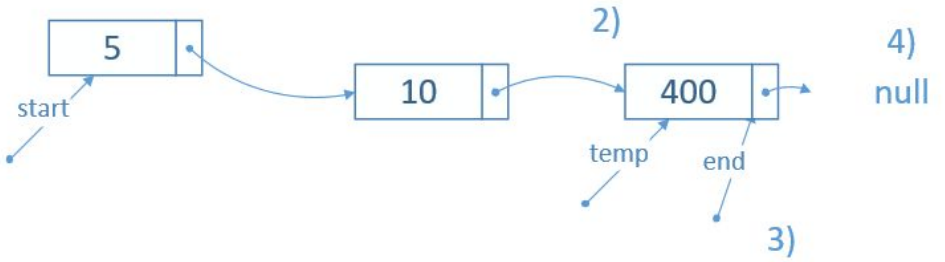
Dynamical Data Structures

Linked list

Let's create our example list step by step
Go on and the final element there!

Example: let's create this

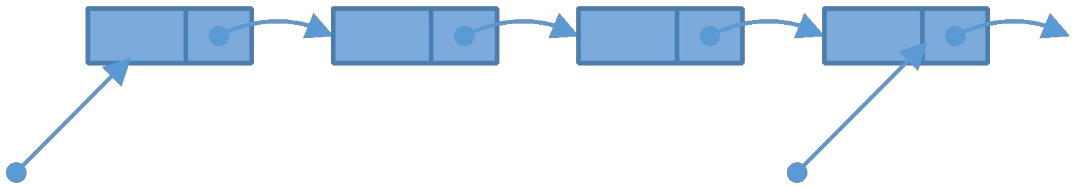




Code School

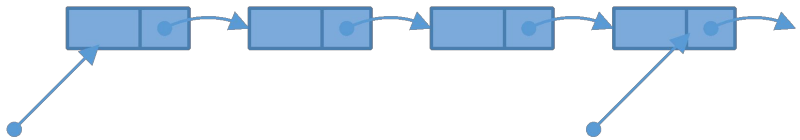
Dynamical Data Structures

Linked list



Let's create our example list step by step
Result

Code School



Dynamical Data Structures

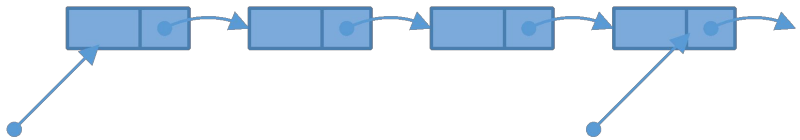
Linked list

Code 1)

```
class Element
{
    public int data;
    public Element next;
}
```

Code School

Dynamical Data Structures



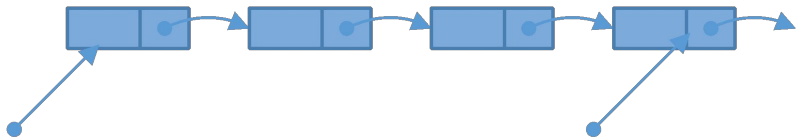
Linked list

```
class Element
{
    public int data;
    public Element next;
}
```

Code 2)

```
Element start = new Element();
start.data = 5;

Element temp = new Element();
temp.data = 10;

start.next = temp;
temp.next = null;

Element end = temp;
```

Code School



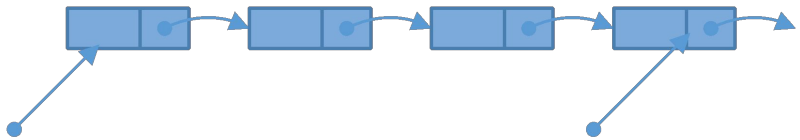
Dynamical Data Structures

Linked list

Code 3)

```
Element end = temp;

temp = new Element();
temp.data = 400;

end.next = temp;
end = temp;
end.next = null;
```

Code School



Dynamical Data Structures

Linked list

Now: go on and complete the code.
Then: print values!

Code School

Dynamical Data Structures

Linked list

Heap memory is used.

Can grow and shrink
(basic arrays have fixes size)

You can add new elements to the beginning of linked list, or to the end
or to the middle.
Element can be removed from the beginning, from the end and from the middle.

Very flexible data structures!

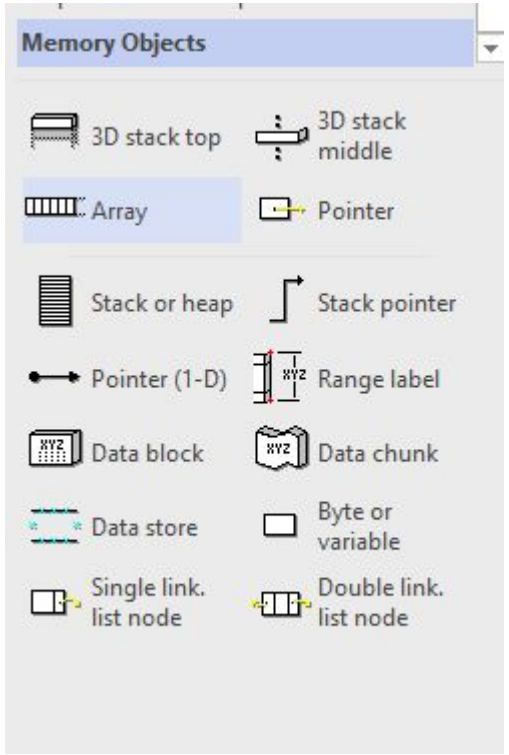
Doubly linked list

Code School

Dynamical Data Structures

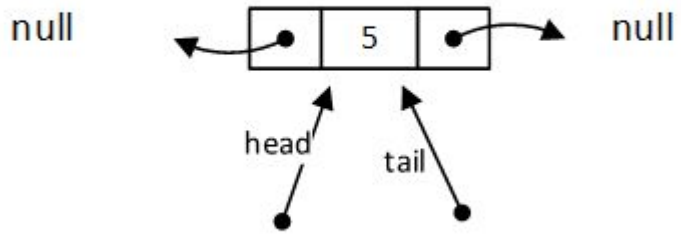
Doubly linked list

Ms Visio has memory object symbols!



Code School

Dynamical Data Structures

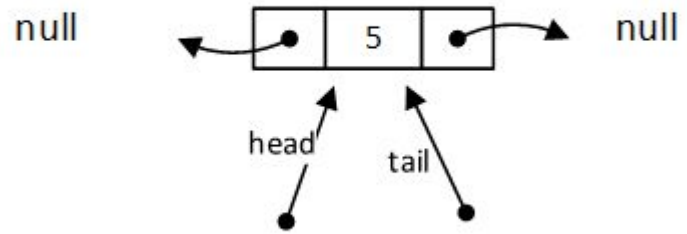


Doubly linked list

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Code School

Dynamical Data Structures



Doubly linked list

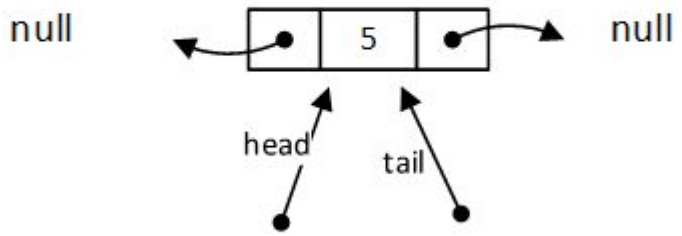
Create the
1. element

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
Element head = new Element();
head.data = 5;
head.next = null;
head.prev = null;
Element tail = head;
```

Code School

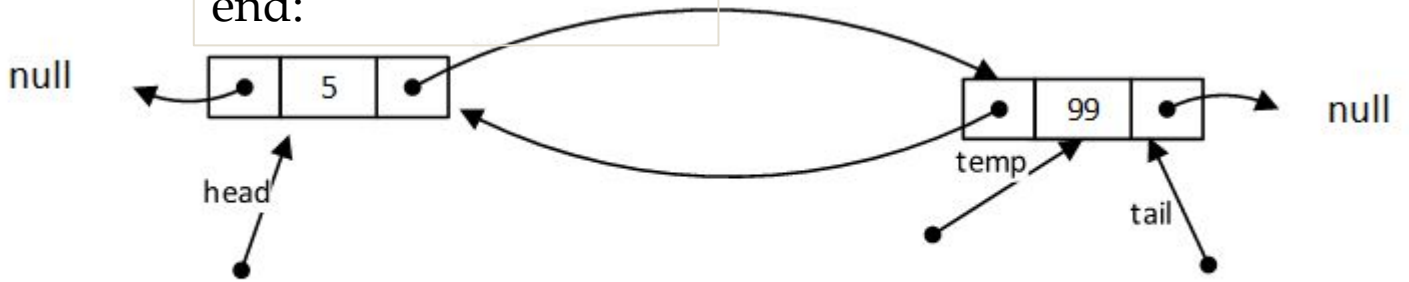
Dynamical Data Structures



Doubly linked list

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Let's add a new node (value is 99) to the end:

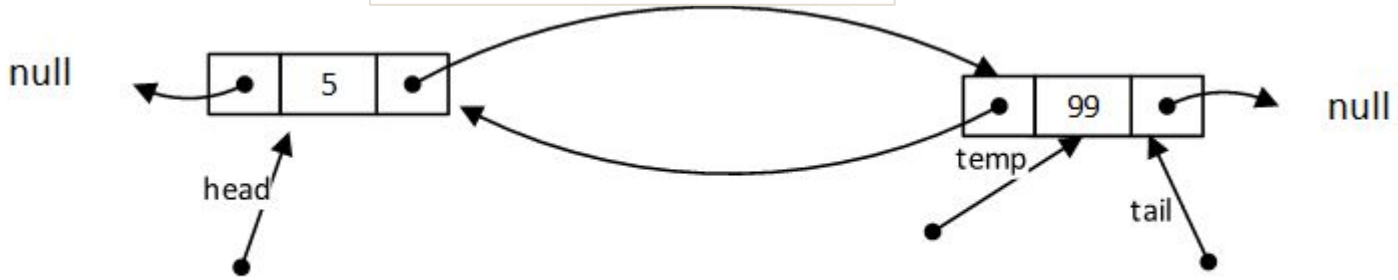


Code School

Dynamical Data Structures

Doubly linked list

Let's add a new node (value is 99) to the end:




```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Code

```
Element temp = new Element();
temp.data = 99;
tail.next = temp;
temp.prev = tail;
tail = tail.next;
tail.next = null;
```

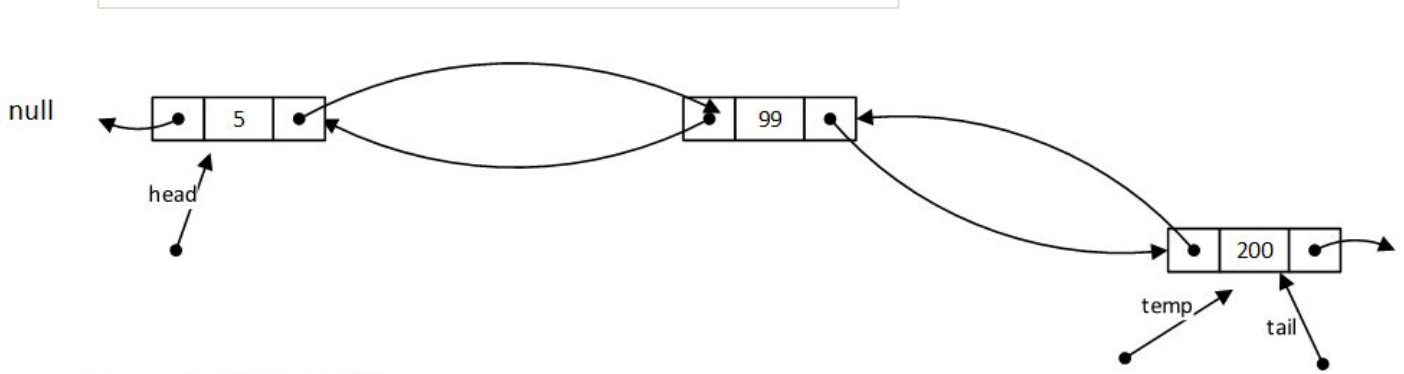
Code School

Dynamical Data Structures

Doubly linked list

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Here is node with value 200 added:



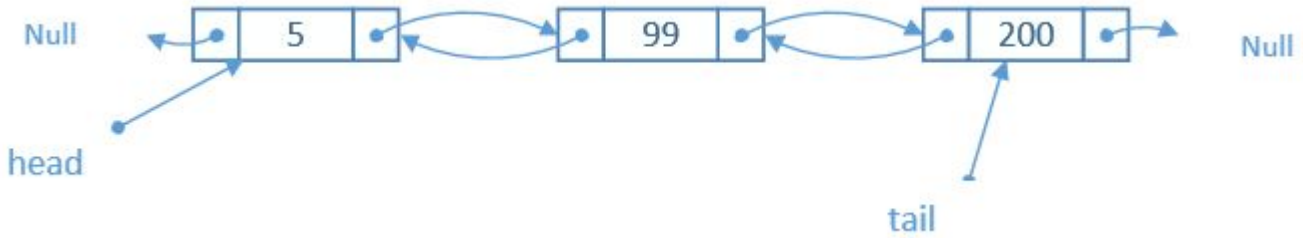

```
temp = new Element();
temp.data = 200;
tail.next = temp;
temp.prev = tail;
tail = tail.next;
tail.next = null;
```

Code School

Dynamical Data Structures

Here is node with value 200 added:

Doubly linked list

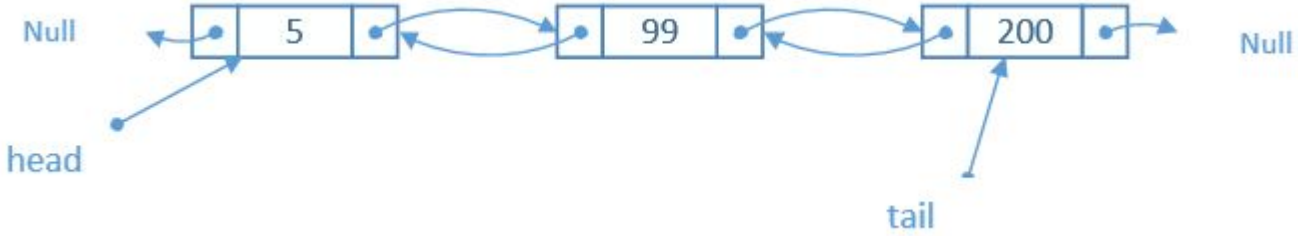



```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
temp = new Element();
temp.data = 200;
tail.next = temp;
temp.prev = tail;
tail = tail.next;
tail.next = null;
```

Code School

Dynamical Data Structures



Doubly linked list

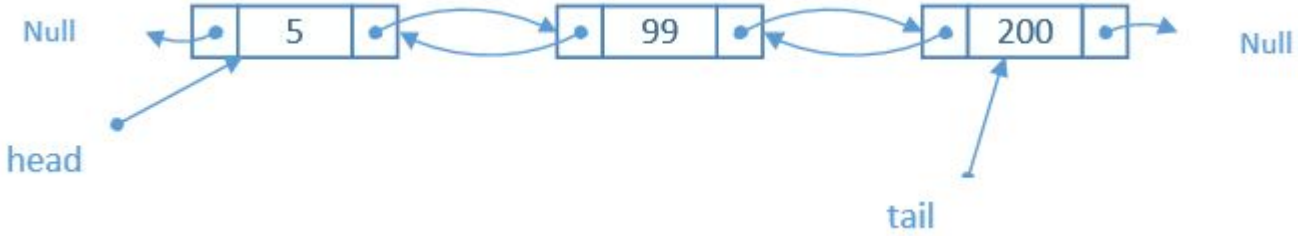
```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Printing values

```
// printing, way 1
Console.WriteLine("{0}", head.data);
Console.WriteLine("{0}", head.next.data);
Console.WriteLine("{0}", head.next.next.data);
```

Code School

Dynamical Data Structures



Doubly linked list

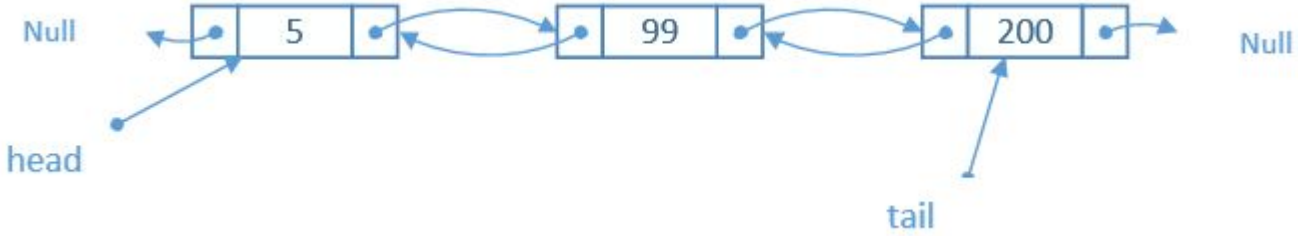
Printing values

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
// printing, way 2
for (Element k = head; k != null; k = k.next )
    Console.WriteLine("{0}", k.data);
```

Code School

Dynamical Data Structures



Doubly linked list

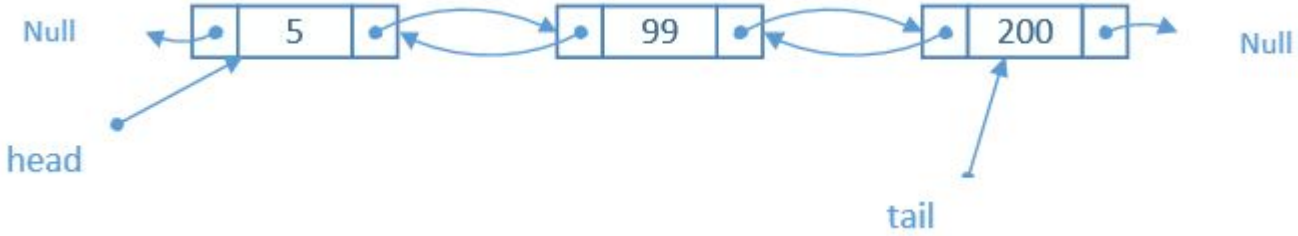
Printing values

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
// printing, reverse order
Console.WriteLine("{0}", tail.data);
Console.WriteLine("{0}", tail.prev.data);
Console.WriteLine("{0}", tail.prev.prev.data);
```

Code School

Dynamical Data Structures



Doubly linked list

Printing values

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
// printing, reverse order
for (Element k = tail; k != null; k = k.prev)
    Console.WriteLine("{0}", k.data);
```

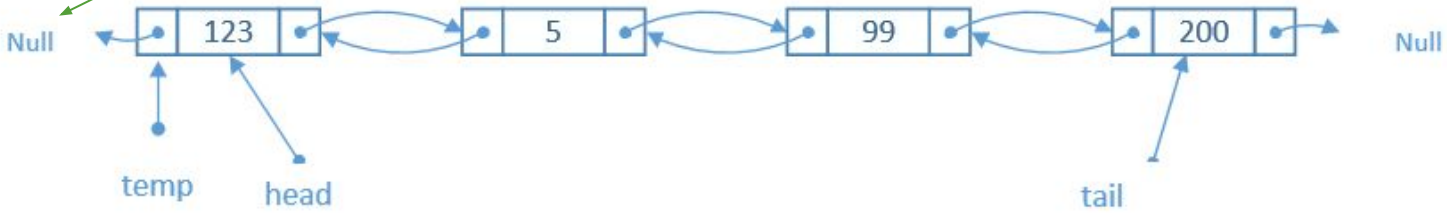
Code School

Dynamical Data Structures

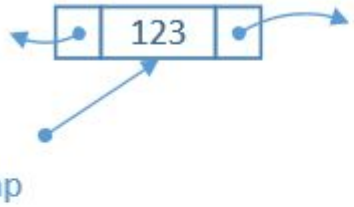
Doubly linked list

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

Remember to move null, too



Add new node to the beginning



Create the new node by using a temporary reference...

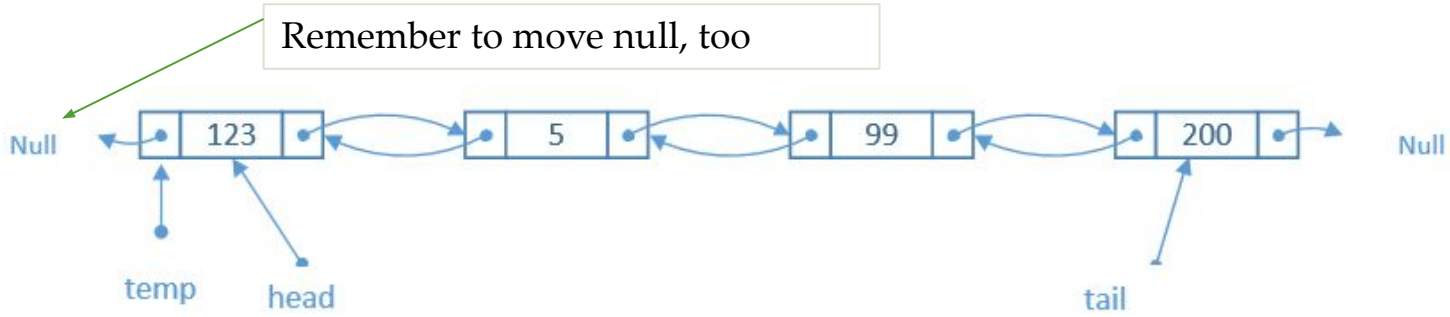
Connect to the beginning...

Move head to point to the new first node

Code School

Dynamical Data Structures

Doubly linked list



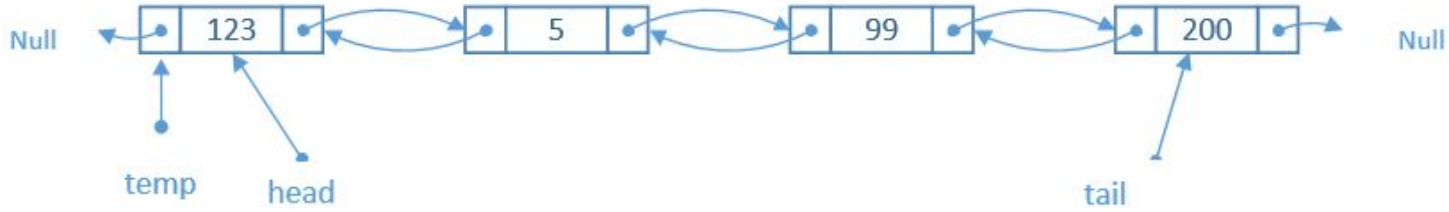

```
Class Element
{
  public int data;
  public Element next;
  public Element prev;
}
```

```
// new to the beginning
temp = new Node();
temp.data = 123;
head.prev = temp;
temp.next = head;
head = temp;
head.prev = null;
```

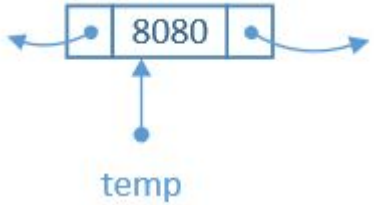
Code School

Dynamical Data Structures

Doubly linked list




```
// add element with 8080 after element 5
      temp = new Node();
      temp.data = 8080;

      for (Node x = head; x != null; x = x.next)
      {
          if (x.data == 5)
          {
              temp.next = x.next;
              x.next.prev = temp;

              x.next = temp;
              temp.prev = x;

              break;
          }
      }
```

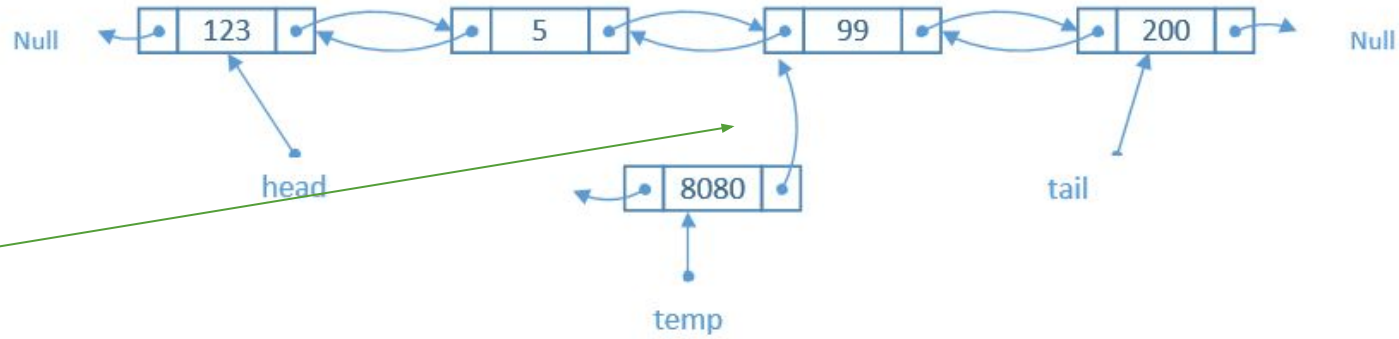


Code School

Dynamical Data Structures

Doubly linked list

```
// add element with 8080 after element 5
      temp = new Node();
      temp.data = 8080;

      for (Node x = head; x != null; x = x.next)
      {
          if (x.data == 5)
          {
              temp.next = x.next;
              x.next.prev = temp;

              x.next = temp;
              temp.prev = x;

              break;
          }
      }
```

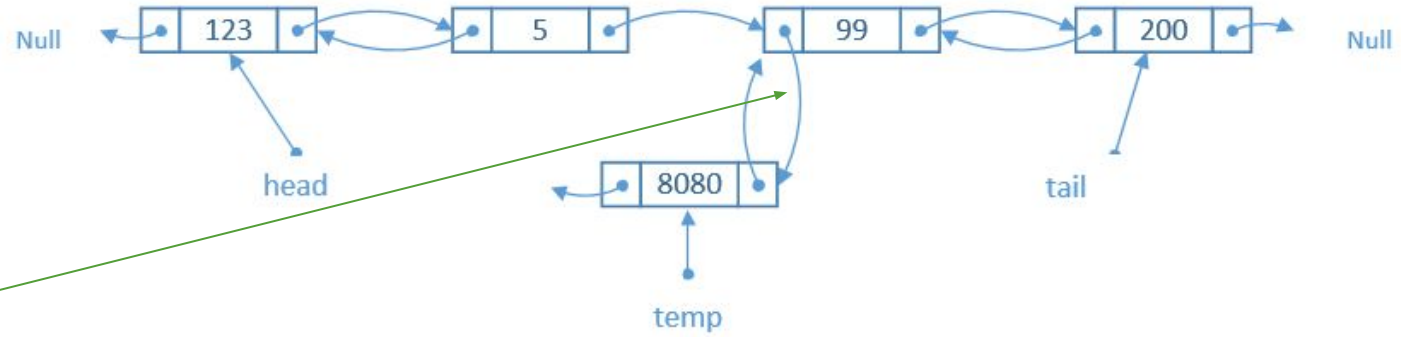


Code School

Dynamical Data Structures

Doubly linked list

```
// add element with 8080 after element 5
      temp = new Node();
      temp.data = 8080;

      for (Node x = head; x != null; x = x.next)
      {
          if (x.data == 5)
          {
              temp.next = x.next;
              x.next.prev = temp;

              x.next = temp;
              temp.prev = x;

              break;
          }
      }
```

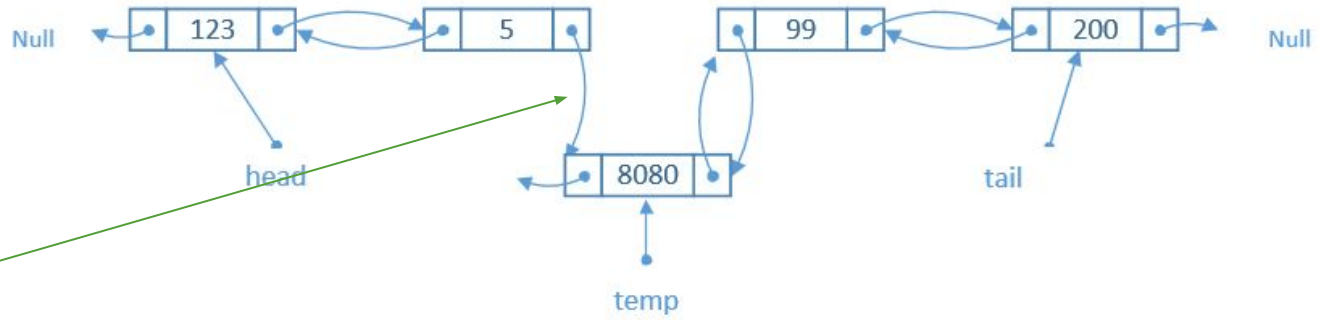


Code School

Dynamical Data Structures

Doubly linked list

```
// add element with 8080 after element 5
      temp = new Node();
      temp.data = 8080;

      for (Node x = head; x != null; x = x.next)
      {
          if (x.data == 5)
          {
              temp.next = x.next;
              x.next.prev = temp;

              x.next = temp;
              temp.prev = x;

              break;
          }
      }
```

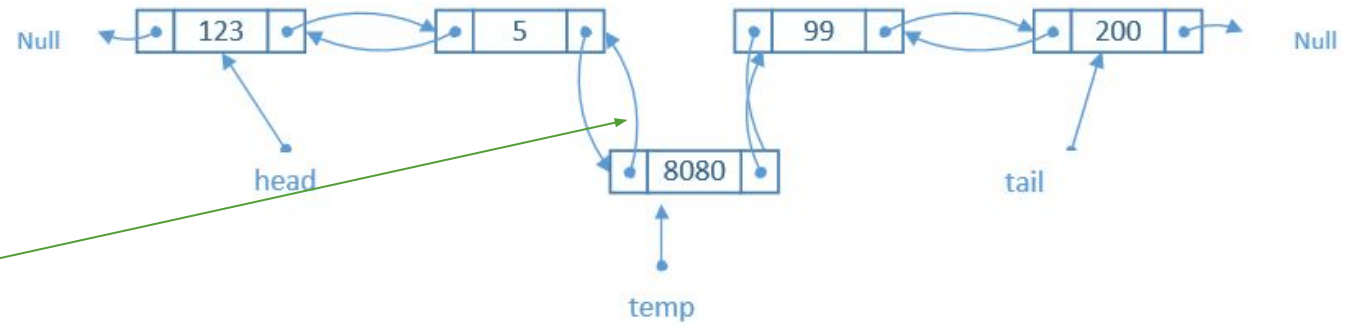


Code School

Dynamical Data Structures

Doubly linked list

```
// add element with 8080 after element 5
      temp = new Node();
      temp.data = 8080;

      for (Node x = head; x != null; x = x.next)
      {
          if (x.data == 5)
          {
              temp.next = x.next;
              x.next.prev = temp;

              x.next = temp;
              temp.prev = x;

              break;
          }
      }
```

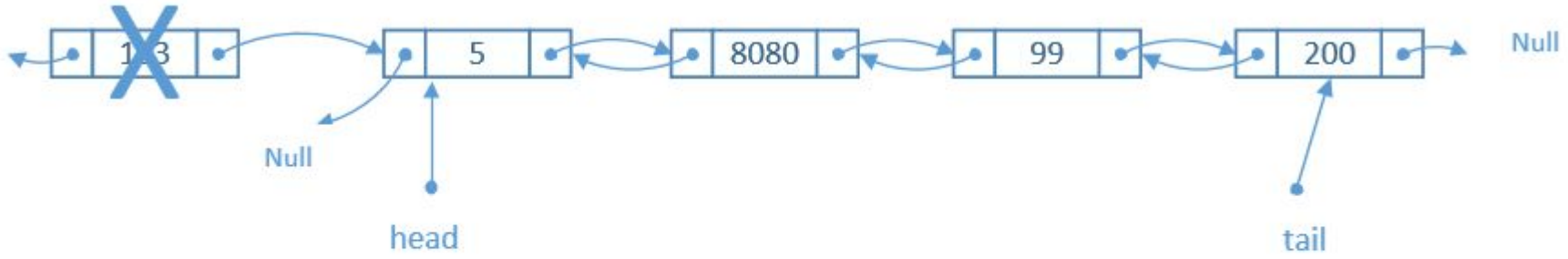


Code School

Dynamical Data Structures

Doubly linked list

Remove 1. node

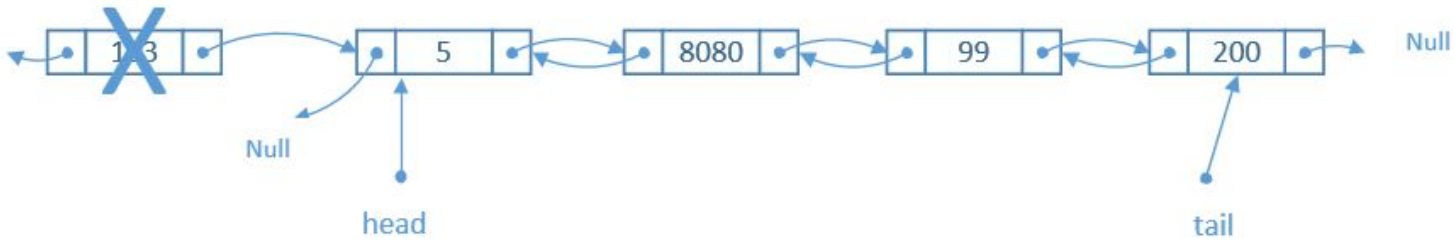


Remove from the beginnng:
move head to point to the second node...
Mov null to point to the new starting node

Code School

Dynamical Data Structures

Doubly linked list



Remove 1. node

Remove from the beginnng:
move head to point to the second node...
Mov null to point to the new starting node

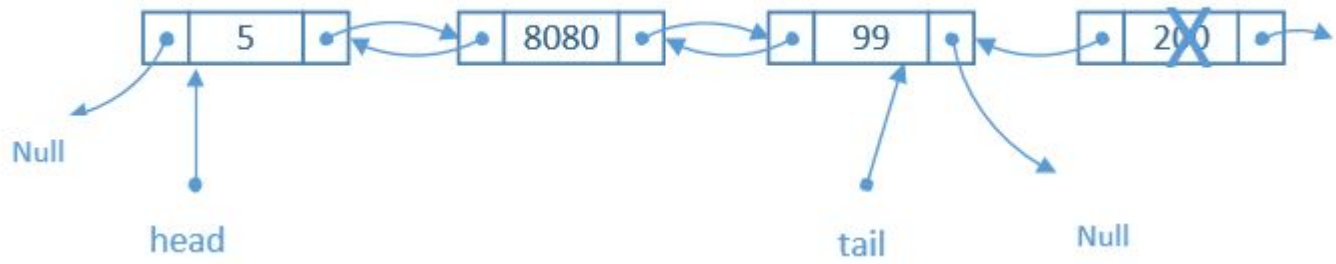
```
// remove from the beginning
head = head.next;
head.prev = null;
```

Code School

Dynamical Data Structures

Doubly linked list

Remove the last node




```
// from the end
 tail = tail.prev;
 tail.next = null;
```

Code School

Dynamical Data Structures

Doubly linked list

Remove from a random position

```
// remove 8080
 for (Node x = head; x != null; x = x.next)
     if (x.data == 8080)
     {
         x.prev.next = x.next;
         x.next.prev = x.prev;
         break;

     }
```

Code School

Dynamical Data Structures

Doubly linked list

Try them!

Hopefully you learned a bit about linked lists!

Create nodes more practical
Add attributes and methods
Study collections that programming enviroment offers: you can use even them as base classes …

Create own stacks and queues..

Code School

Dynamical Data Structures

Binary tree

Heap memory is used.

Can grow and shrink
(basic arrays have fixes size)

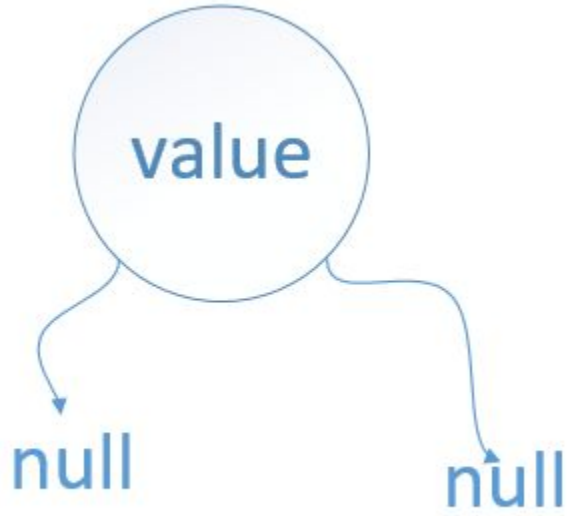
You can add new elements to the beginning of linked list, or to the end
or to the middle.
Element can be removed from the beginning, from the end and from the middle.

Very flexible data structures!

Code School

Dynamical Data Structures

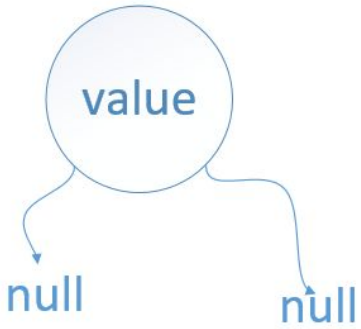
Binary tree



Code School

Dynamical Data Structures

Binary tree



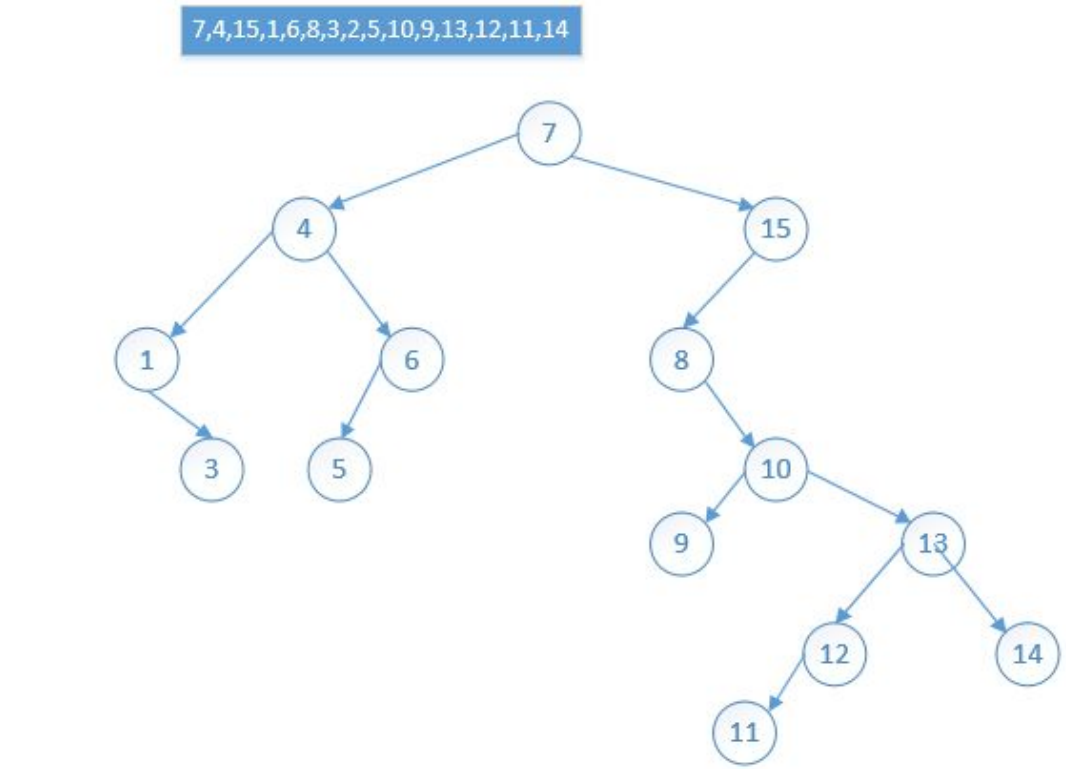

```
class BinaryTree
{
    public int data;
    public BinaryTree leftChild;
    public BinaryTree rightChild;

}
```

Code School

Dynamical Data Structures

Binary tree

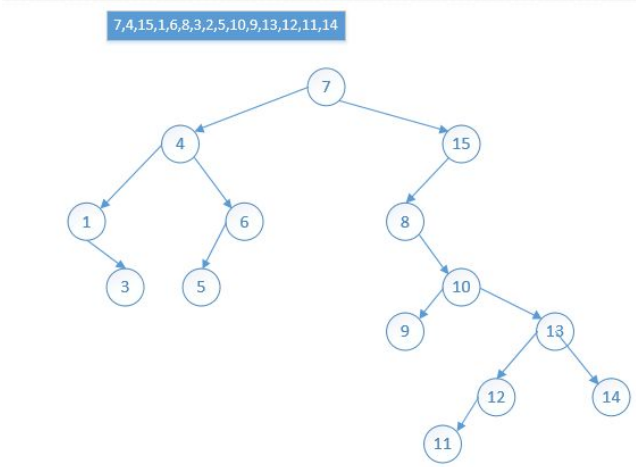
```
class BinaryTree
{
    public int data;
    public BinaryTree leftChild;
    public BinaryTree rightChild;

}
```



Code School

Dynamical Data Structures

Binary tree

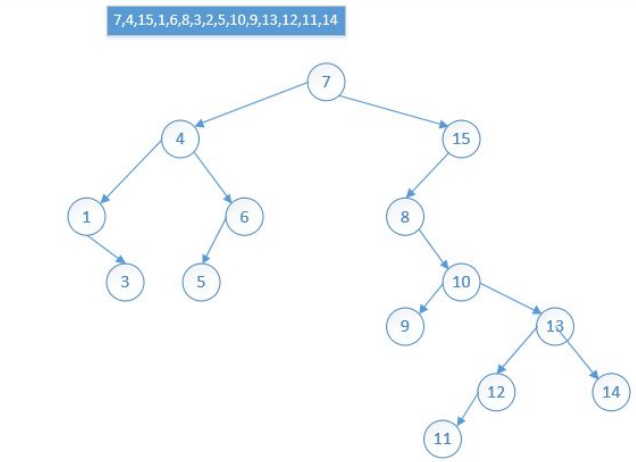



```
public static void add(BinaryTree bt, int val
{
boolean noplace = true;
BinaryTree newNode = new BinaryTree();
newNode.data = value;
newNode.leftChild = null;
newNode.rightChild = null;
while (noplace == true)
{

 if (value < bt.data)
   if (bt.leftChild == null)
     noplace= false;
   else bt = bt.leftChild;

 else
    if (bt.rightChild == null)
      noplace = false;
    else bt = bt.rightChild;
}

if (value < bt.data)
  bt.leftChild = newNode;
else
 bt.rightChild= newNode;

}
```

Code School

Dynamical Data Structures

Binary tree



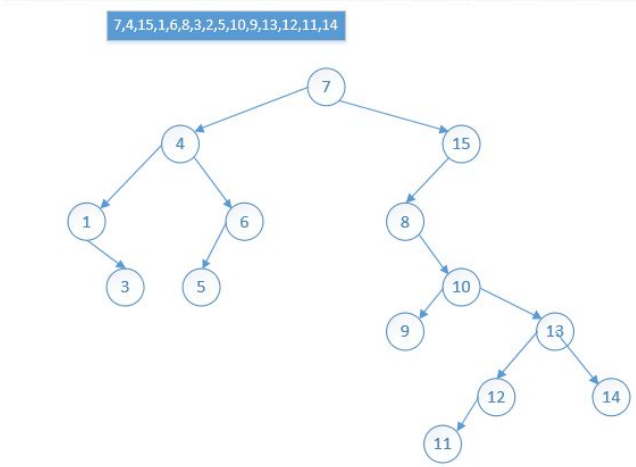

```
int[] values = {7,4,15,1,6,8,3,2,5,10,9,13,12,11,14};

BinaryTree bt = new BinaryTree();
bt.data = values[0];
bt.leftChild = null; bt.rightChild = null;
BinaryTree tree = bt;

for (int k = 1; k < values.length; k++)
{
 add(tree, values[k]);
}
```

Code School

Dynamical Data Structures

orders

Binary tree



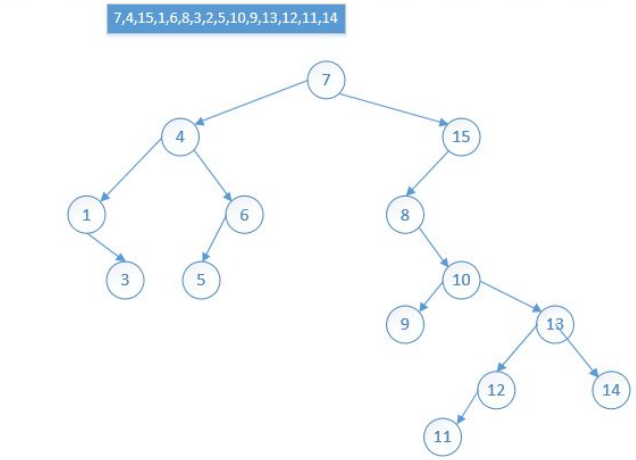

```
public static void inorderPrint(BinaryTree bt)
{
    if (bt == null)
        return;
    inorderPrint(bt.leftChild);
    System.out.print("  " + bt.data);
    inorderPrint(bt.rightChild);
}
public static void preorderPrint(BinaryTree bt)
{
    if (bt == null)
        return;
    System.out.print("  " + bt.data);
    preorderPrint(bt.leftChild);
    preorderPrint(bt.rightChild);
}
public static void postorderPrint(BinaryTree bt)
{
    if (bt == null)
        return;
    postorderPrint(bt.leftChild);
    postorderPrint(bt.rightChild);
    System.out.print("  " + bt.data);
}
```

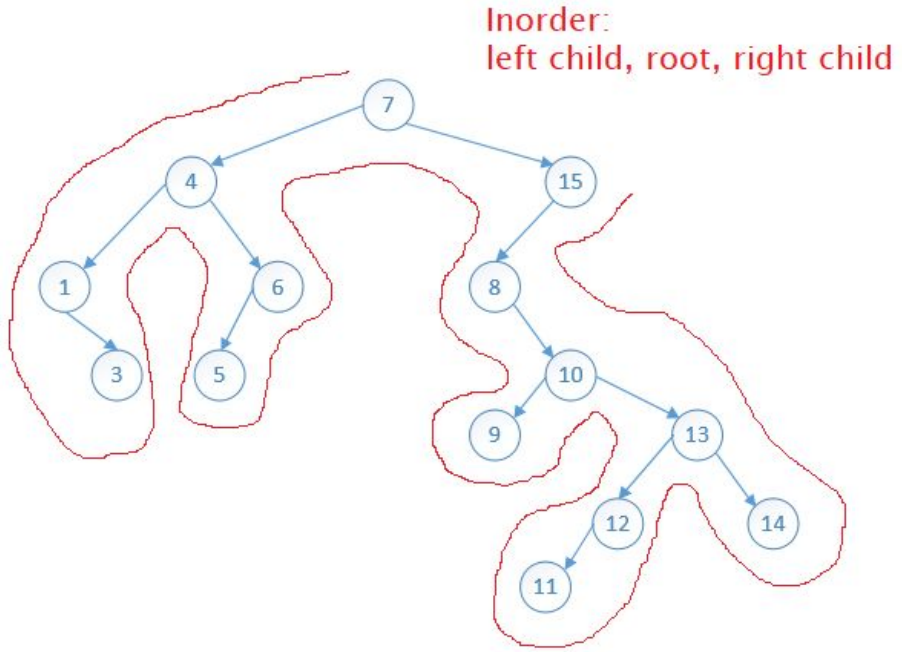
Code School

Dynamical Data Structures

orders

Binary tree



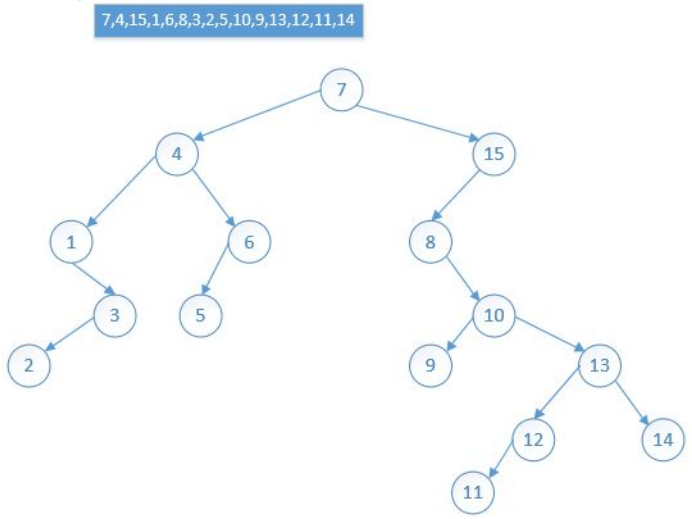


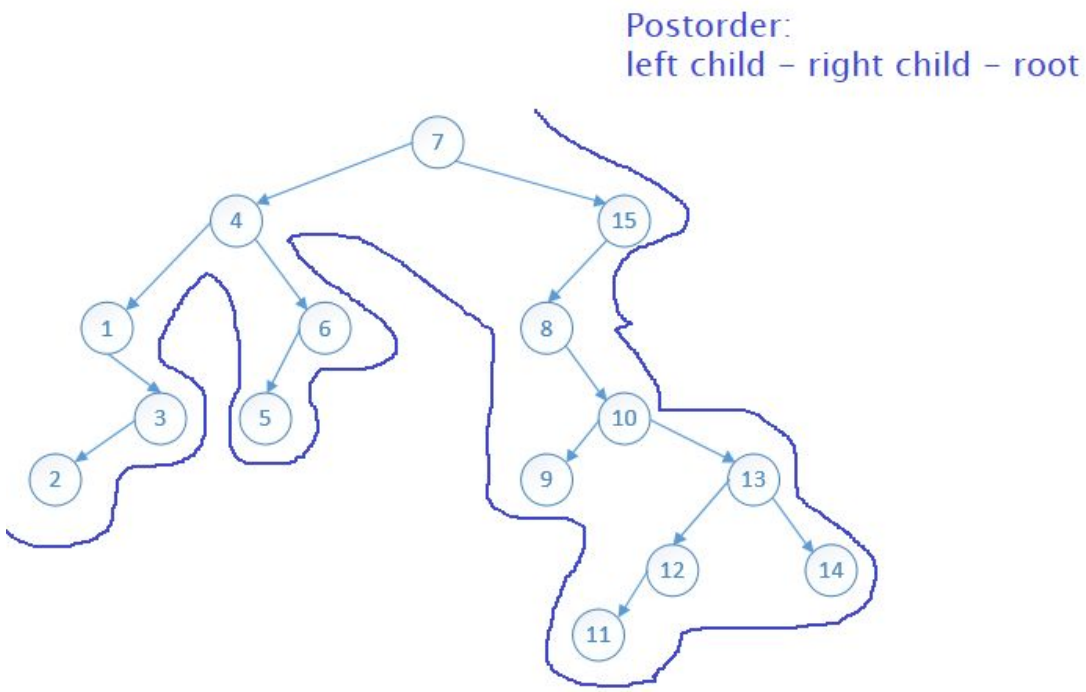
Code School

Dynamical Data Structures

orders

Binary tree



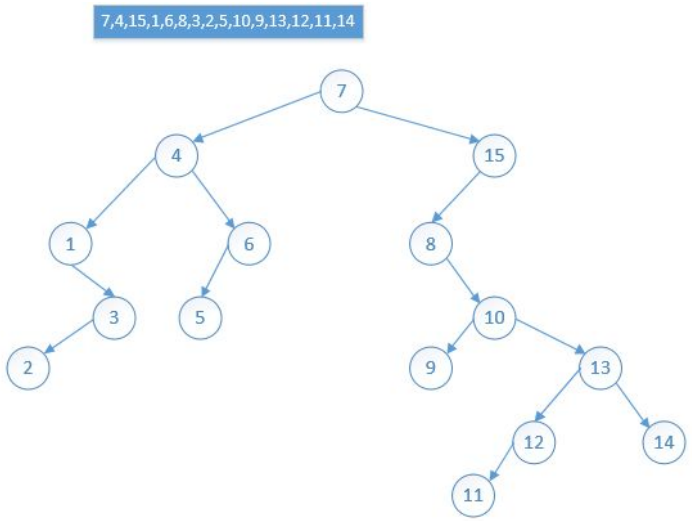


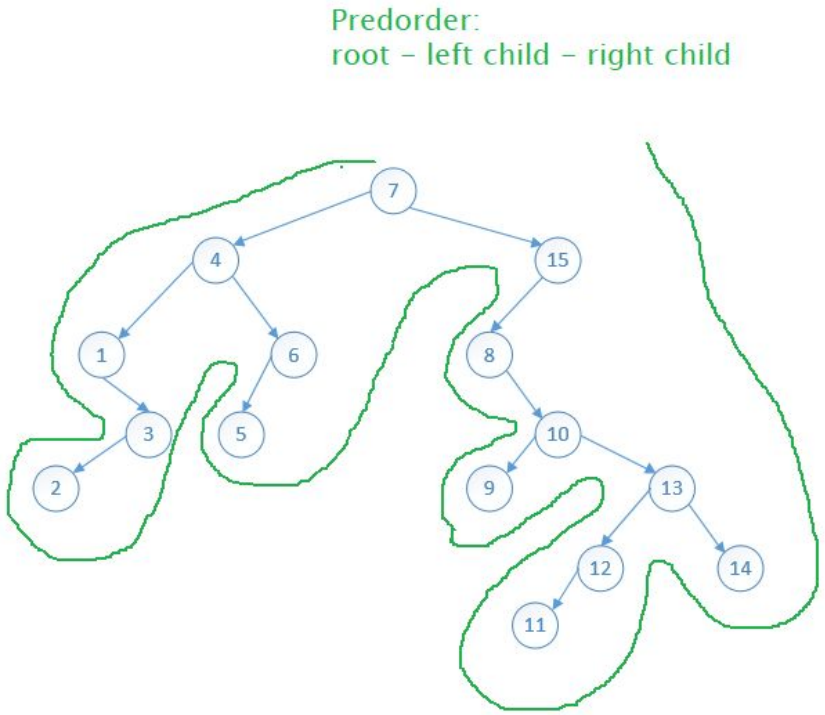
Code School

Dynamical Data Structures

orders

Binary tree

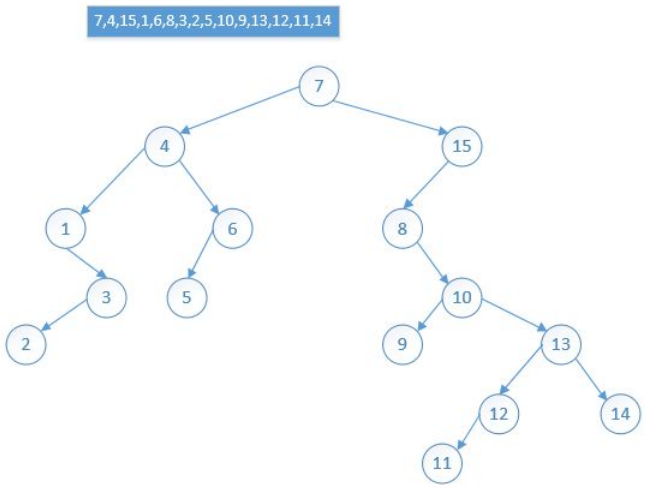




Code School

Dynamical Data Structures

Binary tree



orders

```
INORDER
  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
POSTORDER
  2  3  1  5  6  4  9  11  12  14  13  10  8  15  7
PREORDER
  7  4  1  3  2  6  5  15  8  10  9  13  12  11  14
```

Code School

Try them!

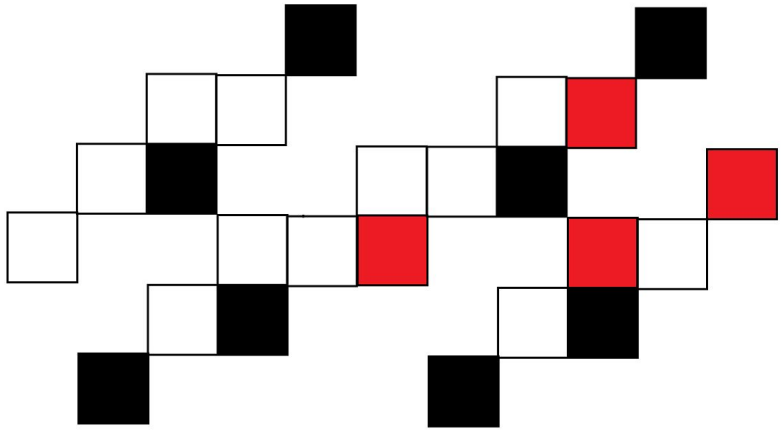
Where are binary trees used?

Order makes them fast in searching…

Add once – read many

Code School

Edsger Dijkstra
shortest routes demo

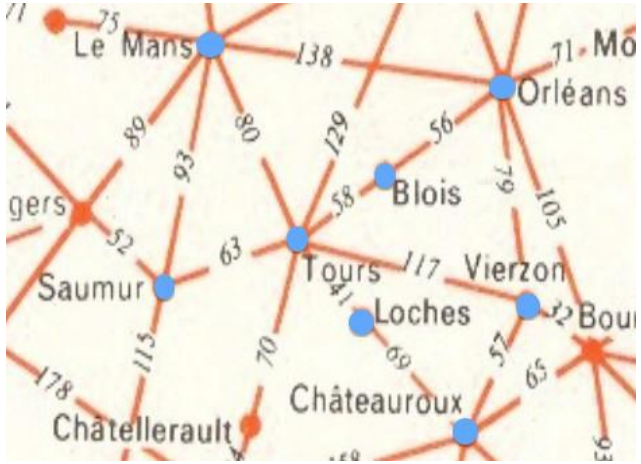


Code School

Edsger Dijkstra
shortest routes demo

Dijkstra Example
Shortest routes from Le Mans to other cities!

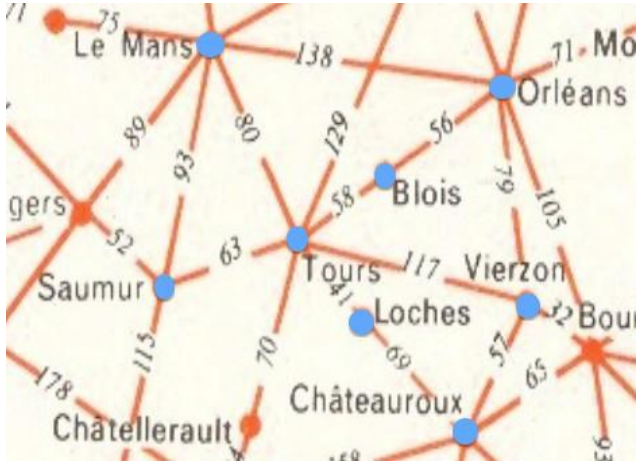
Map of France is here:



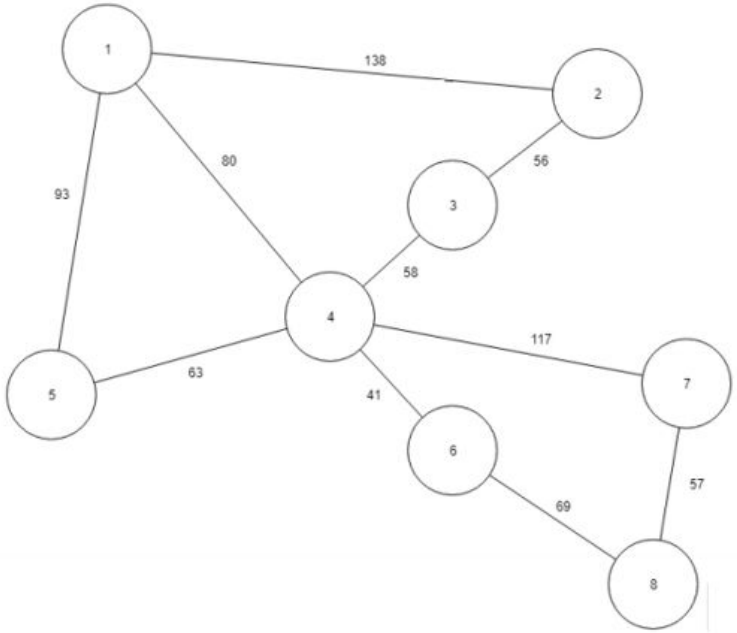
Blue circles are cities. We start from Le Mans.

Code School

Edsger Dijkstra
shortest routes demo

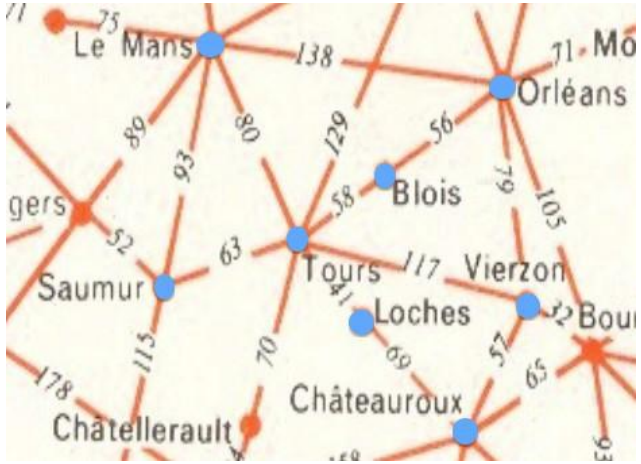


Here the network/graph as a diagram:



Code School

Edsger Dijkstra
shortest routes demo



Here the network/graph as a matrix:

	1	2	3	4	5	6	7	8
1	0	138	INF	80	93	INF	INF	INF
2	138	0	56	INF	INF	INF	79	INF
3	INF	56	0	58	INF	INF	INF	INF
4	80	INF	58	0	63	41	117	INF
5	93	INF	INF	63	0	INF	INF	INF
6	INF	INF	INF	41	INF	0	INF	69
7	INF	79	INF	117	INF	INF	0	57
8	INF	INF	INF	INF	INF	69	57	0

Code School

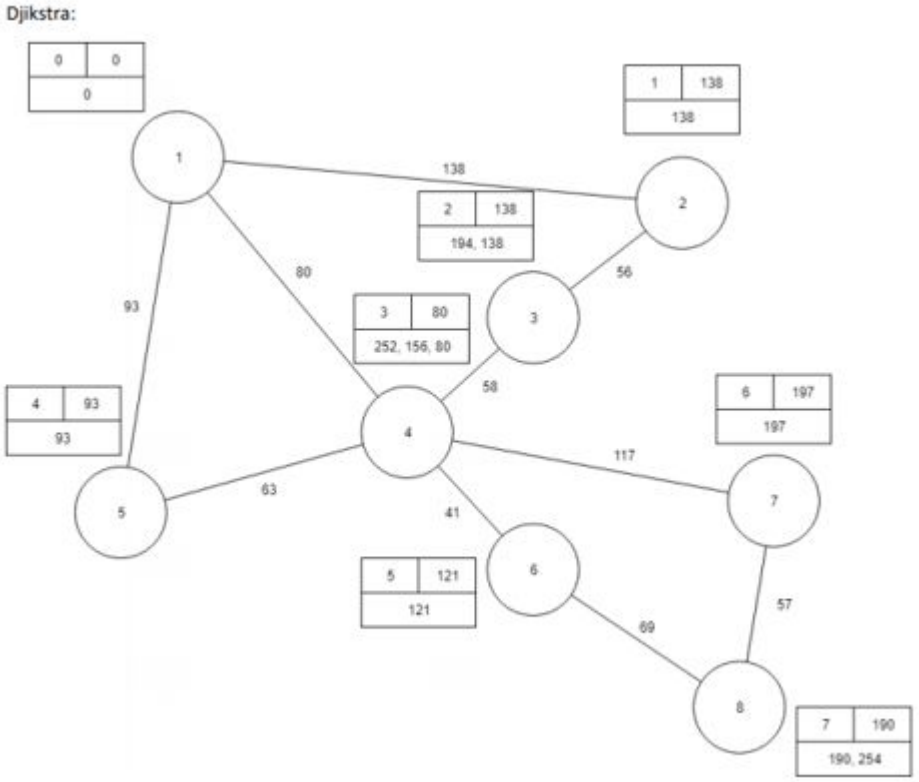
Edsger Dijkstra
shortest routes demo

Here is the solution matrix:
note priority queue

Round nr	Current node	Neighbours	Updates	Queue (priority queue)
1	1	2,4,5	2(true, 138, 1), 4(true, 80, 1),5(true,93,1)	4(true, 80, 1), 5(true, 93, 1), 2(true, 138, 1)
2	4 80	3,5,6,7	3(true,58,4) => 138 5(true,63,4) => 143 NO 6(true,41,4) => 121 7(true,117,4) =>197	5(true, 93, 1) 6(true,121,4) 2(true, 138, 1) 3(true,138,4) 7(true,197,4)
3	5 93	1, 4	1 NO 4 NO	6(true,121,4) 2(true, 138, 1) 3(true,138,4) 7(true,197,4)
4	6 121	4,8	4 NO 8(true, 121 + 69, 6)	2(true, 138, 1) 3(true,138,4) 8(true, 190, 6) 7(true,197,4)
5	2 138	1,3	1 NO 3(true, 138+56, true) NO	3(true,138,4) 8(true, 190, 6) 7(true,197,4)
6	3 138	2,4	2 NO 3 NO	8(true, 190, 6) 7(true,197,4)
7	8 190	6,7	7 NO 7 NO	7(true,197,4)
8	7 197	4,8	4 NO 8 NO	

Code School

Edsger Dijkstra
shortest routes demo

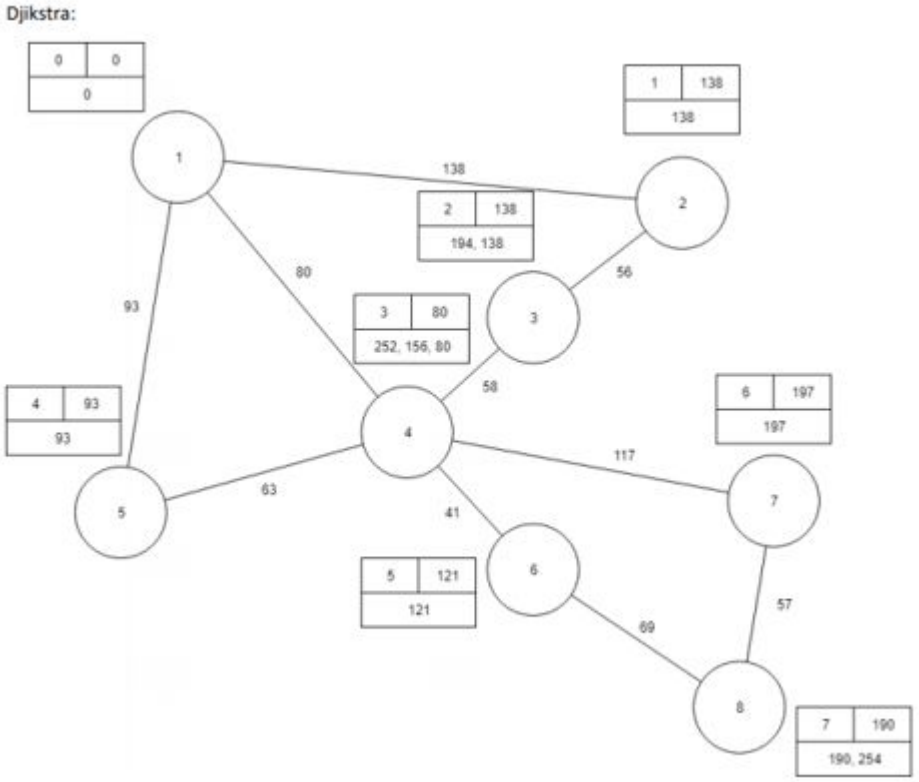


Code School

Edsger Dijkstra
shortest routes demo

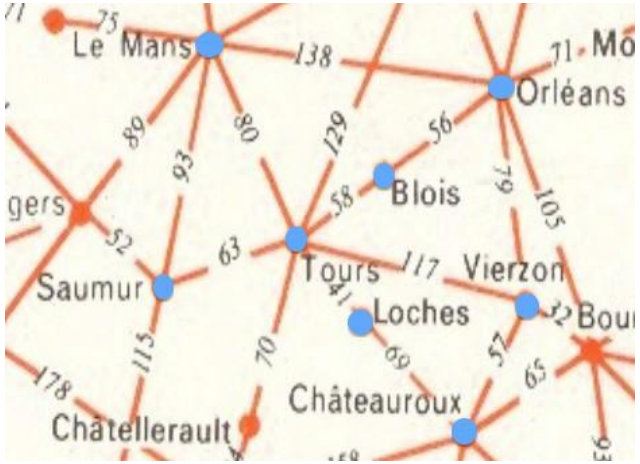
Results of the code:

```
0..0 -> 0
0..1.. -> 138
0..3..2.. -> 138
0..3.. -> 80
0..4.. -> 93
0..3..5.. -> 121
0..3..6.. -> 197
0..3..5..7.. -> 190
```



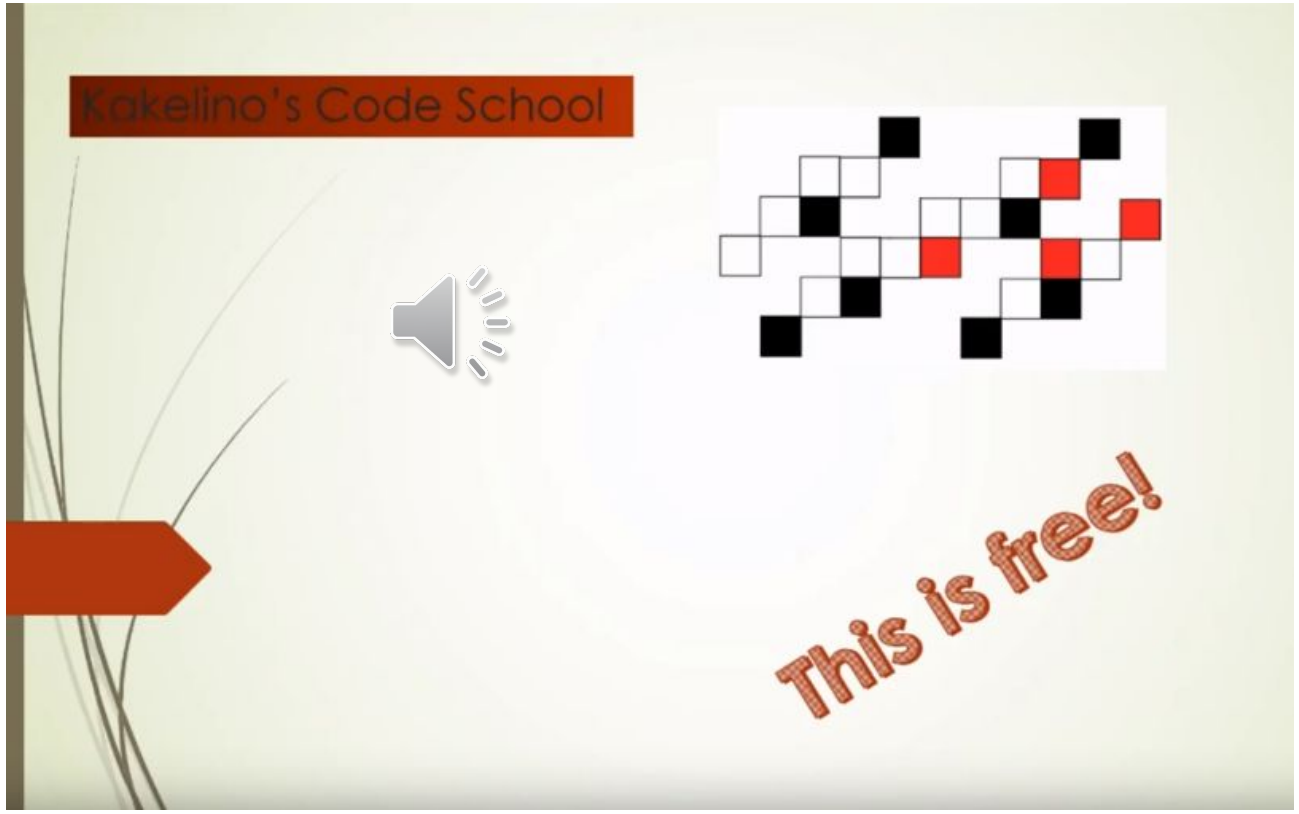
Code School

Edsger Dijkstra
shortest routes demo



Try to simulate it!

Code School



Code School

Bin packing

First fit method

Code School

Bin packing

First fit method

Here are passenger groups
11 groups

Code School

Bin packing

First fit method

First group has
8 persons:
put persons to bus 1

BUS 1

Code School

Bin packing

First fit method

Second group has
7 persons:
put persons to bus 1, too

BUS 1

Code School

Bin packing

First fit method

3. group has
14 persons:
put persons to bus 2

BUS 2

Code School

Bin packing

First fit method

4. group has
9 persons:
put persons to bus 3

BUS 3

Code School

Bin packing

First fit method

5. group has
6 persons:
put persons to bus 2, too

BUS 2

Code School

Bin packing

First fit method

6. group has
9 persons:
put persons to bus 3, too

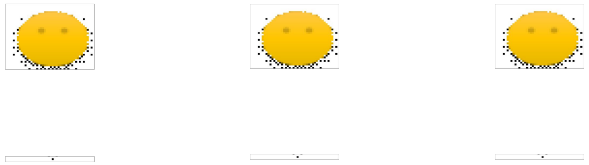
BUS 3

Code School

Bin packing

First fit method

7. group has
5 persons:
put persons to bus 1, too



BUS 1

Code School

Bin packing

First fit method

8. group has
15 persons:
put persons to bus 4

BUS 4

Code School

Bin packing

First fit method

9. group has
6 persons:
put persons to bus 5

BUS 5

Code School

Bin packing

First fit method

10. group has
7 persons:
put persons to bus 5, too

BUS 5

Code School

Bin packing

First fit method

11. group has
8 persons:
put persons to bus 6

BUS 6

That’s all folks!!

This is a free version, part 2, of algorithms…

Final, completed version, real eBook, is coming soon!