CSE 250: ADTs, Sequences, and Arrays

Lecture 8

Sept 13, 2023

Reminders

- PA1 Tests due Sun, Sept 15 at 11:59 PM
  - Recitations will cover writing good test cases.
- PA1 Implementation due Sun, Sept 22 at 11:59 PM
  - Implement a Sorted Linked List

Bubblesort

4 10 9 18 20 6 3 14 13 7

Bubblesort

4	10	9	18	20	6	3	14	13	7

Bubblesort

4	10	9	18	20	6	3	14	7	13

Bubblesort

Bubblesort

Bubblesort

4	10	9	18	20	6	3	7	14	13

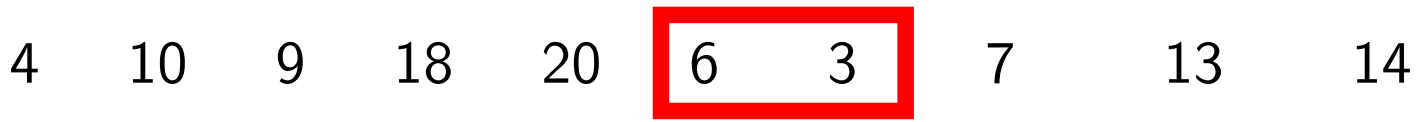
Bubblesort

Bubblesort

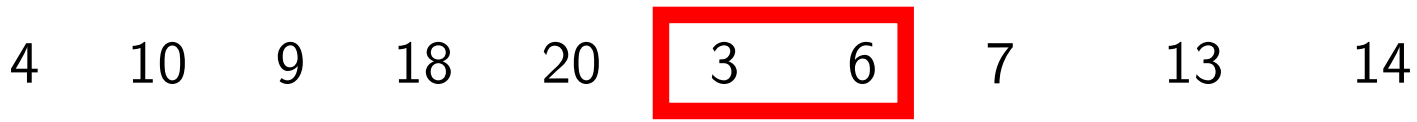
Bubblesort

Bubblesort

Bubblesort



Bubblesort



Bubblesort

Bubblesort

Bubblesort

4	10	9	18	20	3	6	7	13	14

Bubblesort

```
1   public void bubblesort(int[] data)
2   {
3     int N = data.length;
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data[j+1] < data[j])
9         {
10          swap data[j] and data[j+1]
11        }
12      }
13    }
14  }
```

Bubblesort

```
1            if(data[j+1] < data[j])
2            {
3              swap data[j] and data[j+1]
4            }
```

Bubblesort

```
1            if(data[j+1] < data[j])
2            {
3              swap data[j] and data[j+1]
4            }
```

$\theta(1)$

Bubblesort

```
1   public void bubblesort(int[] data)
2   {
3     int N = data.length;
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         θ(1)
9       }
10    }
11  }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        ∑_{j=i}^{N-1} θ(1)
7      }
8    }
```

Wait, what...?

$\theta(1)$ is a set of functions (or complexity class).

$\sum_{j=i}^{N-1} \theta(1) = \sum_{j=i}^{N-1} f(N)$ where we know $f(N) \in \theta(1)$

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        ∑_{j=i}^{N-1} θ(1)
7      }
8    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        (N − 1 + 1 − i) · θ(1)
7      }
8    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        (N − i) · θ(1)
7      }
8    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      ∑_{i=0}^{N-2} (N − i) · θ(1)
5    }
```

Bubblesort

```
1   public void bubblesort(int[] data)
2   {
3     int N = data.length;
4     $\left(\sum_{i=0}^{N-2} N \cdot \theta(1)\right) - \left(\sum_{i=0}^{N-2} i \cdot \theta(1)\right)$
5   }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      ((N − 2 + 1 − 0) · N · θ(1)) − (∑_{i=0}^{N−2} i · θ(1))
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      (N^2 − N) · θ(1) − (∑_{i=0}^{N−2} i · θ(1))
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      (N^2 − N) · θ(1) − (0 + ((N−2)(N−2+1))/2 · θ(1))
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      (N^2 - N) · θ(1) - ( (N^2 - 3N + 2)/2 · θ(1) )
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      (1/2 N^2 + N - 2) · θ(1)
5    }
```

Algebra with θ

Let $f'(N) = c \cdot f(N)$ where $f(N) \in \theta(g(N))$
(c is a constant)

Algebra with θ

Let $f'(N) = c \cdot f(N)$ where $f(N) \in \theta(g(N))$
(c is a constant)

What complexity class is $f'(N)$ in?

Algebra with θ

Let $f'(N) = c \cdot f(N)$ where $f(N) \in \theta(g(N))$
(c is a constant)

What complexity class is $f'(N)$ in?

So $c \cdot \theta(g(N)) = \theta(g(N))$

Algebra with θ

$$c \cdot \theta(f(N)) \quad =$$

Algebra with θ

$$c \cdot \theta(f(N)) \quad = \quad \theta(f(N))$$

Algebra with θ

$$c \cdot \theta(f(N)) = \theta(f(N))$$

$$N \cdot \theta(f(N)) =$$

Algebra with θ

$$c \cdot \theta(f(N)) = \theta(f(N))$$

$$N \cdot \theta(f(N)) = \theta(N \cdot f(N))$$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &=
\end{aligned}
$$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists})
\end{aligned}
$$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists}) \\
\theta(g(N)) + \theta(f(N)) &=
\end{aligned}
$$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists}) \\
\theta(g(N)) + \theta(f(N)) &= \theta(g(N) + f(N))
\end{aligned}
$$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists}) \\
\theta(g(N)) + \theta(f(N)) &= \theta(g(N) + f(N)) \\
&= \text{The greater of } \theta(f(N)) \textbf{ or } \theta(g(N))
\end{aligned}
$$

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      (1/2 N^2 + N - 2) · θ(1)
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      θ((1/2 N^2 + N − 2) · 1)
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      θ(½N² + N − 2)
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      θ(½N²)
5    }
```

Bubblesort

```
1     public void bubblesort(int[] data)
2     {
3       int N = data.length;
4       θ(N^2)
5     }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      θ(1)
4      θ(N^2)
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      θ(N^2)
4    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      θ(N^2)
4    }
```

Bubblesort on an array is $\theta(N^2)$

Rules of Thumb

- **Lines of Code**: Add Complexities
- **Loops**: Multiply Complexities
- **If/Then**: Cases block '{'

Bubblesort

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          int temp = data.get(j);
11          data.set(j, data.get(j+1));
12          data.set(j+1, temp);
13        }
14      }
15    }
16  }
```

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$ (this implies $O(1)$)

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$ (this implies $O(1)$)

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = \begin{cases} O(1) & \textbf{if } \texttt{data} \text{ is a } \texttt{ArrayList} \\ O(N) & \textbf{if } \texttt{data} \text{ is a } \texttt{LinkedList} \end{cases}$$

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$ (this implies $O(1)$)

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = \begin{cases} O(1) & \textbf{if } \texttt{data} \text{ is a } \texttt{ArrayList} \\ O(N) & \textbf{if } \texttt{data} \text{ is a } \texttt{LinkedList} \end{cases}$$

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = ???$$

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$ (this implies $O(1)$)

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = \begin{cases} O(1) & \textbf{if } \texttt{data} \text{ is a } \texttt{ArrayList} \\ O(N) & \textbf{if } \texttt{data} \text{ is a } \texttt{LinkedList} \end{cases}$$

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = O(N)$$

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Algebra with O

$$c \cdot O(f(N)) \quad =$$

Algebra with O

$$c \cdot O(f(N)) \quad = \quad O(f(N))$$

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) =$$

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

$$g(N) \cdot O(f(N)) =$$

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

$$g(N) \cdot O(f(N)) = O(g(N) \cdot f(N))$$

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

$$g(N) \cdot O(f(N)) = O(g(N) \cdot f(N))$$

$$O(g(N)) + O(f(N)) =$$

Algebra with O

$$
\begin{aligned}
c \cdot O(f(N)) &= O(f(N)) \\
N \cdot O(f(N)) &= O(N \cdot f(N)) \\
g(N) \cdot O(f(N)) &= O(g(N) \cdot f(N)) \\
O(g(N)) + O(f(N)) &= O(g(N) + f(N))
\end{aligned}
$$

Algebra with O

$$
\begin{aligned}
c \cdot O(f(N)) &= O(f(N)) \\
N \cdot O(f(N)) &= O(N \cdot f(N)) \\
g(N) \cdot O(f(N)) &= O(g(N) \cdot f(N)) \\
O(g(N)) + O(f(N)) &= O(g(N) + f(N)) \\
&= \text{the greater of } O(f(N)) \textbf{ or } O(g(N))
\end{aligned}
$$

$$
\begin{cases} O(g(N)) & \textbf{if one thing} \\ O(f(N)) & \textbf{otherwise} \end{cases} =
$$

Algebra with O

$$
\begin{aligned}
c \cdot O(f(N)) &= O(f(N)) \\
N \cdot O(f(N)) &= O(N \cdot f(N)) \\
g(N) \cdot O(f(N)) &= O(g(N) \cdot f(N)) \\
O(g(N)) + O(f(N)) &= O(g(N) + f(N)) \\
&= \text{the greater of } O(f(N)) \textbf{ or } O(g(N))
\end{aligned}
$$

$$
\begin{cases} O(g(N)) & \textbf{if one thing} \\ O(f(N)) & \textbf{otherwise} \end{cases} = \text{the greater of } O(f(N)) \textbf{ or } O(g(N))
$$

Algebra with O

$$
\begin{aligned}
c \cdot O(f(N)) &= O(f(N)) \\
N \cdot O(f(N)) &= O(N \cdot f(N)) \\
g(N) \cdot O(f(N)) &= O(g(N) \cdot f(N)) \\
O(g(N)) + O(f(N)) &= O(g(N) + f(N)) \\
&= \text{the greater of } O(f(N)) \textbf{ or } O(g(N)) \\
\begin{cases} O(g(N)) & \textbf{if one thing} \\ O(f(N)) & \textbf{otherwise} \end{cases} &= \text{the greater of } O(f(N)) \textbf{ or } O(g(N)) \\
&= O(g(N) + f(N))
\end{aligned}
$$

Algebra with Ω

$$c \cdot \Omega(f(N)) = \Omega(f(N))$$

$$N \cdot \Omega(f(N)) = \Omega(N \cdot f(N))$$

$$g(N) \cdot \Omega(f(N)) = \Omega(g(N) \cdot f(N))$$

$$\Omega(g(N)) + \Omega(f(N)) = \Omega(g(N) + f(N))$$

$$= \text{the greater of } \Omega(f(N)) \textbf{ or } \Omega(g(N))$$

$$\begin{cases} \Omega(g(N)) & \textbf{if one thing} \\ \Omega(f(N)) & \textbf{otherwise} \end{cases} = \text{the lesser of } \Omega(f(N)) \textbf{ or } \Omega(g(N))$$

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          int temp = data.get(j);
11          data.set(j, data.get(j+1));
12          data.set(j+1, temp);
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11          O(N + N)
12          O(N)
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11        }
12      }
13    }
14  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11        } else {
12          O(1)
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         O(2N) + O(N OR 1)
9       }
10    }
11  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         O(2N) + O(N + 1)
9       }
10    }
11  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         O(2N) + O(N)
9       }
10    }
11  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         O(N)
9       }
10    }
11  }
```

Bubblesort on Lists

```
1  public void bubblesort(List[Integer] data)
2  {
3    int N = data.size();
4    for(int i = N - 2; i >= 0; i--)
5    {
6      ∑_{j=i}^{N-1} O(N)
7    }
8  }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      for(int i = N - 2; i >= 0; i--)
5      {
6        (N − 1 + 1 − i)O(N)
7      }
8    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      for(int i = N - 2; i >= 0; i--)
5      {
6        (N - i)O(N)
7      }
8    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      ∑_{i=0}^{N-2} (N − i) · O(N)
5    }
```

Bubblesort on Lists

```
1  public void bubblesort(List[Integer] data)
2  {
3    int N = data.size();
4    (∑_{i=0}^{N-2} N · O(N)) − (∑_{i=0}^{N-2} i · O(N))
5  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     ((N − 2 + 1 − 0) · N · O(N)) − (∑_{i=0}^{N−2} i · O(N))
5   }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     (N^2 − N) · O(N) − O(N) · (∑_{i=0}^{N−2} i)
5   }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      (N^2 − N) · O(N) − O(N) · ( ((N−2)(N−2+1)) / 2 )
5    }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     (N^2 − N) · O(N) − O(N) · ( (N^2 − 3N + 2) / 2 )
5   }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      (½N² + ½N − 1) · O(N)
5    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      O(N^3)
5    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(1)
4       O(N^3)
5    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      O(N^3)
4    }
```

Bubblesort on Lists

Can we do better?

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int[] array = data.toArray()
4       bubblesort(array) // Use the array implementation
5       data.clear()
6       data.addAll(Arrays.toList(array))
7     }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       O(N)
4       bubblesort(array) // Use the array implementation
5       data.clear()
6       data.addAll(Arrays.toList(array))
7     }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     O(N)
4     O(N^2)
5     data.clear()
6     data.addAll(Arrays.toList(array))
7   }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     O(N)
4     O(N^2)
5     O(N)
6     data.addAll(Arrays.toList(array))
7   }
```

Bubblesort on Lists

```
1  public void bubblesort(List[Integer] data)
2  {
3    O(N)
4    O(N^2)
5    O(N)
6    O(N)
7  }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(N + N^2 + N + N)
4    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      O(N^2)
4    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(N^2)
4    }
```

Organizing data first can make code faster

Sequences

- 1, 2, 3, 4, 5

Sequences

- 1, 2, 3, 4, 5
- 4, 7, 2, 13, 8, 12, 14, 20, 12, 1

Sequences

- 1, 2, 3, 4, 5
- 4, 7, 2, 13, 8, 12, 14, 20, 12, 1
- 17, 14, 20, 14, 17, 12, 8, 9, 8, 6

Sequences

- 1, 2, 3, 4, 5
- 4, 7, 2, 13, 8, 12, 14, 20, 12, 1
- 17, 14, 20, 14, 17, 12, 8, 9, 8, 6
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Sequences

- 1, 2, 3, 4, 5
- 4, 7, 2, 13, 8, 12, 14, 20, 12, 1
- 17, 14, 20, 14, 17, 12, 8, 9, 8, 6
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . (Fibbonacci Sequence)

Sequences

- 1, 2, 3, 4, 5
- 4, 7, 2, 13, 8, 12, 14, 20, 12, 1
- 17, 14, 20, 14, 17, 12, 8, 9, 8, 6
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . (Fibbonacci Sequence)
- 'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd'
- The lines of a file

What makes a sequence

What is common to these?

- A collection of elements of some type (e.g., numbers, characters, strings).
  - Two elements can have the same value.

What makes a sequence

What is common to these?

- A collection of elements of some type (e.g., numbers, characters, strings).
  - Two elements can have the same value.
- Each element has a unique position.
  - No two elements have the same position

What makes a sequence

What is common to these?

- A collection of elements of some type (e.g., numbers, characters, strings).
  - Two elements can have the same value.
- Each element has a unique position.
  - No two elements have the same position
- The positions of all elements are contiguous.
  - For an N element sequence, every position from $[0, N)$ is occupied.

What can we do with a sequence?

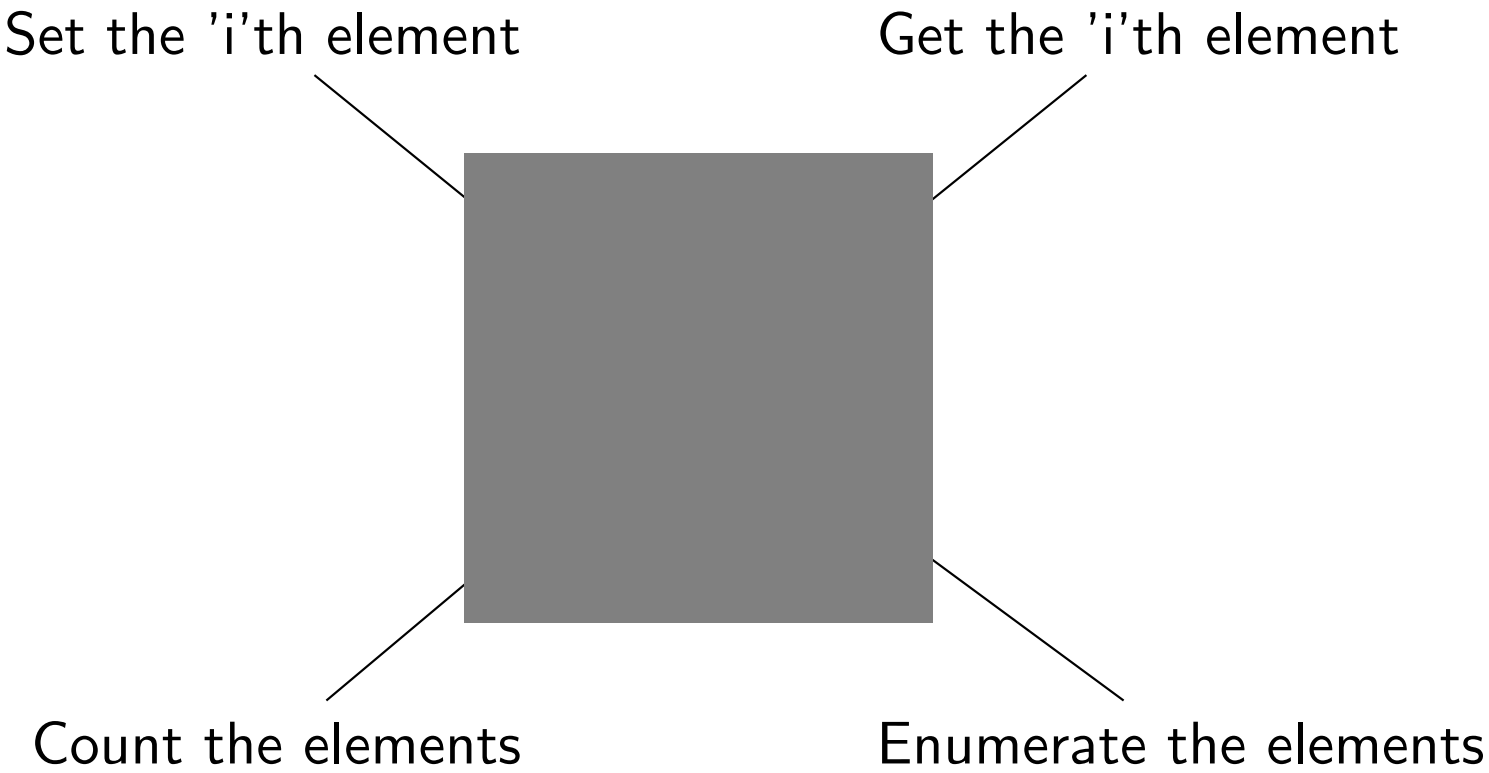
What are the simplest things we can do with a sequence?

What can we do with a sequence?

What are the simplest things we can do with a sequence?

- Get the 'i'th element.
- Modify the 'i'th element.
- Enumerate the elements in order.
- Get the number of elements

Abstract Data Types



Abstract Data Types

Abstract Data Type defines...

- Domain: What kind of data is stored? (e.g., elements, key/value pairs)
- Constraints: How are items related? (e.g., ordered keys)
- Operations: How can the data be accessed/modified (e.g., 'i'th item)

Abstract Data Types

Abstract Data Type defines...

- Domain: What kind of data is stored? (e.g., elements, key/value pairs)
- Constraints: How are items related? (e.g., ordered keys)
- Operations: How can the data be accessed/modified (e.g., 'i'th item)

Like a Java `interface`[2]

[2] The term `interface` is not quite the same as ADT; The interface only formalizes the permitted operations.

The Sequence ADT

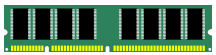
```
1  public interface Sequence<E>
2  {
3    public E get(int idx);
4    public void set(int idx, E value);
5    public int size();
6    public Iterator<E> iterator();
7  }
```

E is the type of thing in the Sequence.

The Sequence ADT

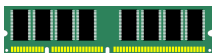
"The Wizard of Oz"; ©1939 Metro-Goldwyn-Mayer

CSE 220 Crossover



OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

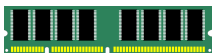
CSE 220 Crossover



01001000011001010110110001101100011101111...

OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover

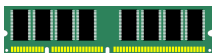


01001000011001010110110001101100011011111...

01001000

OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover



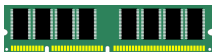
0100100001100101011011000110110001101111...

01001000

'H'

OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover



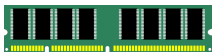
01001000011001010110110001101100011011111...

01001000 01100101 01101100 01101100 01101111

'H' 'e' 'l' 'l' 'o'

OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover



0100100001100101011011000110110001101111...

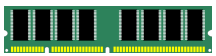
01001000 01100101 01101100 01101100 01101111

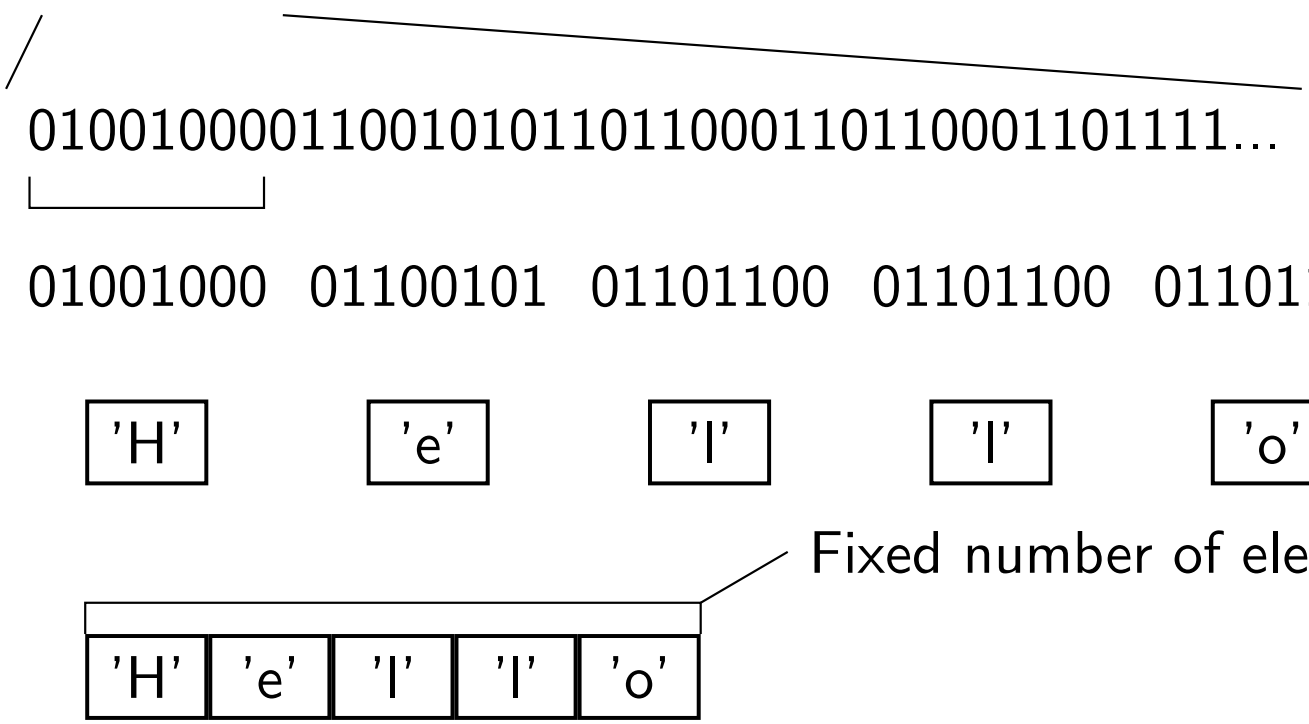
'H' 'e' 'l' 'l' 'o'

'H' 'e' 'l' 'l' 'o'

OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

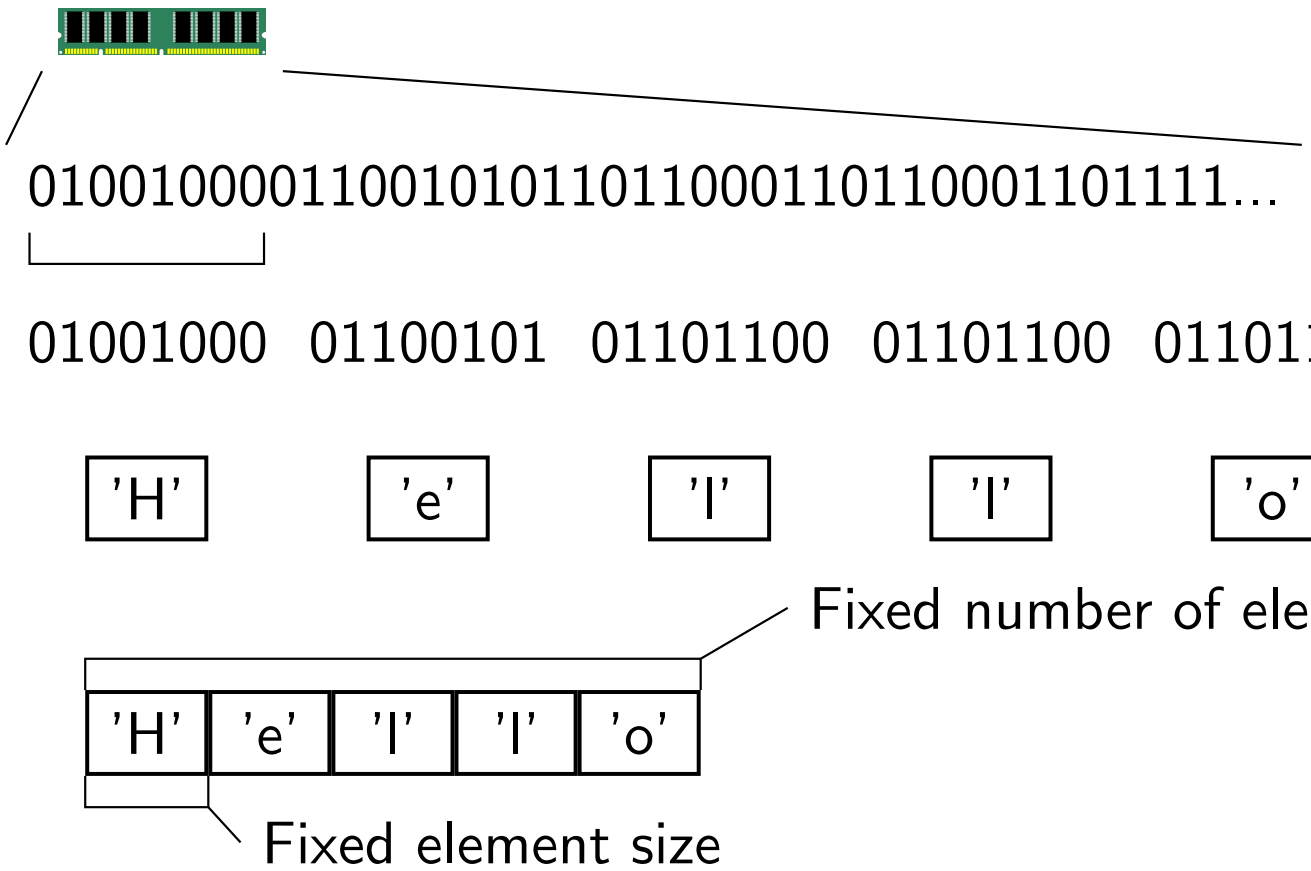
CSE 220 Crossover





OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover



OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

[3] You'll get very familiar with memory allocation in 220.
[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

[3] You'll get very familiar with memory allocation in 220.

[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

... allocates $4 \cdot 50 = 200$ bytes[4].

[3] You'll get very familiar with memory allocation in 220.

[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

... allocates $4 \cdot 50 = 200$ bytes[4].

If `data` is at address a: where do you look for `data[19]`?

[3] You'll get very familiar with memory allocation in 220.

[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

... allocates $4 \cdot 50 = 200$ bytes[4].

If `data` is at address a: where do you look for `data[19]`?
$a + 4 \cdot 19 = a + 76$

[3] You'll get very familiar with memory allocation in 220.
[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

... allocates $4 \cdot 50 = 200$ bytes[4].

If `data` is at address a: where do you look for `data[19]`?
$a + 4 \cdot 19 = a + 76$

How long does it take to locate `data[19]`?

[3] You'll get very familiar with memory allocation in 220.
[4] An `int` in java is 4 bytes (32 bits).

RAM

`new T` finds some unused part of memory big enough to fit a `T`, marks it used, and returns the address[3].

```
1    int[] data = new int[50];
```

... allocates $4 \cdot 50 = 200$ bytes[4].

If `data` is at address a: where do you look for `data[19]`?
$a + 4 \cdot 19 = a + 76$

How long does it take to locate `data[19]`?
$\theta(1)$

[3] You'll get very familiar with memory allocation in 220.
[4] An `int` in java is 4 bytes (32 bits).

Arrays

What information goes into an `T[]` array?

- `size`: 4 bytes for the number of elements.
- `bytesPerElement`: 4 bytes for `sizeof(T)`.
- `data`: `size` $\times$ `bytesPerElement` bytes.

Arrays

What information goes into an `T[]` array?

- `size`: 4 bytes for the number of elements[5].
- `bytesPerElement`: 4 bytes for `sizeof(T)`.
- `data`: `size` $\times$ `bytesPerElement` bytes.

[5] Some languages (e.g., C) skip this, relying on the programmer to track it.

Arrays

What information goes into an `T[]` array?

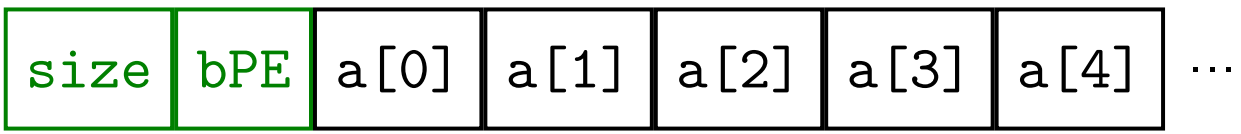
- `size`: 4 bytes for the number of elements[5].
- `bytesPerElement`: 4 bytes for `sizeof(T)`[6].
- `data`: `size` × `bytesPerElement` bytes.

[5] Some languages (e.g., C) skip this, relying on the programmer to track it.
[6] Some languages (e.g., C, C++) skip this, since it's fixed at compile time.

Arrays

What information goes into an `T[]` array?

- `size`: 4 bytes for the number of elements[5].
- `bytesPerElement`: 4 bytes for `sizeof(T)`[6].
- `data`: `size` × `bytesPerElement` bytes.



[5]Some languages (e.g., C) skip this, relying on the programmer to track it.
[6]Some languages (e.g., C, C++) skip this, since it's fixed at compile time.

How do we implement...

- `public E get(int idx)`

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`
  - Update bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`
  - Update bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)

How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`
  - Update bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public int size()`

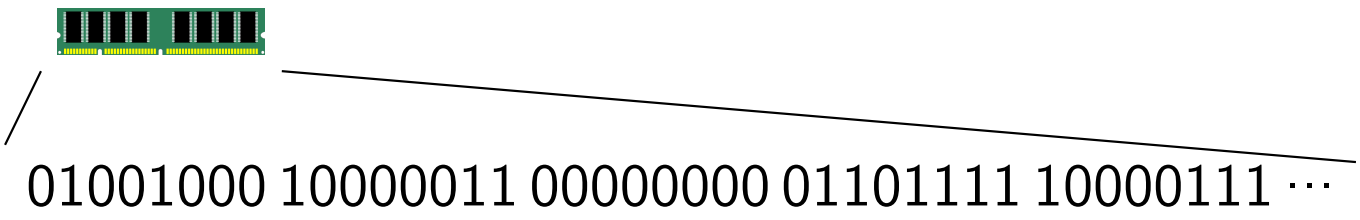
How do we implement...

- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`
  - Update bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public int size()`
  - Return `size`

How do we implement...

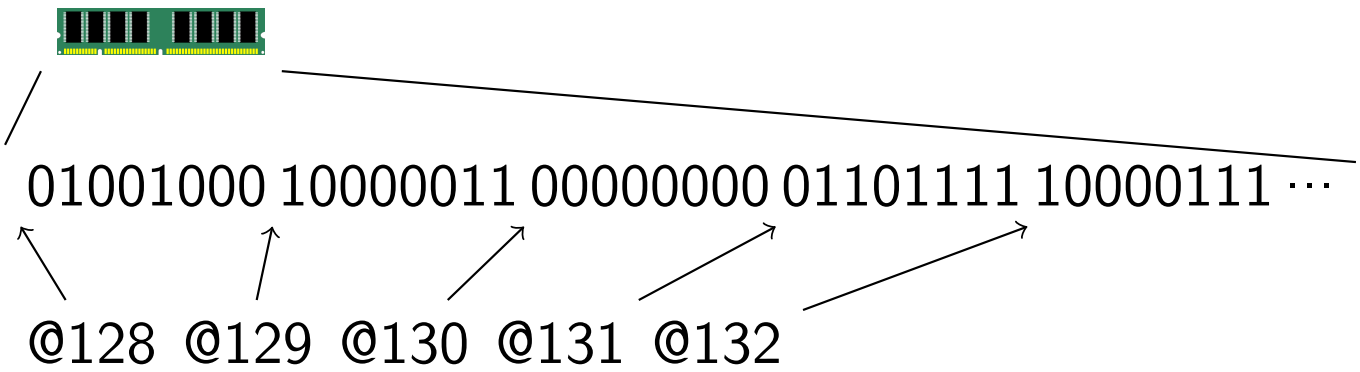
- `public E get(int idx)`
  - Return bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public void set(int idx, E value)`
  - Update bytes $\texttt{bPE} \times idx$ to $\texttt{bPE} \times (idx + 1) - 1$
  - $\theta(1)$ (if we treat `bPE` as a constant)
- `public int size()`
  - Return `size`
  - $\theta(1)$

CSE 220 Crossover 2: List Harder



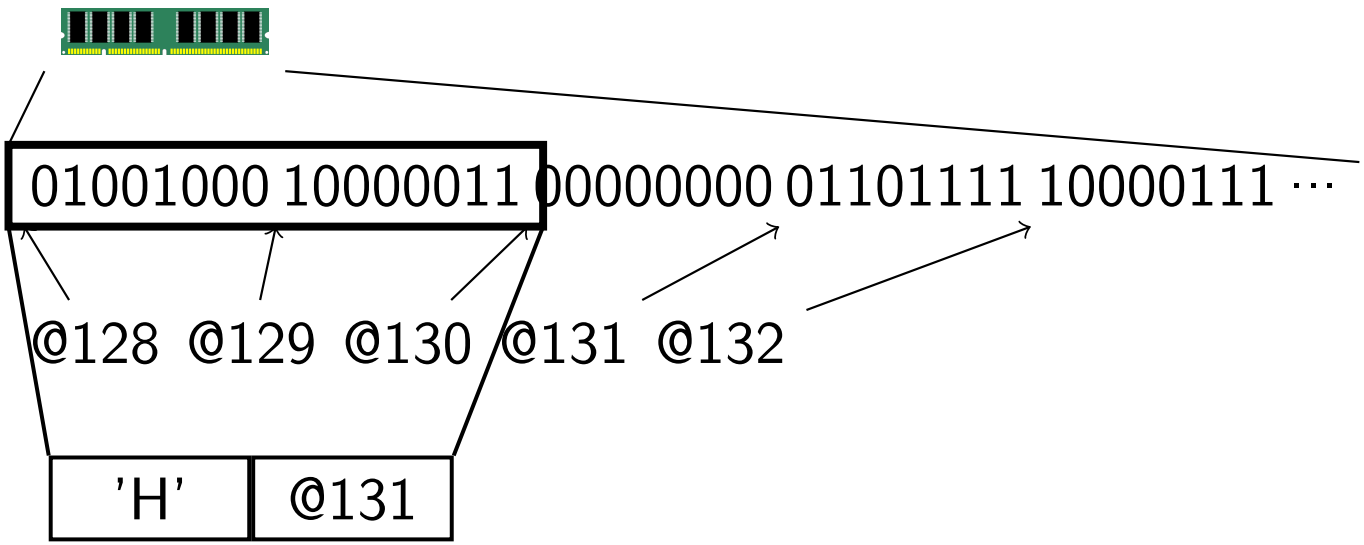
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder



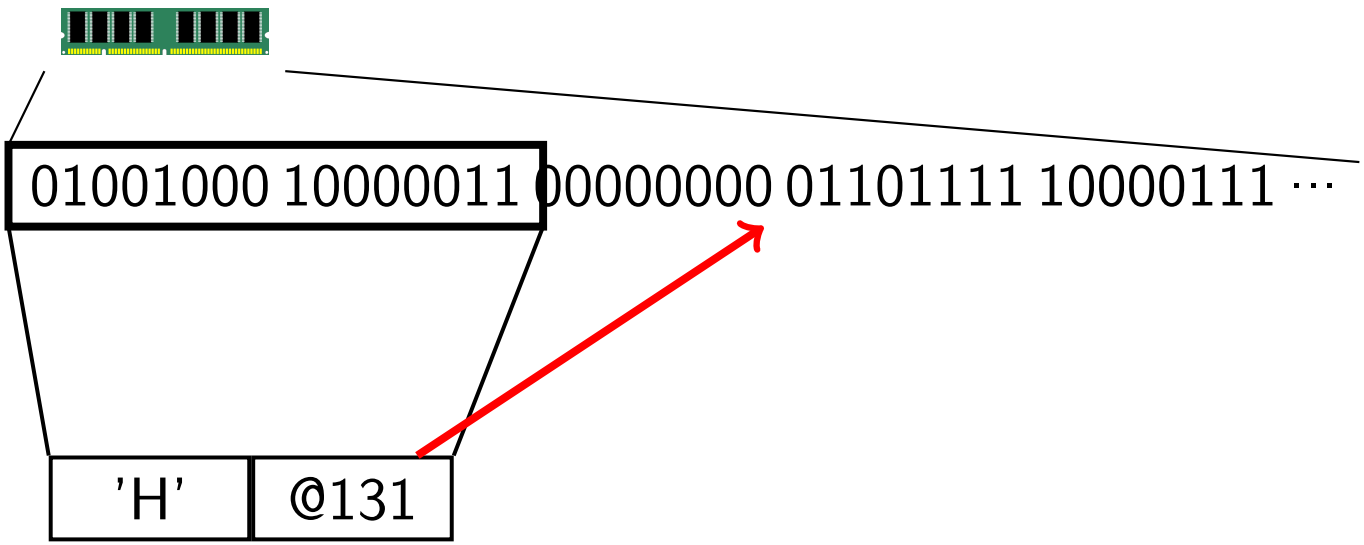
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder



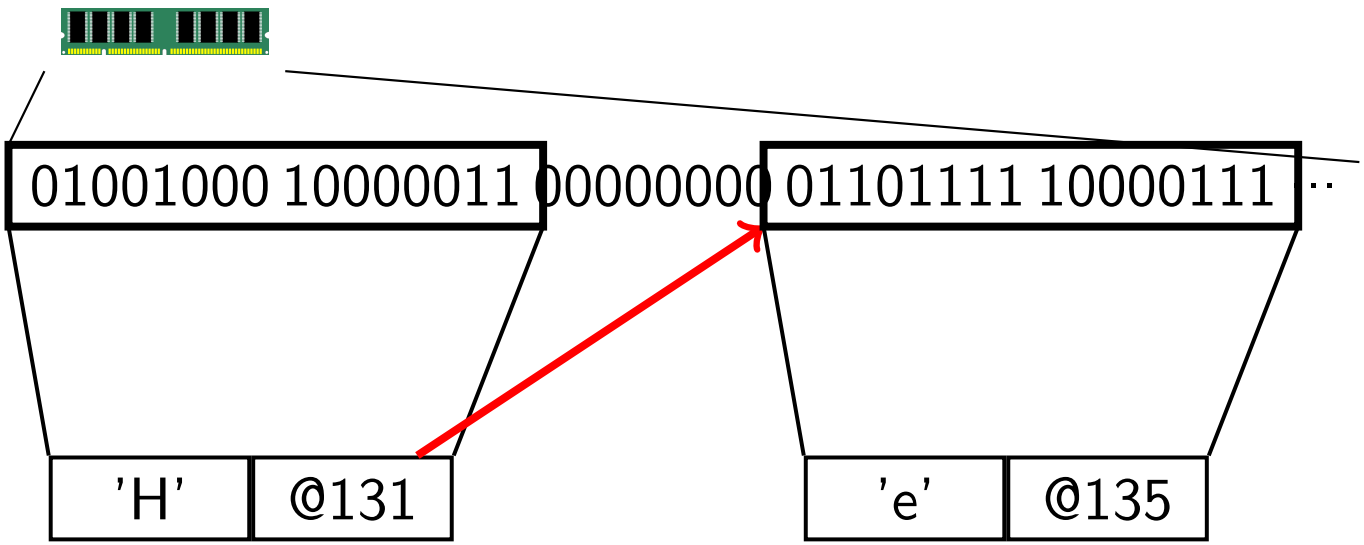
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder



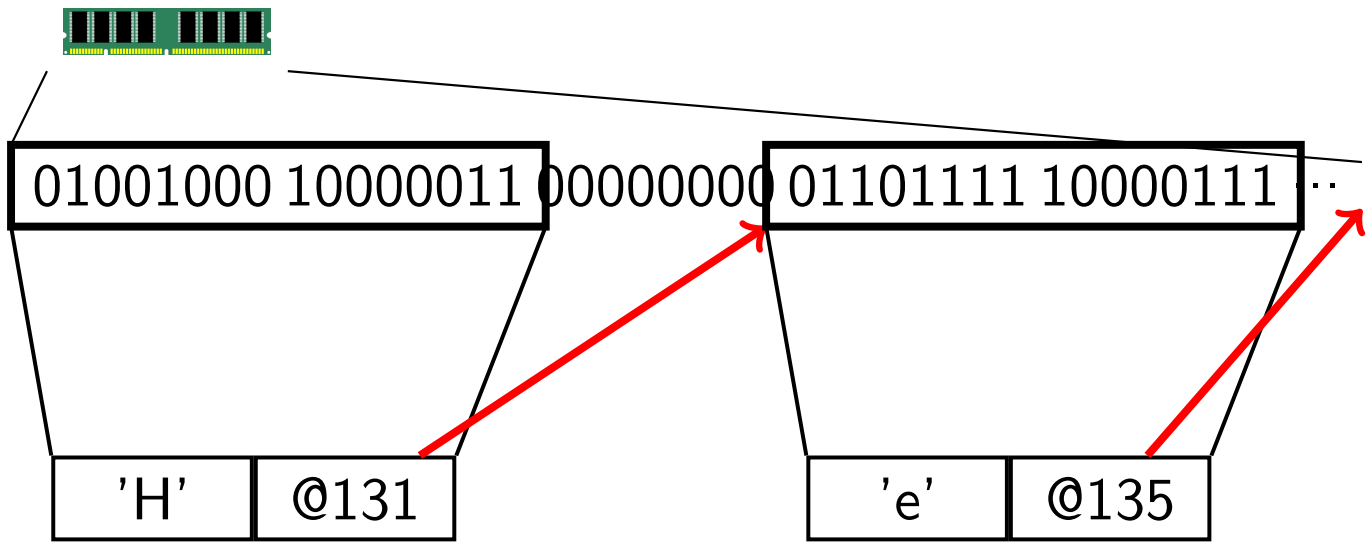
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder



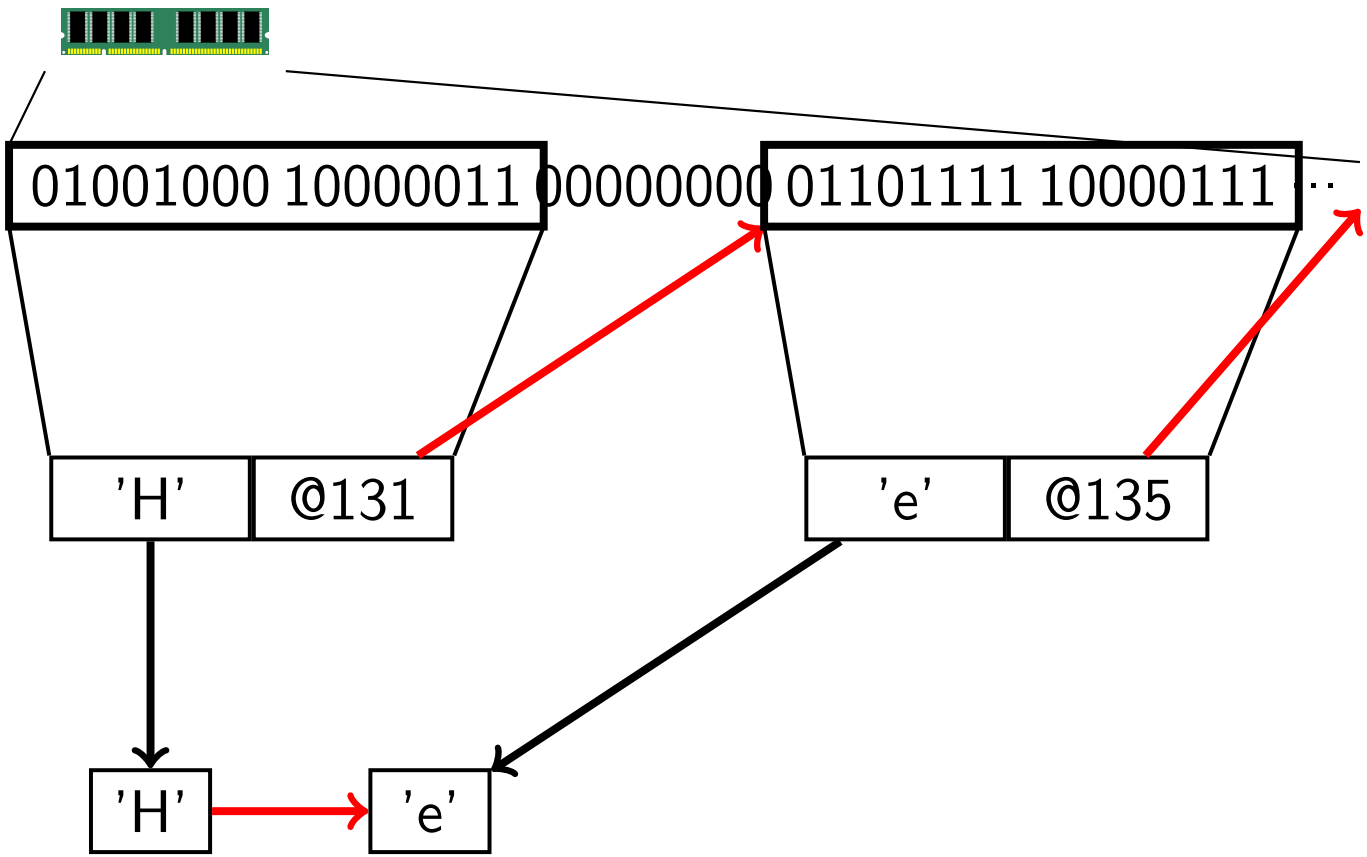
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder



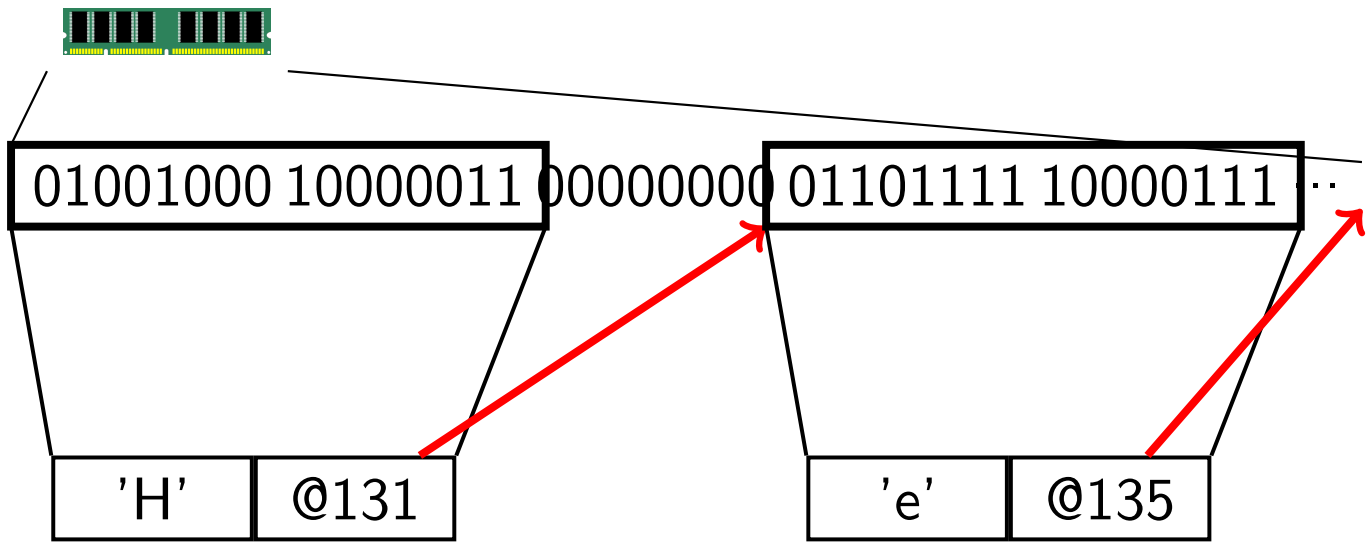
OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

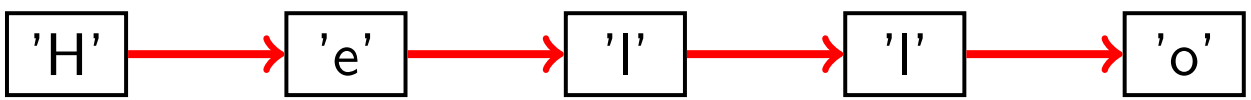
CSE 220 Crossover 2: List Harder



OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

CSE 220 Crossover 2: List Harder





OpenClipArt: https://freesvg.org/random-access-computer-memory-ram-vector-image

Linked Lists

```
1    public class LinkedListNode<T>
2    {
3      T value;
4      LinkedListNode<T> next = null;
5    }
```

Linked Lists

```java
public class LinkedListNode<T>
{
  T value;
  LinkedListNode<T> next = null;
}
```

```java
public class LinkedList<T> implements List<T>
{
  LinkedListNode<T> head = null;
  /* ... */
}
```

How do we implement...

- `public E get(int idx)`

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`
  - Start at `head`, and move to the `next` element `idx` times. Update the element's `value`.

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`
  - Start at `head`, and move to the `next` element `idx` times. Update the element's `value`.
  - $\theta(idx)$, $O(N)$

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`
  - Start at `head`, and move to the `next` element `idx` times. Update the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public int size()`

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`
  - Start at `head`, and move to the `next` element `idx` times. Update the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public int size()`
  - Start at `head`, and move to the `next` element until you reach the end. Return the number of steps taken.

How do we implement...

- `public E get(int idx)`
  - Start at `head`, and move to the `next` element `idx` times. Return the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public void set(int idx, E value)`
  - Start at `head`, and move to the `next` element `idx` times. Update the element's `value`.
  - $\theta(idx)$, $O(N)$
- `public int size()`
  - Start at `head`, and move to the `next` element until you reach the end. Return the number of steps taken.
  - $\theta(N)$

Linked Lists' `size`

Can we do better?

Store `size`

```
1    public class LinkedList<T> implements List<T>
2    {
3      LinkedListNode<T> head = null;
4      int size = 0;
5      /* ... */
6    }
```

- How expensive is `public int size()` now?

Store `size`

```
1    public class LinkedList<T> implements List<T>
2    {
3      LinkedListNode<T> head = null;
4      int size = 0;
5      /* ... */
6    }
```

- How expensive is `public int size()` now? ($\theta(1)$)
- How expensive is it to maintain `size`?

Store `size`

```
1   public class LinkedList<T> implements List<T>
2   {
3     LinkedListNode<T> head = null;
4     int size = 0;
5     /* ... */
6   }
```

- How expensive is `public int size()` now? ($\theta(1)$)
- How expensive is it to maintain `size`? (Extra $\theta(1)$ work on insert/remove).

Store `size`

```
1    public class LinkedList<T> implements List<T>
2    {
3      LinkedListNode<T> head = null;
4      int size = 0;
5      /* ... */
6    }
```

- How expensive is `public int size()` now? ($\theta(1)$)
- How expensive is it to maintain `size`? (Extra $\theta(1)$ work on insert/remove).

Storing redundant information can reduce complexity.

Optional

```java
public class LinkedListNode<T>
{
  T value;
  LinkedListNode<T> next = null;
}
```

```java
public class LinkedList<T> implements List<T>
{
  LinkedListNode<T> head = null;
  /* ... */
}
```

null is risky

```
1  public T get(int idx)
2  {
3    LinkedListNode<T> curr = head;
4    for(int j = 0; j < i; j++)
5    {
6      curr = curr.next;
7    }
8    return curr.value;
9  }
```

null is risky

```
1   public T get(int idx)
2   {
3     LinkedListNode<T> curr = head;
4     for(int j = 0; j < i; j++)
5     {
6       curr = curr.next;
7     }
8     return curr.value;
9   }
```

NullPointerException

null is risky

```
1   public T get(int idx)
2   {
3     LinkedListNode<T> curr = head;
4     for(int j = 0; j < i; j++)
5     {
6       if(curr.next == null){ /* do something useful */ }
7       curr = curr.next;
8     }
9     return curr.value;
10  }
```

null is risky

```
1   public T get(int idx)
2   {
3     LinkedListNode<T> curr = head;
4     for(int j = 0; j < i; j++)
5     {
6       if(curr.next == null){ /* do something useful */ }
7       curr = curr.next;
8     }
9     return curr.value;
10  }
```

It can be hard to remember when a value might be null.

Optional

```java
public class LinkedListNode<T>
{
  T value;
  Optional<LinkedListNode<T>> next = Optional.empty();
}
```

```java
public class LinkedList<T> implements List<T>
{
  Optional<LinkedListNode<T>> head = Optional.empty();
  /* ... */
}
```

Optional

```
1  public T get(int idx)
2  {
3    Optional<LinkedListNode<T>> curr = head;
4    for(int j = 0; j < i; j++)
5    {
6      curr = curr.next;
7    }
8    return curr.value;
9  }
```

Optional

```
1  public T get(int idx)
2  {
3    Optional<LinkedListNode<T>> curr = head;
4    for(int j = 0; j < i; j++)
5    {
6      curr = curr.next; ←
7    }
8    return curr.value; ←
9  }
```

Compiler Error! (No such method)

Optional

```
1  public T get(int idx)
2  {
3    Optional<LinkedListNode<T>> curr = head;
4    for(int j = 0; j < i; j++)
5    {
6      curr = curr.get().next;
7    }
8    return curr.get().value;
9  }
```

Optional

```
1  public T get(int idx)
2  {
3    Optional<LinkedListNode<T>> curr = head;
4    for(int j = 0; j < i; j++)
5    {
6      curr = curr.get().next; ←
7    }
8    return curr.get().value; ←
9  }
```

Compiler Error! (Unhandled Exception)

Optional

```
1   public T get(int idx)
2   {
3     Optional<LinkedListNode<T>> curr = head;
4     for(int j = 0; j < i; j++)
5     {
6       try {
7         curr.get().next;
8       } catch(NoSuchElementException e) {
9         /* do something useful */
10      }
11    }
12    try {
13      return curr.get().value;
14    } catch(NoSuchElementException e) {
15      /* do something useful */
16    }
17  }
```

Optional

Creating an Optional

- `Optional.empty()`: Like `null`.
- `Optional.of(x)`: Get an Optional with value `x`.
- `Optional.ofNullable(x)`: Like `.of(x)`, but return `.empty()` if `x == null`;

Using an Optional

- `.isPresent()`: Return true if a value is present.
- `.get()`: Return the value, or throw exception if not present.
- `.orElse(y)`: Return the value if present, or `y` if not.

Referential Access

What's the complexity of getting the value of the i'th element of a `LinkedList`?

Referential Access

What's the complexity of getting the value of the *i*'th element of a `LinkedList`?

If you have a pointer to the *i*'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting its value?

Referential Access

What's the complexity of getting the value of the i'th element of a `LinkedList`?

If you have a pointer to the i'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting its value?

If you have a pointer to the i'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting the value of the $i+1$'th element?

Referential Access

What's the complexity of getting the value of the i'th element of a `LinkedList`?

If you have a pointer to the i'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting its value?

If you have a pointer to the i'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting the value of the $i+1$'th element?

If you have a pointer to the i'th `LinkedListNode` of a `LinkedList`, what's the complexity of getting the value of the $i-1$'th element?

Doubly Linked Lists

```java
public class LinkedListNode<T>
{
  T value;
  Optional<LinkedListNode<T>> next = Optional.empty();
  Optional<LinkedListNode<T>> prev = Optional.empty();
}
```

```java
public class LinkedList<T> implements List<T>
{
  Optional<LinkedListNode<T>> head = Optional.empty();
  Optional<LinkedListNode<T>> tail = Optional.empty();
  /* ... */
}
```

Linked Lists

Next time...

- Referential Access
- The full `List` type (insertions/deletions).
- More on Doubly-Linked Lists