Week 9

p3b: final submission due before 10pm on Thursday, 3/28
p4: available soon
x4: due before 10pm on Monday 4/1
x5: meet with x-team coach for design review 4/1-4/8

Module: Week 9 (and start on week 10 before next week)

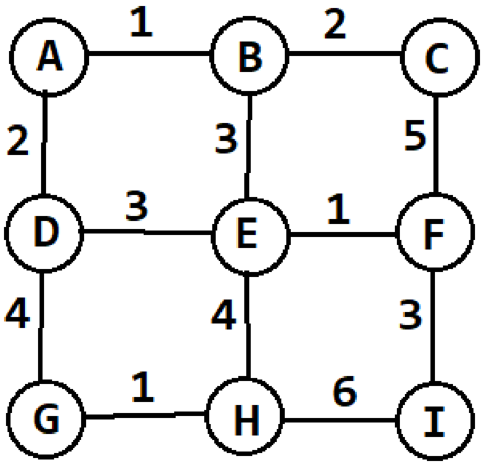
THIS WEEK:

- DFS and BFS Spanning Trees
- Minimum Spanning Trees
    - Prim's
    - Kruskal's
- Topological Ordering
- Dijkstra's Shortest Path algorithm
- Set operations

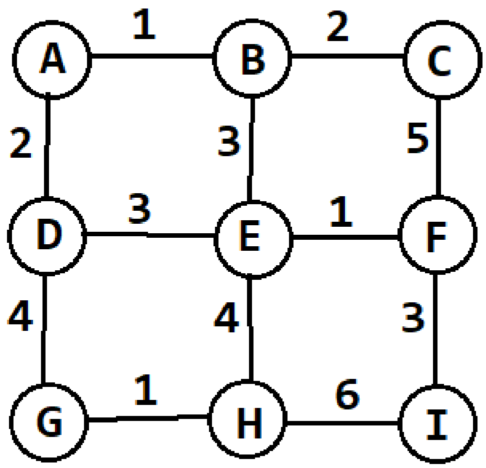
NEXT WEEK

- Linear Sorts (maybe next week)
    - radix
    - flashsort
- Java FX (in-class demo)
- Programming Project Assignment: Part 1, the Design
  (Start organizing your teams now – or keep your x-team for final Team Project)

Prim's Minimum Spanning Tree

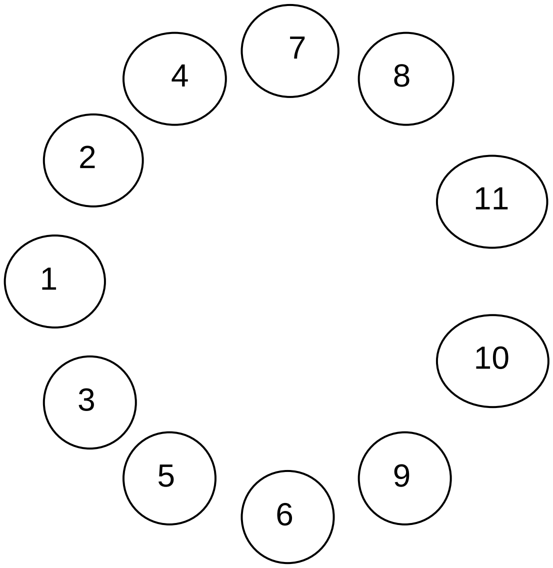


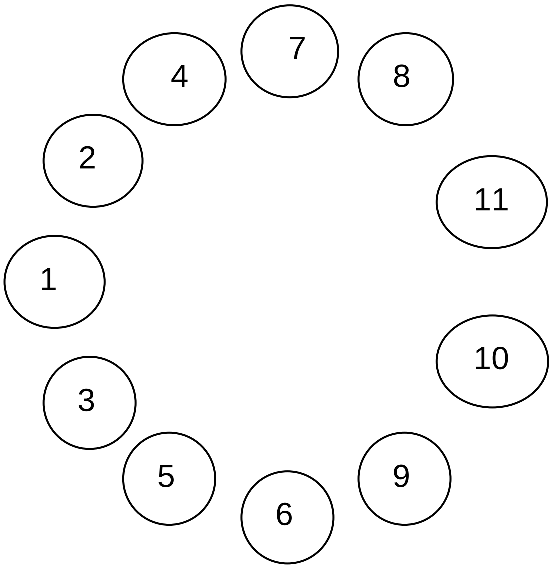
Kruskal's Minimum Spanning Tree



Topological Ordering

1. get bread
2. get jelly
3. get peanut butter
4. get butter knife
5. open jelly
6. open peanut butter
7. take bread slice 1
8. take bread slice 2
9. use knife to spread jelly on bread slice
10. use knife to spread peanut butter on bread slice
11. put slices together with spreaded sides facing each other



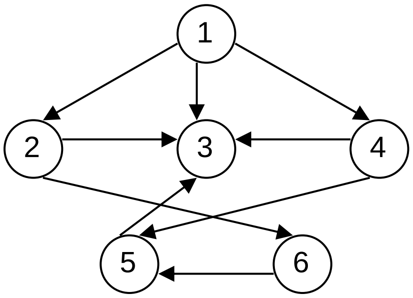


IDEA:

Topological Ordering

Iterative Algorithm (see readings for recursive algorithm)

Example



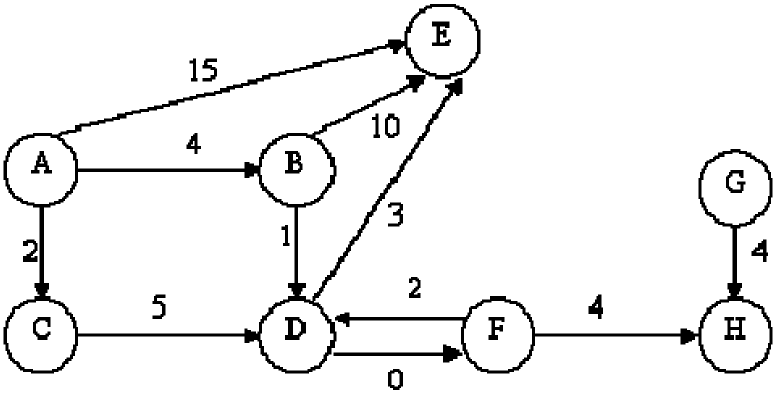
Dijkstra's Algorithm

-
   -
   -
   -
   -
   -
   -

Psuedo Code

```
for each vertex V
   initialize V’s visited mark to false
   initialize V's total weight to “infinity”
   initialize V's predecessor to null

set start vertex’s total weight to 0

create new priority queue pq
pq.insert( [start vertex total weight,start vertex] )

while !pq.isEmpty()
    [C’s total weight,C] = pq.removeMin()
   set C’s visited mark to true

   for each unvisited successor S adjacent to C
      if S's total weight can be reduced
         S's total weight = C's total weight + edge weight from C to S
         update S's predecessor to C
         pq.insert( [S’s total weight,S] )
         (if S already in pq we’ll just update S's total weight)
```

Dijkstra's Practice



Iteration	Priority Queue (just list smallest to largest)
0	
1	
2	
3	
4	
5	
6	
7	

Vertex	Visited	Total Weight	Predecessor
A			
B			
C			
D			
E			
F			
H			
G			

Reconstruct shortest path from A to F

Reconstruct shortest path from C to F

Sets

Definition

Examples:

Terminology

subset

proper subset

superset

proper superset

unit set

disjoint sets

Notation

https://www.rapidtables.com/math/symbols/Set_Symbols.html

$\emptyset$

$\mathbb{N}$

$\mathbb{R}$

$\mathbf{U}$

$\mathbf{A} \in \mathbf{letters}$

$|\mathbf{A}|$

$\mathbf{A} \subseteq \mathbf{B}$ $\mathbf{A} \subset \mathbf{B}$

$\mathbf{A} \nsubseteq \mathbf{B}$ $\mathbf{A} \not\subset \mathbf{B}$

$\mathbf{A} \supseteq \mathbf{B}$ $\mathbf{A} \supset \mathbf{B}$

$\mathbf{A} \nsupseteq \mathbf{B}$ $\mathbf{A} \not\supset \mathbf{B}$

$\mathbf{A}'$

Basic Set Operations

$$\mathbf{A} \cup \mathbf{B}$$

$$\mathbf{A} \cap \mathbf{B}$$

$$\mathbf{A} - \mathbf{B}$$

$$\mathbf{A} \setminus \mathbf{B}$$

$$\mathbf{A \Delta B} = (\mathbf{A} \setminus \mathbf{B}) \cup (\mathbf{B} \setminus \mathbf{A})$$

SetADT

Operations in java.util.Set

```
boolean add(E e) – add if item is not present
boolean contains(Object o) – true iff o is present
boolean remove(Object o) – remove o if present
boolean isEmpty() – true if no elements
int size() – returns number of elements
```

Implementation

Complexity analysis, if N is number of nodes

	insert	lookup	remove	iteration
Array				
Sorted array				
Linked list				
BST				
Balanced search tree				
Hash table				

De Morgan's Laws

If A and B are any two sets then:

```
        (A union B)'  = A' intersection B'
        (A intersection B)' = A' union B'
```

Java's Set is an interface