Sorting, Searching, and Big O

Doug Blank, Fall 2010

Overview

- Sorting
  - Slowsort
  - Bubblesort
  - Mergesort
  - Quicksort
- Searching a Graph
  - Breadth First Search
  - Depth First Search
- Additional Data Structures and terms

Slowsort

```
def slowsort(list):
    for i in range(0, len(list) - 1):
        for j in range(i + 1, len(list)):
            if list[i] > list[j]:
                list[i], list[j] = list[j], list[i]
```

Bubblesort

```
def bubblesort(list):
    for j in range(len(list) - 1, 1, -1):
        swap = False
        for i in range(j):
            if list[i] > list[i + 1]:
                list[i], list[i + 1] = list[i + 1], list[i]
                swap = True
        if not swap:
            break
```

Mergesort

```
def mergesort(list):
    if len(list) < 2:
        return list
    middle = len(list) / 2
    left = mergesort(list[:middle])
    right = mergesort(list[middle:])
    return merge(left, right)

def merge(left, right):
    i ,j, result = 0, 0, []
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result += left[i:]
    result += right[j:]
    return result
```

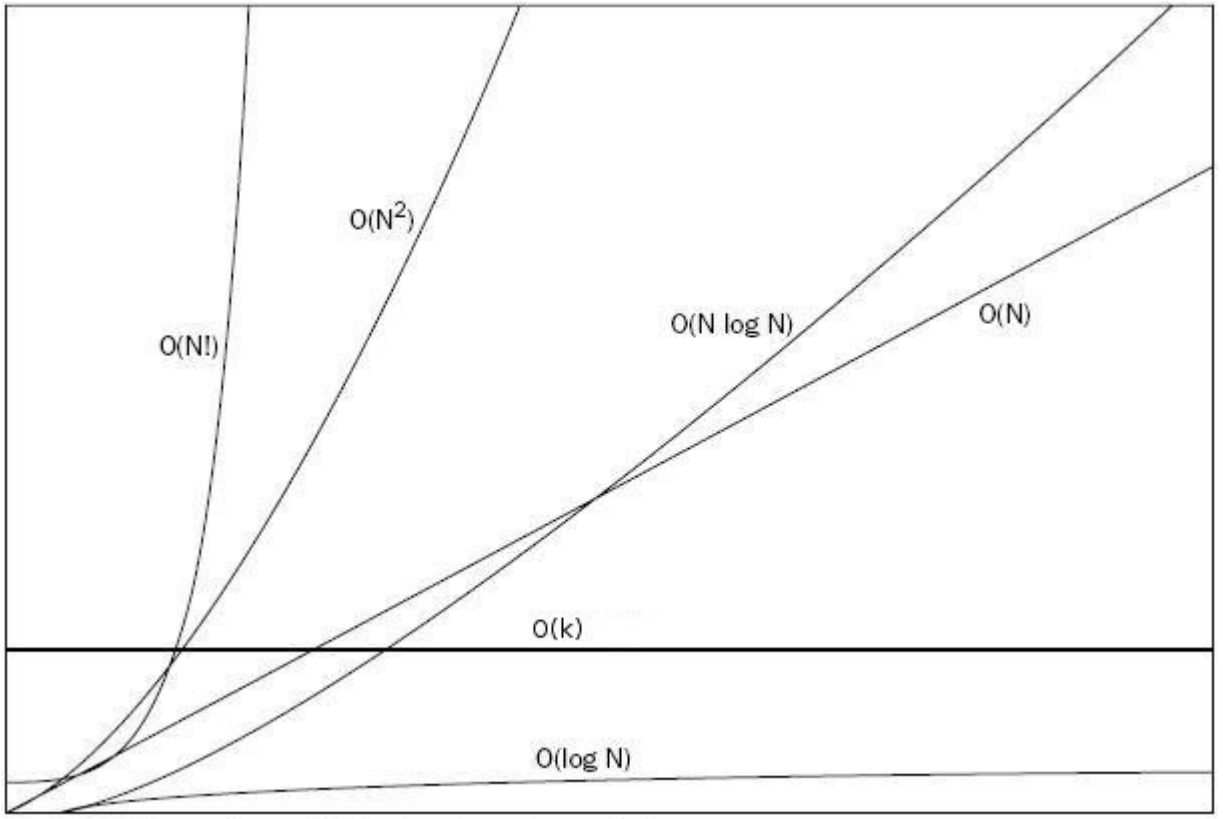
Quicksort

```
def quicksort(list):
    if len(list) > 1:
        pivot = len(list)/2
        elements = list[:pivot] + list[pivot+1:]
        left, right = [], []
        for element in elements:
                if element < list[pivot]:
                    left.append(element)
                else:
                    right.append(element)
        return quicksort(left) + [list[pivot]] + quicksort(right)
    else:
        return list
```

Sorting Comparison

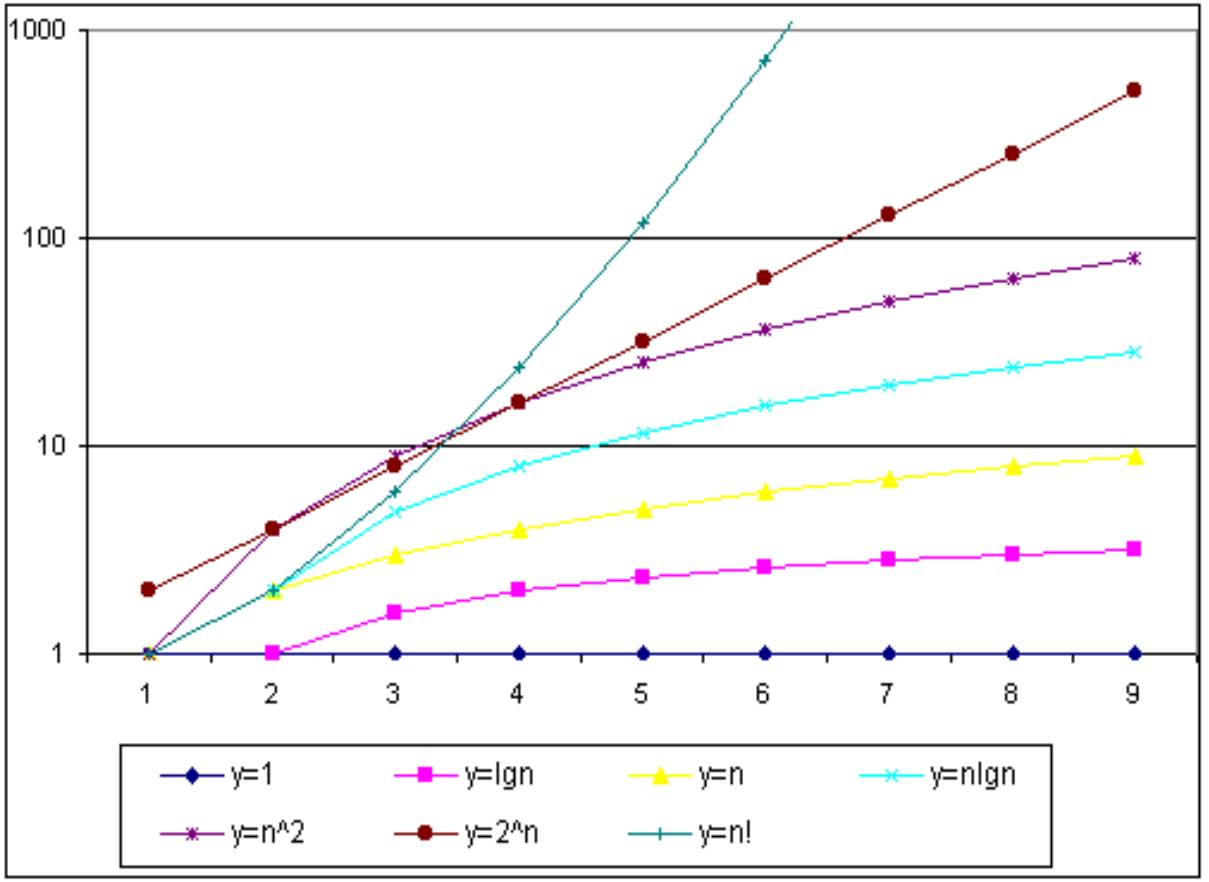
- Slowsort
  - best: O(n ** 2), worst: O(n ** 2), in-place
- Bubblesort
  - best: O(n), worst: O(n ** 2), in-place
- Mergesort
  - best: O(n log n), worst: O(n log n)
- Quicksort
  - best: O(n log n), worst: O(n log n)

Orders of Complexity

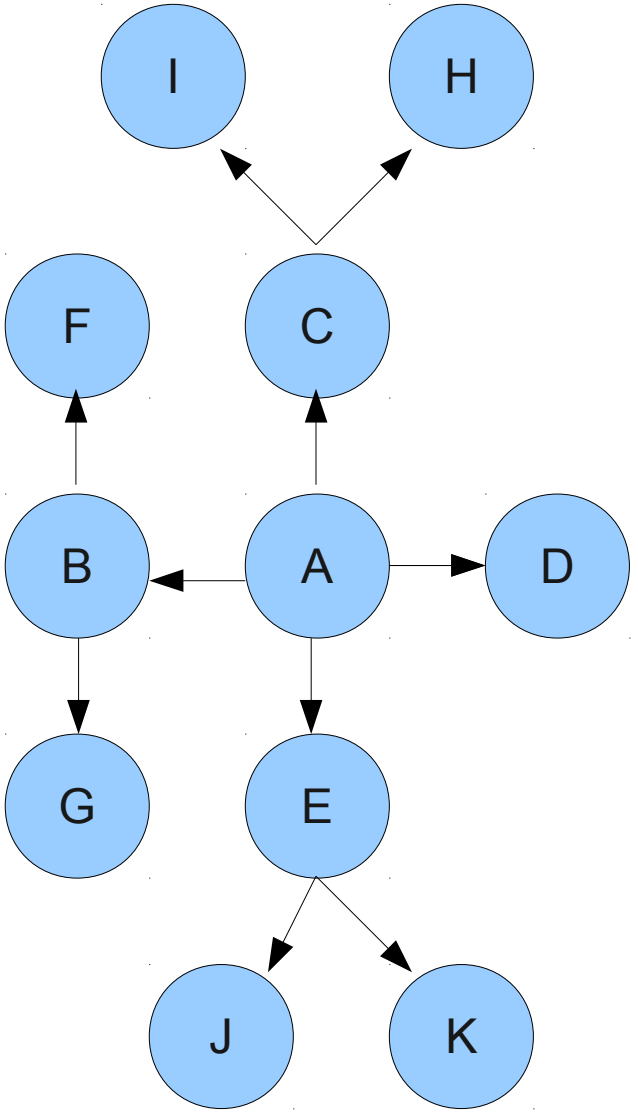


Comparison of different orders of complexity.

Orders of Complexity, Log Plot



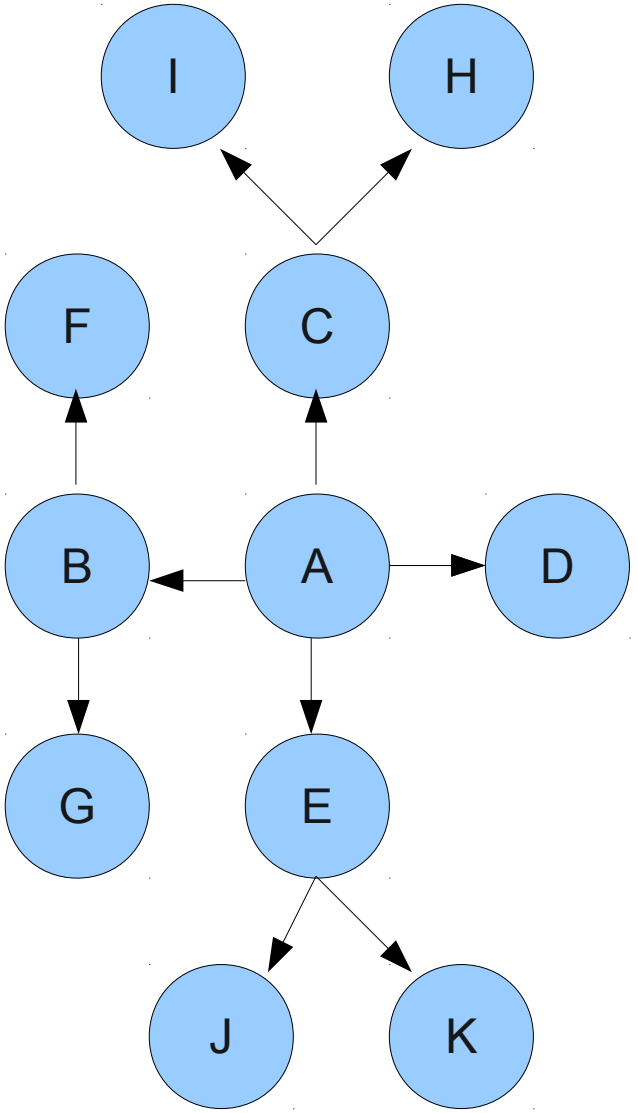
Searching a Graph



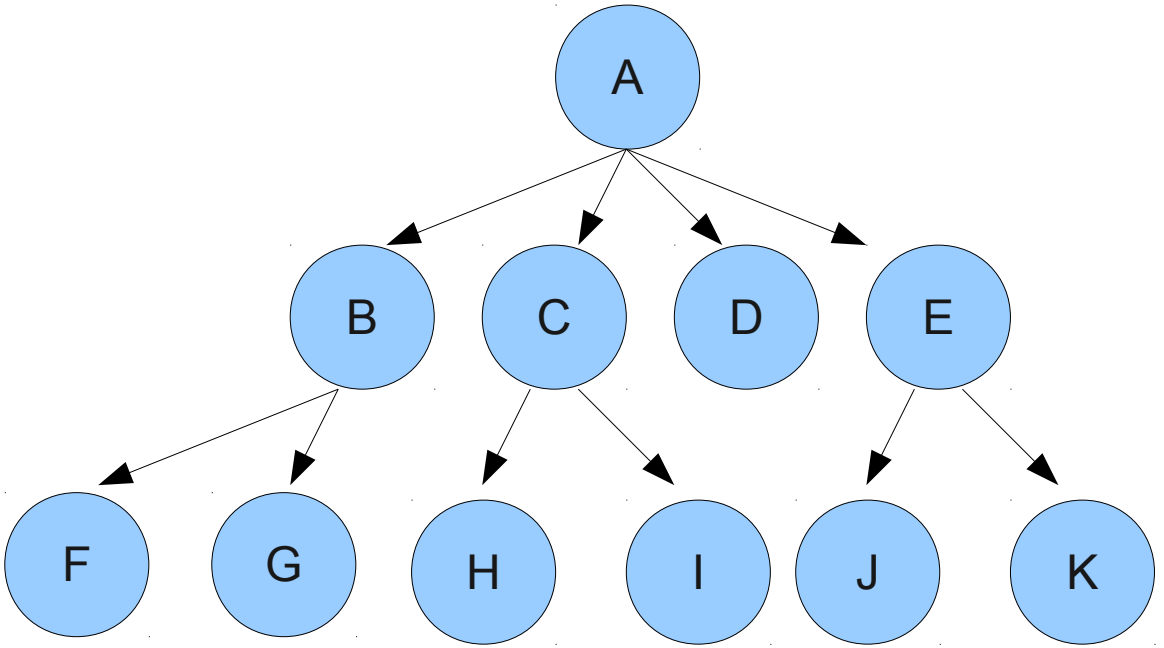
Graph Questions

- Does a path from A to K exist?
- What is the longest path in the graph?
- Are there cycles in the graph?
- What is the shortest path from A to K?

Searching a Graph



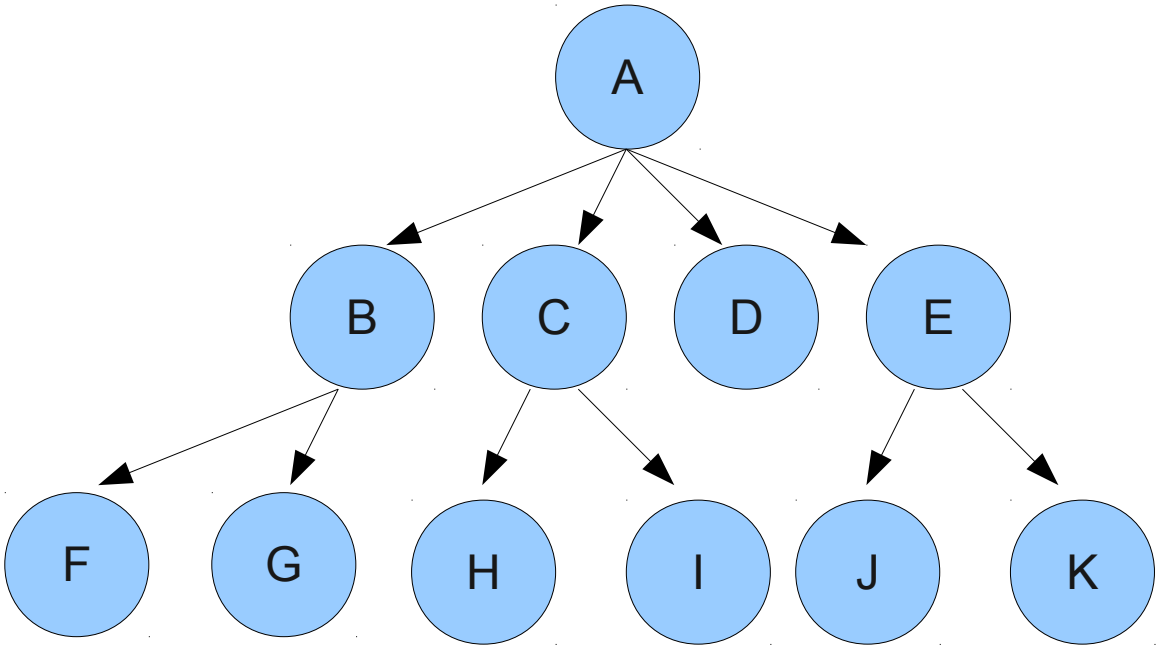
Searching a Graph



Breadth First Search

```
def BFS(graph, goal):
    current = graph.initial_state
    to_explore = expand(current)
    while to_explore:
        if current == goal:
            break
        current = to_explore[0]
        to_explore = to_explore + expand(current)
```

Searching a Graph



Depth First Search

```
def DFS(graph, goal):
    current = graph.initial_state
    to_explore = expand(current)
    while to_explore:
        if current == goal:
            break
        current = to_explore[0]
        to_explore = expand(current) + to_explore
```

Need to prevent Cycles

```
def BFS(graph, goal):
    visited = [ ]
    current = graph.initial_state
    to_explore = expand(current)
    while to_explore:
        visited.append(current)
        if current == goal:
            break
        current = to_explore[0]
        if not current in visited:
            to_explore = to_explore + expand(current)
```

Additional Data Structures and Terms

- ADT – Abstract Data Type, OOP data structure
- Dictionary: Hash, Hash table, Associative list
  - Lookup: O(1)
- Stack – first-in, last-out
- Queue – first-in, first-out