Chapter 4, Graphs:

- Concepts and implementation (**Today**)
- Traversal (next week Monday)
- Minimum spanning trees (next week Wednesday and Friday)
- Single-source shortest paths (Feb 22 and 24)

Today:

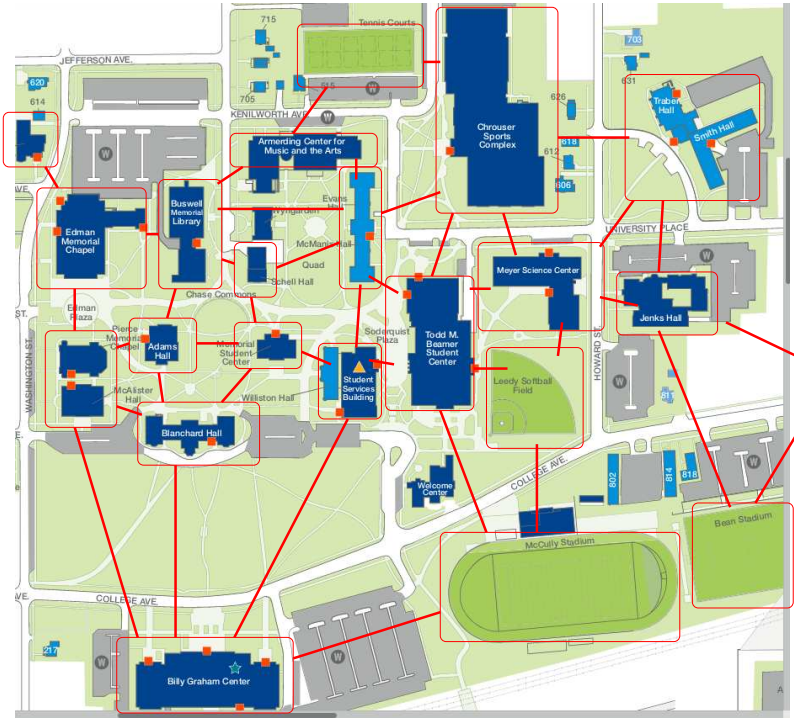
- Recent quiz questions
- Applications of graphs
- Vocabulary, taxonomy, and theory
- Representing and implementing graphs

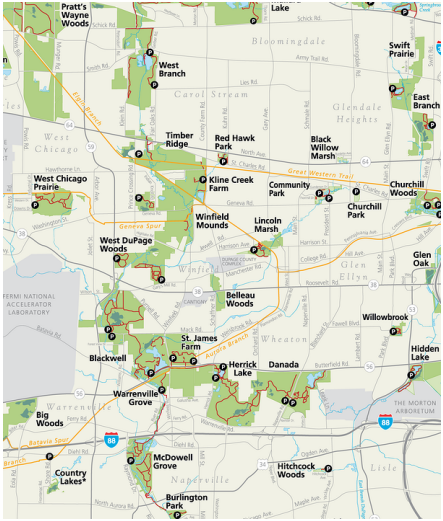
Indicate the worst case running time for each operation in each implementation of a priority queue.

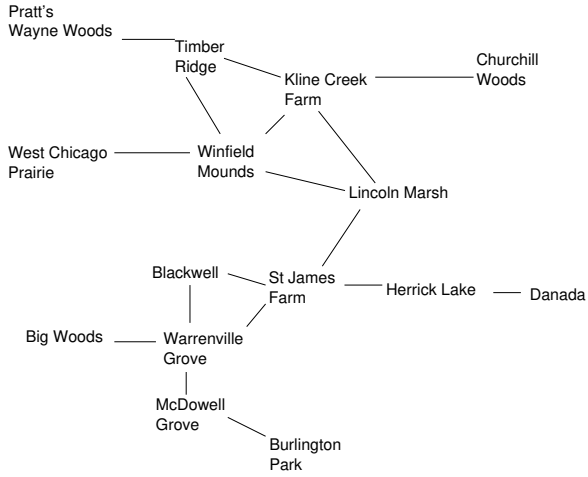
	ListPriorityQueue	SortedListPriorityQueue	HeapPriorityQueue
`insert()`	$\Theta(1)$	$\Theta(n)$	$\Theta(\lg n)$
`max()`	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$
`extractMax()`	$\Theta(n)$	$\Theta(1)$	$\Theta(\lg n)$
`contains()`	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$

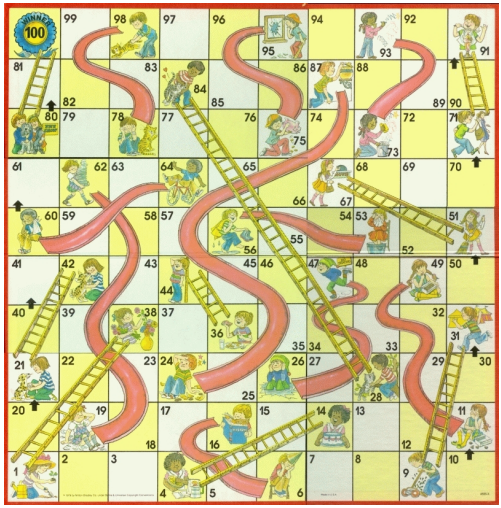
3.26 In the `NaiveNSet`, why does the `add()` method have an `@Override` annotation but `range()`, `complement()`, `union()`, `intersection()`, and `difference()` do not?

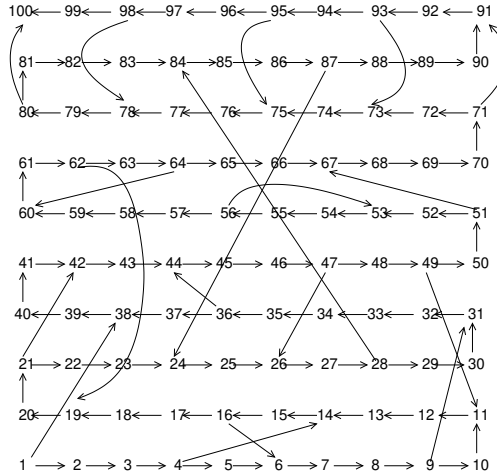
3.27 Explain the `+ 1` in the array creation `new byte[range / 8 + 1]` in the `BitVecNSet` constructor.

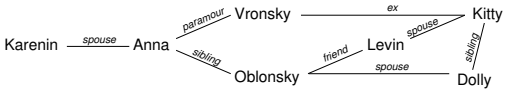

Tennis Courts
JEFFERSON AVE.
KENILWORTH AVE
Armerding Center for Music and the Arts
Chrouser Sports Complex
Trabert Hall
Smith Hall
Buswell Memorial Library
Edman Memorial Chapel
Evans Mall
McMinn Hall
Quad
Schell Hall
Chase Commons
Edman Plaza
Pierce Memorial Center
Adams Hall
Memorial Student Center
McAlister Hall
Williston Hall
Blanchard Hall
Soderquist Plaza
Student Services Building
Todd M. Beamer Student Center
Meyer Science Center
Jenks Hall
UNIVERSITY PLACE
HOWARD ST
WASHINGTON ST
Leedy Softball Field
Welcome Center
COLLEGE AVE
McCully Stadium
Bean Stadium
Billy Graham Center
East Campus

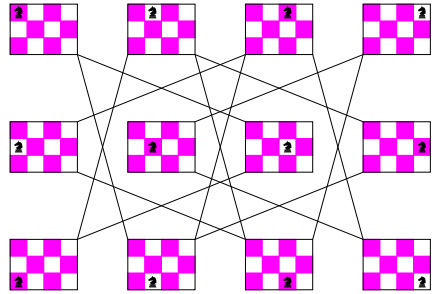

Pratt's Wayne Woods
Lake
Bloomingdale
Swift Prairie
West Branch
Army Trail Rd
Carol Stream
East Branch
Glendale Heights
Timber Ridge
Red Hawk Park
Black Willow Marsh
West Chicago Prairie
Kline Creek Farm
Community Park
Churchill Woods
Churchill Park
Great Western Trail
Winfield Mounds
Lincoln Marsh
West DuPage Woods
Geneva Spur
Glen Oak
Belleau Woods
Willowbrook
Hidden Lake
St. James Farm
Herrick Lake
Danada
Blackwell
Warrenville Grove
Big Woods
Illinois Prairie Path
McDowell Grove
Hitchcock Woods
Country Lakes
Burlington Park
Naperville
Lisle
Wheaton
Glen Ellyn
FERMI NATIONAL ACCELERATOR LABORATORY
THE MORTON ARBORETUM

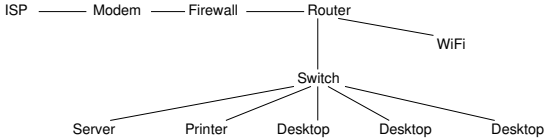

Pratt's Wayne Woods
Timber Ridge
Kline Creek Farm
Churchill Woods
West Chicago Prairie
Winfield Mounds
Lincoln Marsh
Blackwell
St James Farm
Herrick Lake
Danada
Big Woods
Warrenville Grove
McDowell Grove
Burlington Park



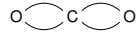

100 99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10



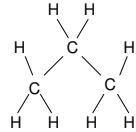




Water



Carbon dioxide



Propane

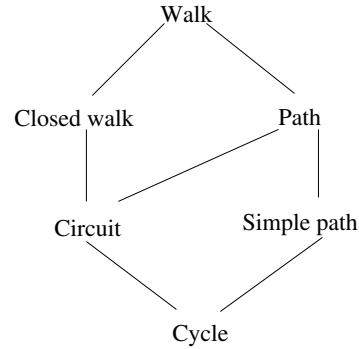
- Graph
- Vertex (compare *node*)
- Edge (compare *link*)
- Incident
- Adjacent
- Degree
- Complete
- Dense
- Sparse
- Directed graph
- Undirected graph
- Parallel edge
- Self loop
- Simple graph
- Weighted graph

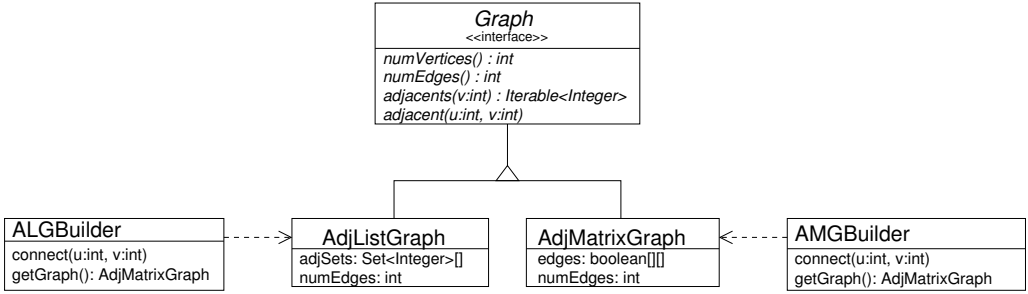
Adjectives

Trivial	Having only one vertex and no edges.
Simple	Having no repeated *vertices* (except, possibly, the initial and terminal).
Closed	Having the same vertex as initial and terminal.

Nouns

Walk	An alternating sequence of vertices and edges, each edge coming between its end points.
Path	A walk with no repeated *edge* (repeated vertices are ok).
Circuit	A closed path (no repeated edges, initial and terminal the same).
Cycle	A simple circuit (no repeated edges or vertices, except the initial and terminal, which are the same).




Graph
<<interface>>
numVertices() : int
numEdges() : int
adjacents(v:int) : Iterable<Integer>
adjacent(u:int, v:int)
ALGBuilder
connect(u:int, v:int)
getGraph(): AdjMatrixGraph
AdjListGraph
adjSets: Set<Integer>[]
numEdges: int
AdjMatrixGraph
edges: boolean[][]
numEdges: int
AMGBuilder
connect(u:int, v:int)
getGraph(): AdjMatrixGraph

	Adjacency matrix	Adjacency list
Space	$\Theta(V^2)$	$\Theta(V + E)$
`adjacent(u, v)`	$\Theta(1)$	$\Theta(deg(u))$ (expected case)
`getAdjacents(u)`	$\Theta(V)$	$\Theta(deg(u))$

Coming up:

Do **heaps and priority queue** *project (suggested by Mon, Feb 13)*
Do **bit vector and N-set** *project (suggested by Wed, Feb 15)*

Due **Wed, Feb 15** *(but spread it out):*
Read Section 4.(1–3)
Do Exercises 4.(26-29).
Take graph quiz