Algorithms

Part a: Basic Data Structures

Ian Ludden

Learning Objectives

By the end of this lesson, you will be able to:

Learning Objectives

By the end of this lesson, you will be able to:

- Be able to read code that uses basic linked list operations (first, rest, cons).

Learning Objectives

By the end of this lesson, you will be able to:

- Be able to read code that uses basic linked list operations (first, rest, cons).
- Know the big-O running times of basic operations on linked lists and arrays.

Learning Objectives

By the end of this lesson, you will be able to:

- Be able to read code that uses basic linked list operations (first, rest, cons).
- Know the big-O running times of basic operations on linked lists and arrays.
- For an algorithm involving loops (perhaps nested), express its running time using summations.

Learning Objectives

By the end of this lesson, you will be able to:

- Be able to read code that uses basic linked list operations (first, rest, cons).
- Know the big-O running times of basic operations on linked lists and arrays.
- For an algorithm involving loops (perhaps nested), express its running time using summations.
- Given an unfamiliar but fairly simple function in pseudo-code, analyze how long it takes using big-O notation.

Arrays vs. Linked Lists

	Arrays			**Linked Lists**		
	R/W	Add	Remove	R/W	Add	Remove
At/near start						
In middle						
At end						

Arrays vs. Linked Lists

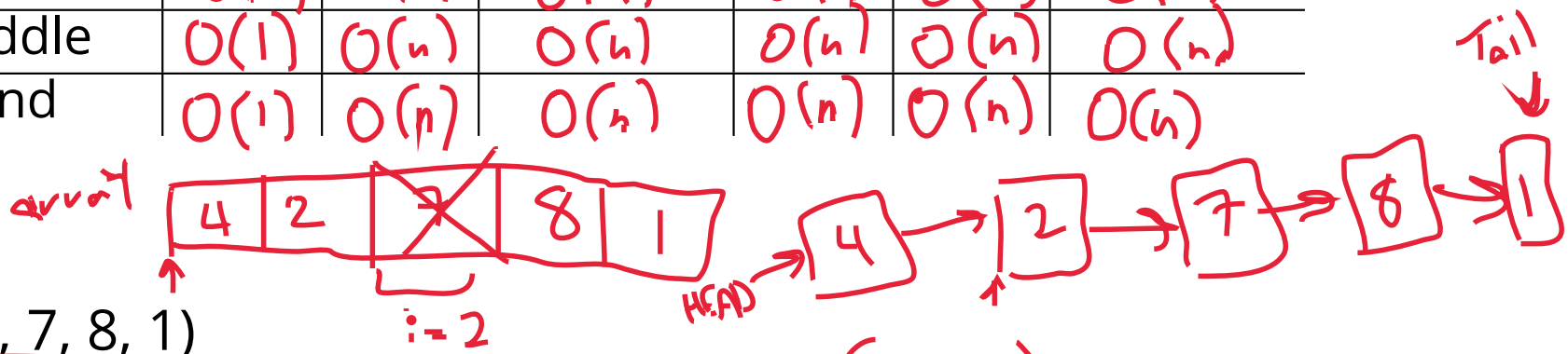
147 is O(1)

"constant" time

"linear" time

	Arrays			Linked Lists		
	R/W	Add	Remove	R/W	Add	Remove
At/near start	O(1)	O(n)	O(n)	O(1)	O(1)	O(1)
In middle	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)
At end	O(1)	O(n)	O(n)	O(n)	O(n)	O(n)

$\frac{n}{2}$



L = (4, 2, 7, 8, 1)

- first(L) returns first element (head)
- rest(L) : everything after first element
- cons(element, L) adds element as head of list

O(1)

↓

"constructs"

Loops, Example 1: Max-area Triangle with Origin

Given an array of n 2D points, find a pair of points that maximizes the area of the triangle formed with the origin.

```
function max_triangle(arr)
    origin = (0, 0)

    max_area = 0
    best_x = null
    best_y = null

    for i from 1 to n
        x = arr[i]
        for j from i + 1 to n
            y = arr[j]

            a = dist(x, origin) // length of side a
            b = dist(y, origin) // length of side b
            c = dist(x, y)      // length of side c

            s = (a + b + c) / 2 // semiperimeter

            area = sqrt(s * (s - a) * (s - b) * (s - c)) // Heron's Formula

            if area > max_area
                max_area = area
                best_x = x
                best_y = y

    return max_area // Could also return best_x and best_y if desired
```

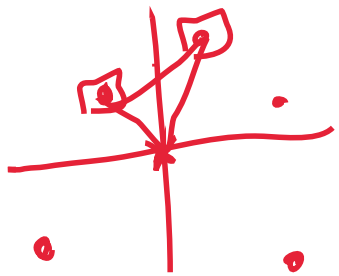
$$\sum_{i=1}^{n}\left(\sum_{j=i+1}^{n} c\right) = \sum_{i=1}^{n} c(n-i)$$

$$= c\sum_{i=1}^{n} n - c\sum_{i=1}^{n} i$$

$$= cn^2 - c\left(\frac{n(n+1)}{2}\right)$$

$$= c\cdot\frac{n(n-1)}{2}$$

, which is $O(n^2)$.



Loops, Example 1: Max-area Triangle with Origin

Given an array of n 2D points, find a pair of points that maximizes the area of the triangle formed with the origin.

```
function max_triangle(arr)
    origin = (0, 0)

    max_area = 0
    best_x = null
    best_y = null

    for i from 1 to n
        x = arr[i]
        for j from i + 1 to n
            y = arr[j]

            a = dist(x, origin) // length of side a
            b = dist(y, origin) // length of side b
            c = dist(x, y)      // length of side c

            s = (a + b + c) / 2 // semiperimeter

            area = sqrt(s * (s - a) * (s - b) * (s - c)) // Heron's Formula

            if area > max_area
                max_area = area
                best_x = x
                best_y = y

    return max_area // Could also return best_x and best_y if desired
```

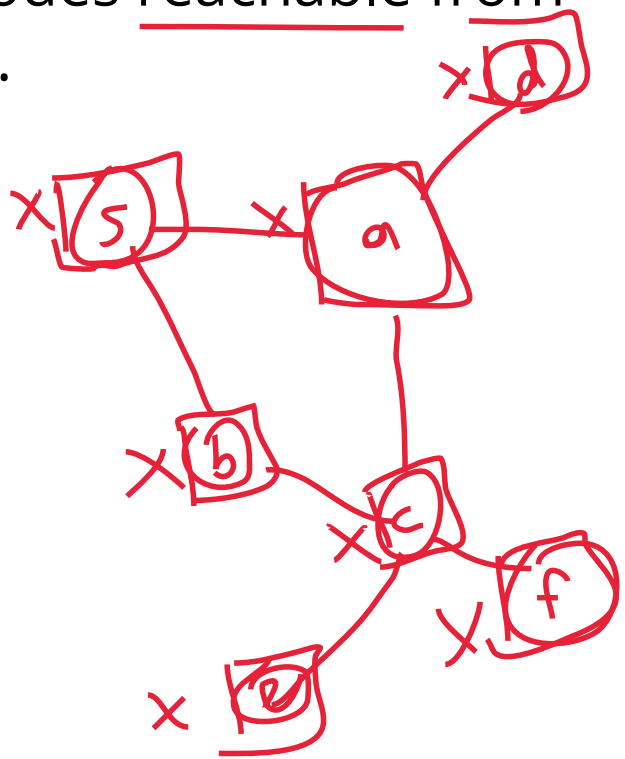
Loops, Example 2: Graph Reachability

Given a graph G and a start vertex s, find all nodes reachable from s (i.e., in the same connected component as s).

```
1  function component(G = (V, E), s)
2      for v in V
3          unmark(v)
4
5      RV = emptylist
6      Q = emptylist
7      mark(s)
8      cons(s, Q)
9
10     while (Q is not empty)
11         u = first(Q)
12         Q = rest(Q)
13         cons(u, RV)
14
15         for every v in V such that uv in E
16             if v is not marked
17                 mark v
18                 cons(v, Q)
19
20     return RV
21
```

// return vertices "to be explored" vertex

remove head



Loops, Example 2: Graph Reachability

$|V| = n$, $|E| = m$

Given a graph G and a start vertex s, find all nodes reachable from s (i.e., in the same connected component as s).

```
1  function component(G = (V, E), s)
2      for v in V
3          unmark(v)
4
5      RV = emptylist
6      Q = emptylist
7      mark(s)
8      cons(s, Q)
9
10     while (Q is not empty)
11         u = first(Q)
12         Q = rest(Q)
13         cons(u, RV)
14
15         for every v in V such that uv in E
16             if v is not marked
17                 mark v
18                 cons(v, Q)
19
20     return RV
21
```

① ② ③ ④

$O(n) + O(1) + O(n) + O(m) = O(1 + 2n + m) = O(n + m)$

at most n iterations

at most $2m$ iterations total!

$n \cdot O(1) = O(n)$

$2m \cdot O(1) = O(m)$

Learning Objectives

By the end of this lesson, you will be able to:

- Be able to read code that uses basic linked list operations (first, rest, cons).
- Know the big-O running times of basic operations on linked lists and arrays.
- For an algorithm involving loops (perhaps nested), express its running time using summations.
- Given an unfamiliar but fairly simple function in pseudo-code, analyze how long it takes using big-O notation.