

## Objectives

- Data structure: Heaps
- Data structure: Graphs

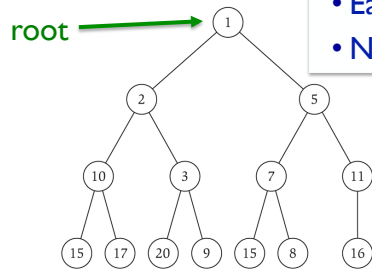
Submit problem set 2

## Review

- What is a priority queue?
- What is a heap?
  - Properties
  - Implementation
- What is the process for finding the smallest element in a heap?
- What is the process for adding to a heap?

## Review: Heap Defined

- Combines benefits of sorted array and list
- Balanced binary tree



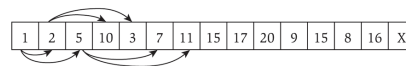
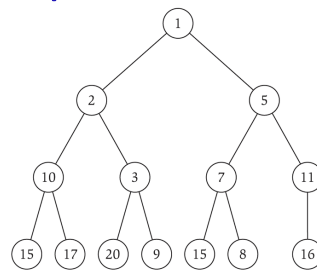
- Each node has *at most* 2 children
- Node value is its key

**Heap order:** each node's key is at least as large as its parent's

Note: **not** a binary search tree

## Review: Implementing a Heap

- Option 1: Use pointers
  - Each node keeps
    - Element it stores (key)
    - 3 pointers: 2 children, parent
- Option 2: No pointers
  - Requires knowing upper bound on  $n$
  - For node at position  $i$ 
    - left child is at  $2i$
    - right child is at  $2i+1$



## Review: Implementing a Heap

- Finding the minimal element
  - First element
  - $O(1)$

Jan 26, 2018

CSCI211 - Srenkle

5

## Review: Heapify-Up

Heap      Position where node added

```

Heapify-up(H, i):
  if i > 1 then
    j=parent(i)=floor(i/2)
    if key[H[i]] < key[H[j]] then
      swap array entries H[i] and H[j]
    Heapify-up(H, j)
  
```

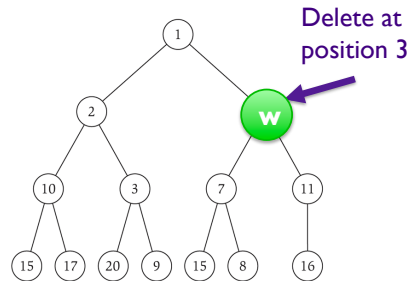
- $O(\log i)$

Jan 26, 2018

CSCI211 - Srenkle

6

## Deleting an Element



Jan 26, 2018

CSCI211 - Sprenkle

7

## Deleting an Element

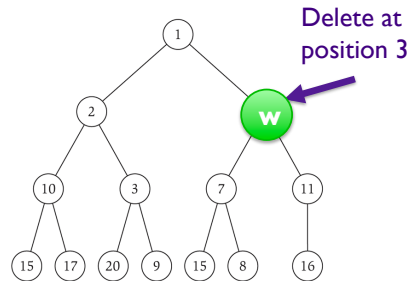
- Delete at position  $i$
- Removing an element:
  - Messes up heap order
  - Leaves a “hole” in the heap
- Not as straightforward as **Heapify-Up**
- Algorithm
  1. Fill in element where hole was
    - Patch hole: move  $n^{\text{th}}$  element into  $i^{\text{th}}$  spot
  2. Adjust heap to be in order
    - At position  $i$  because moved  $n^{\text{th}}$  item up to  $i$

Jan 26, 2018

CSCI211 - Sprenkle

8

## Deleting an Element



Example of OK:  
11 deleted, replaced by 16

- Two “bad” possibilities: element  $w$  is
  - Too small: violation is between it and parent → **Heapify-Up**
  - Too big: with one or both children → **Heapify-Down** (example:  $w$  becomes 12)

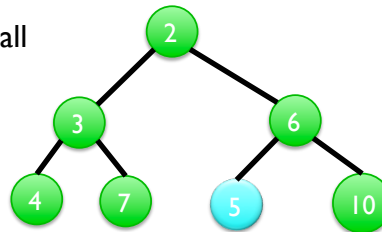
Jan 26, 2018

CSCI211 - Sprenkle

9

## Deleting an Element

Example where new key is too small



- Delete 9
- Replace with 5 (from other side of heap)
- But  $5 < 6$ , so need to **Heapify-Up**

Jan 26, 2018

CSCI211 - Sprenkle

10

## Heapify-Down

```

Heapify-down(H, i):
  n = length(H)
  if 2i > n then           Why can we stop?
    Terminate with H unchanged
  else if 2i < n then
    left=2i and right=2i+1
    j be index that minimizes
      key[H[left]] and key[[H[right]]]
  else if 2i = n then
    j=2i

  if key[H[j]] < key[H[i]] then
    swap array entries H[i] and H[j]
    Heapify-down(H, j)

```

Jan 26, 2018

CSCI211 - Sprenkle

11

## Heapify-Down

```

Heapify-down(H, i):
  n = length(H)
  if 2i > n then           i is a leaf – nowhere to go
    Terminate with H unchanged
  else if 2i < n then
    left=2i and right=2i+1
    j be index that minimizes
      key[H[left]] and key[[H[right]]]
  else if 2i = n then
    j=2i

  if key[H[j]] < key[H[i]] then
    swap array entries H[i] and H[j]
    Heapify-down(H, j)

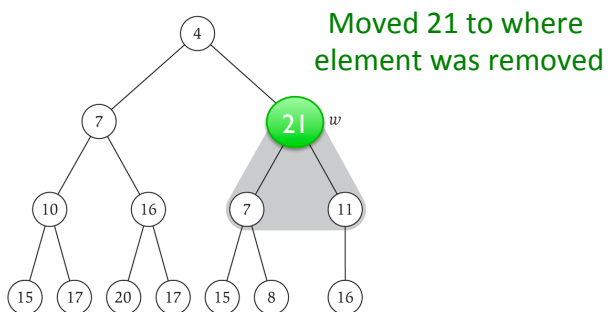
```

Jan 26, 2018

CSCI211 - Sprenkle

12

## Practice: Heapify-Down

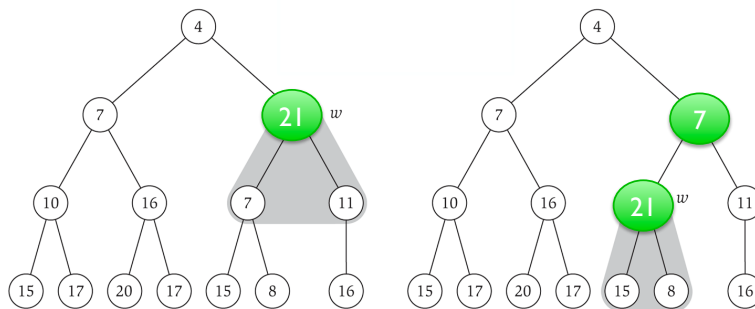


Jan 26, 2018

CSCI211 - Sprenkle

13

## Practice: Heapify-Down

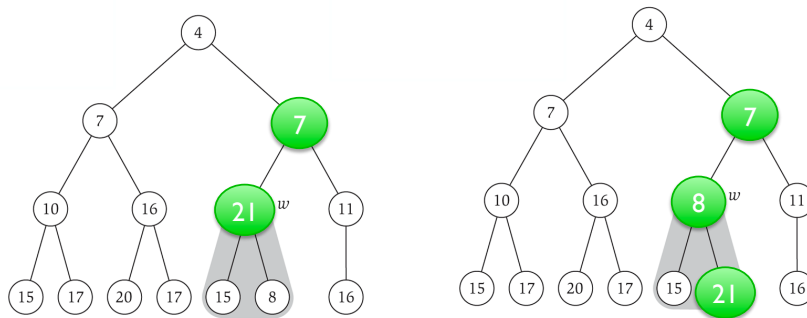


Jan 26, 2018

CSCI211 - Sprenkle

14

## Practice: Heapify-Down



Jan 26, 2018

CSCI211 - Sprenkle

15

## Runtime of Heapify-Down?

```

Heapify-down(H, i):
  n = length(H)
  if 2i > n then
    Terminate with H unchanged
  else if 2i < n then
    left=2i and right=2i+1
    j be index that minimizes O(1)
      key[H[left]] and key[H[right]]
  else if 2i = n then
    j=2i

  if key[H[j]] < key[H[i]] then
    swap array entries H[i] and H[j] O(1)
    Heapify-down(H, j)
  
```

Num swaps:  $O(\log n)$

Jan 26, 2018

CSCI211 - Sprenkle

16



## Implementing Priority Queues with Heaps

Operation	Description	Run Time
StartHeap(N)	Creates an empty heap that can hold N elements	
Insert(v)	Inserts item v into heap	
FindMin()	Identifies minimum element in heap but does not remove it	
Delete(i)	Deletes element in heap at position i	
ExtractMin()	Identifies and deletes an element with minimum key from heap	

Jan 26, 2018

CSCI211 - Sprenkle

17

## Implementing Priority Queues with Heaps

Operation	Description	Run Time
StartHeap(N)	Creates an empty heap that can hold N elements	$O(N)$
Insert(v)	Inserts item v into heap	$O(\log n)$
FindMin()	Identifies minimum element in heap but does not remove it	$O(1)$
Delete(i)	Deletes element in heap at position i	$O(\log n)$
ExtractMin()	Identifies and deletes an element with minimum key from heap	$O(\log n)$

Jan 26, 2018

CSCI211 - Sprenkle

18

## Putting It All Together...

1. Add elements into PQ with the number's value as its priority
2. Then extract the smallest number until done
  - Come out in sorted order

What is the running time of sorting numbers using a PQ implemented with a **heap**?

$O(n \log n)$

## Comparing Data Structures

Operation	Heap	Unsorted List	Sorted List
Start(N)		$O(1)$	$O(1)$
Insert(v)		$O(1)$	$O(n)$
FindMin()		$O(1)$	$O(1)$
Delete(i)		$O(n)$	$O(1)$
ExtractMin()		$O(n)$	$O(1)$

## Comparing Data Structures

Operation	Heap	Unsorted List	Sorted List
Start(N)	$O(N)$	$O(1)$	$O(1)$
Insert(v)	$O(\log n)$	$O(1)$	$O(n)$
FindMin()	$O(1)$	$O(1)$	$O(1)$
Delete(i)	$O(\log n)$	$O(n)$	$O(1)$
ExtractMin()	$O(\log n)$	$O(n)$	$O(1)$

Jan 26, 2018

CSCI211 - Sprenkle

21

## Additional Heap Operations

- Access elements in PQ by “name”

Key	2	4	5	6	9	20
Value	3542	5143	8712	1264	9123	5954

- Maintain additional array **Position** that stores current position of each element in heap



- Operations:

- Delete(Position[v])
    - Does not increase overall running time
  - ChangeKey(v,  $\alpha$ )
    - Changes key of element v to  $\alpha$
    - Identify position of element v in array (Position array)
    - Change key, heapify

Jan 26, 2018

CSCI211 - Sprenkle

22

# GRAPHS

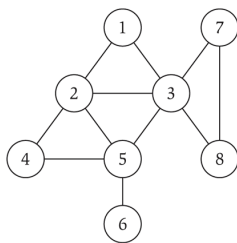
Jan 26, 2018

CSCI211 - Sprenkle

23

## Undirected Graphs $G = (V, E)$

- $V$  = nodes (vertices)
- $E$  = edges between pairs of nodes
- Captures pairwise relationship between objects
- Graph size parameters:  $n = |V|$ ,  $m = |E|$


 $V = \{ 1, 2, 3, 4, 5, 6, 7, 8 \}$ 
 $E = \{ 1-2, 1-3, 2-3, 2-4, 2-5, 3-5, 3-7, 3-8, 4-5, 5-6 \}$ 
 $n = 8$ 
 $m = 11$ 

Jan 26, 2018

CSCI211 - Sprenkle

24

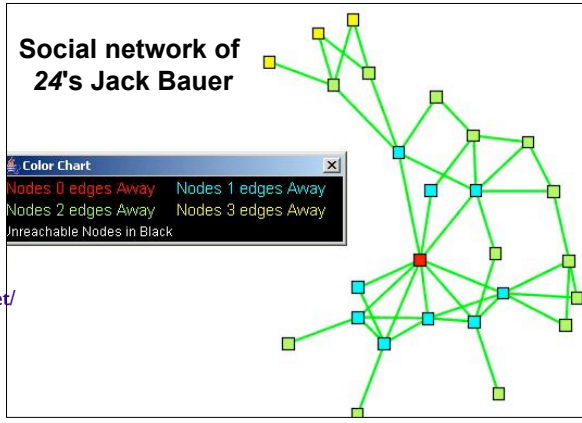
## Social Networks

- **Node:** people; **Edge:** relationship between 2 people
- *Everything Bad Is Good for You: How Today's Popular Culture Is Actually Making Us Smarter*

- Television shows have complex plots, complex social networks

<http://www.cs.duke.edu/csed/harambeenet/modules.html>

Jan 26, 2018



## Facebook: Visualizing Friends



<http://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919>

Jan 26, 2018

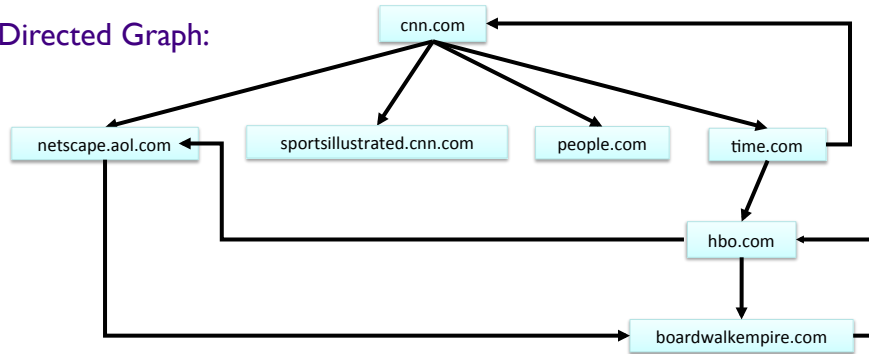
CSCI211 - Sprenkle

26

## World Wide Web

- Web graph
  - Node: web page
  - Edge: hyperlink from one page to another

Directed Graph:

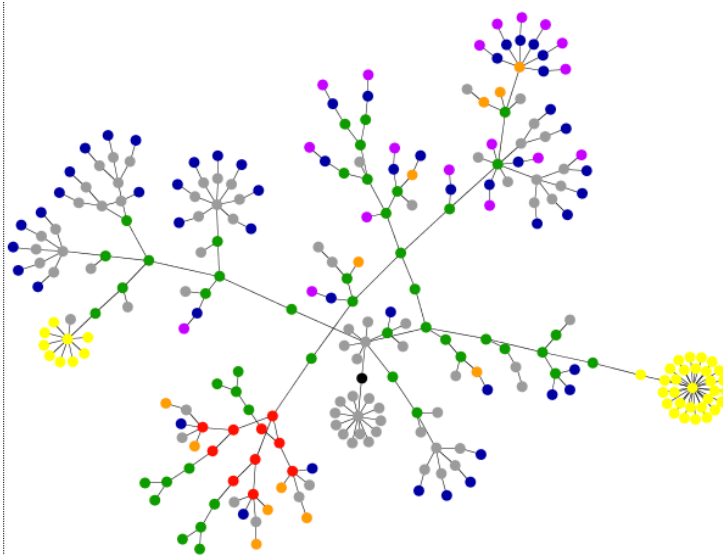


Jan 26, 2018

CSCI211 - Sprenkle

27

## Graph of Web Page www.wlu.edu



Jan 26, 2018

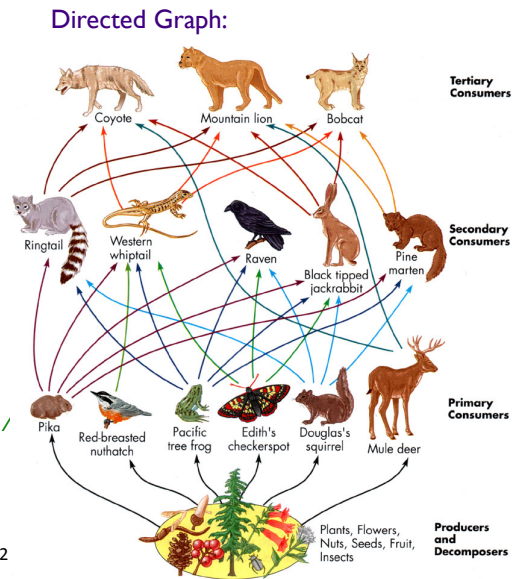
CSCI211 - Sprenkle

28

## Ecological Food Web

- Food web graph
  - Node = species
  - Edge = from prey to predator

Reference:  
<https://www.msu.edu/course/isb/202/ebertmay/images/foodweb.jpg>



Jan 26, 2018

CSCI2

## Graph Applications

Graph	Nodes	Edges
transportation	street intersections	highways
communication	computers	fiber optic cables
World Wide Web	web pages	hyperlinks
social	people	relationships
food web	species	predator-prey
software systems	functions	function calls
scheduling	tasks	precedence constraints
circuits	gates	wires

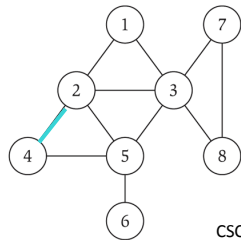
Jan 26, 2018

CSCI211 - Sprenkle

30

## Graph Representation: Adjacency Matrix

- $n \times n$  matrix with  $A_{uv} = 1$  if  $(u, v)$  is an edge
  - Two representations of each edge (symmetric matrix)
  - Space?
  - Checking if  $(u, v)$  is an edge?
  - Identifying all edges?



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

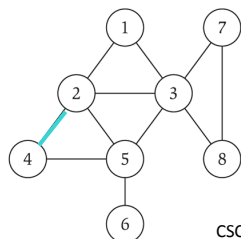
Jan 26, 2018

CSCI211 - Sprenkle

31

## Graph Representation: Adjacency Matrix

- $n \times n$  matrix with  $A_{uv} = 1$  if  $(u, v)$  is an edge
  - Two representations of each edge (symmetric matrix)
  - Space:  $\Theta(n^2)$
  - Checking if  $(u, v)$  is an edge:  $\Theta(1)$  time
  - Identifying all edges:  $\Theta(n^2)$  time



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Jan 26, 2018

CSCI211 - Sprenkle

32



## Looking Ahead

- Problem Set 3 due next Friday
- Wiki due Monday
  - 2.4, 2.5
  - 3.1