Data Structures & Algorithms

Who Are We?

- Ben Banavige
- Karthik Garimella
- ESDIS -> 586 -> Jupyter Notebooks for Earth Science

Karthenjamin

- https://github.com/karthenjamin

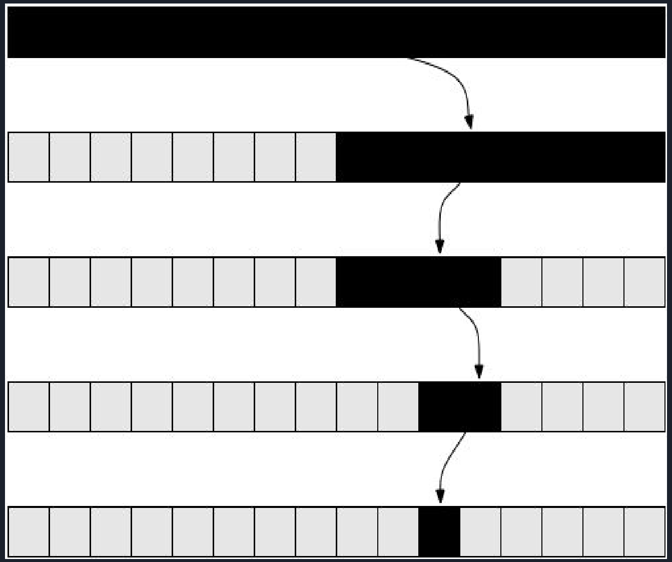
Outline

- Big O notation
- Graph Theory
- Dijkstra’s Shortest Paths Algorithm
    - Different implementation -> Effect on Big O

Big O Notation

- Performance of an algorithm

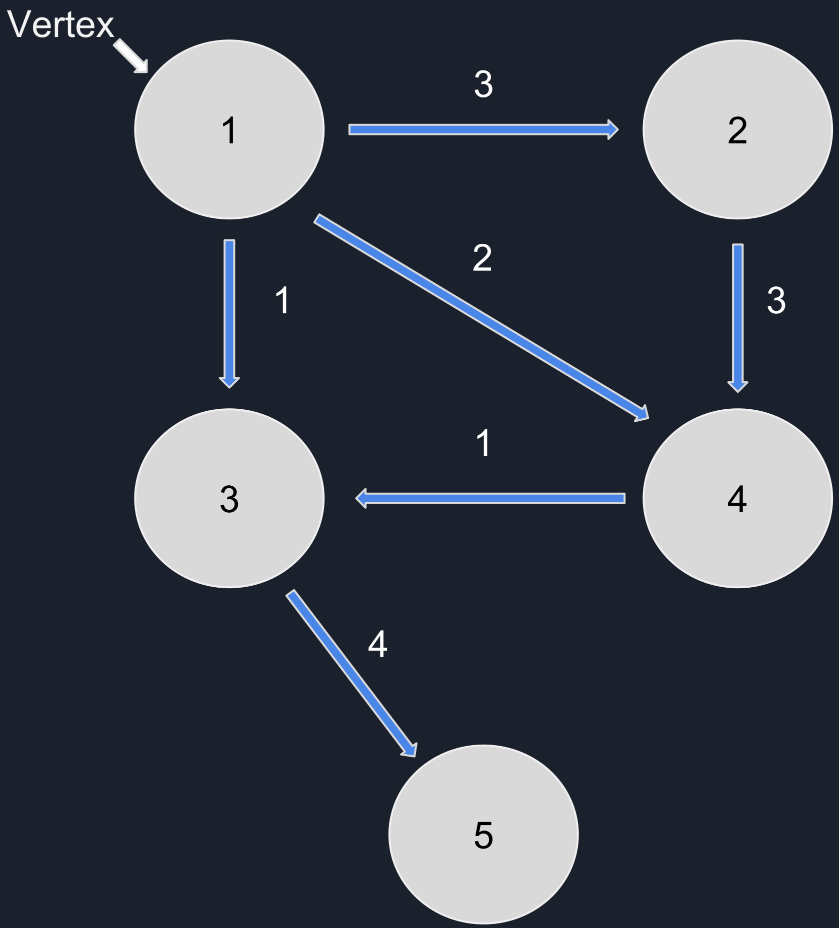
$O(n)$



$O(\log(n))$

Graph Theory

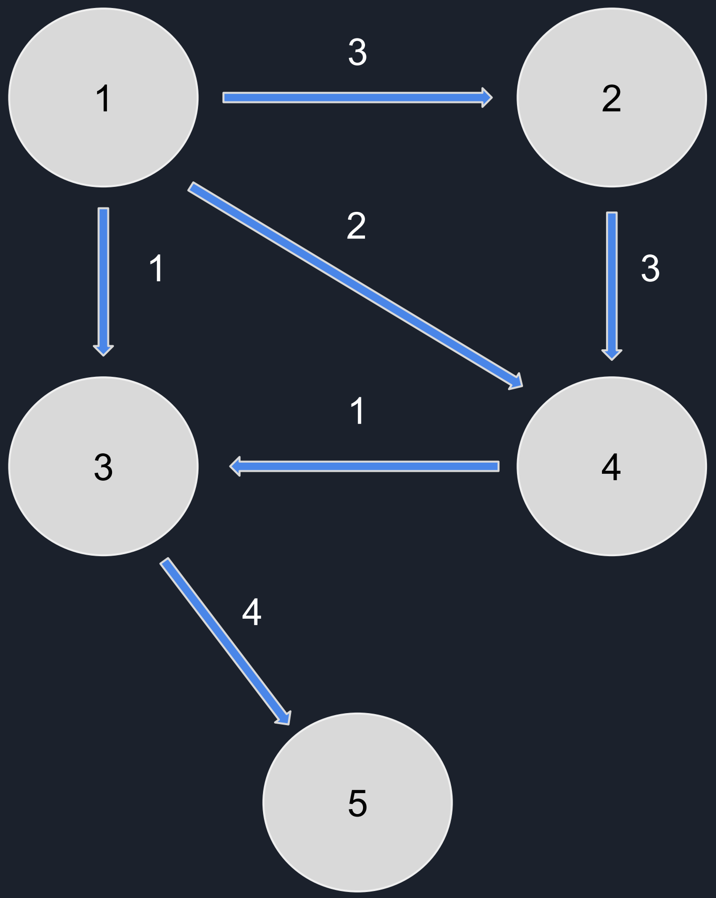
- Vertices
- Weighted Edges
- Representation
  - Adjacency Matrix
  - Adjacency List



Adjacency Matrix

- *A* is an adjacency matrix for graph *G*
- *A*[*i*,*j*] is weight of edge from vertex *i* to vertex *j*

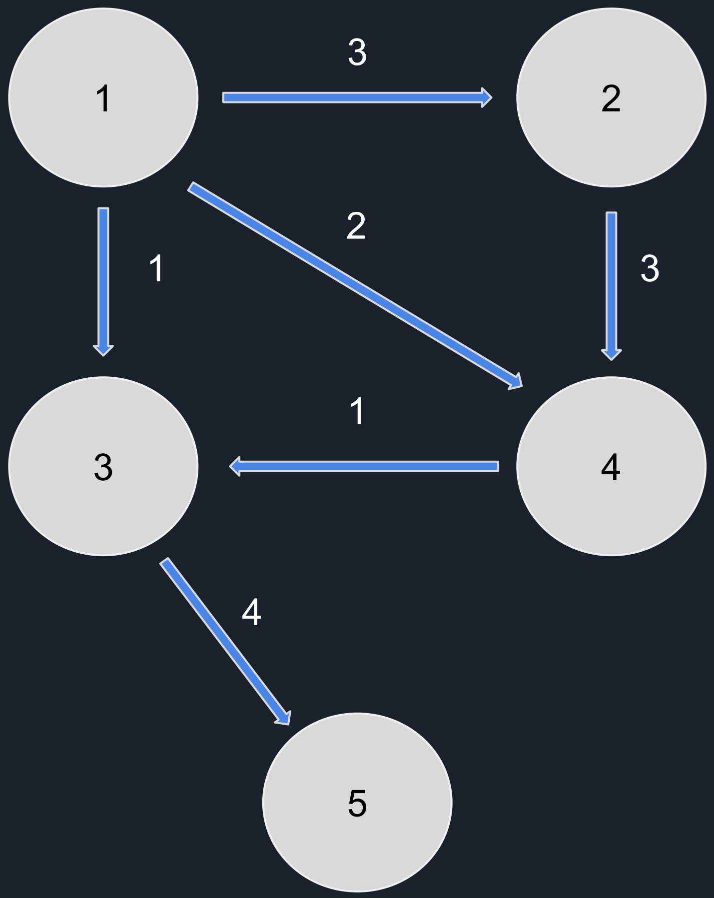
	1	2	3	4	5
1	∞	3	1	2	∞
2	∞	∞	∞	3	∞
3	∞	∞	∞	∞	4
4	∞	∞	1	∞	∞
5	∞	∞	∞	∞	∞



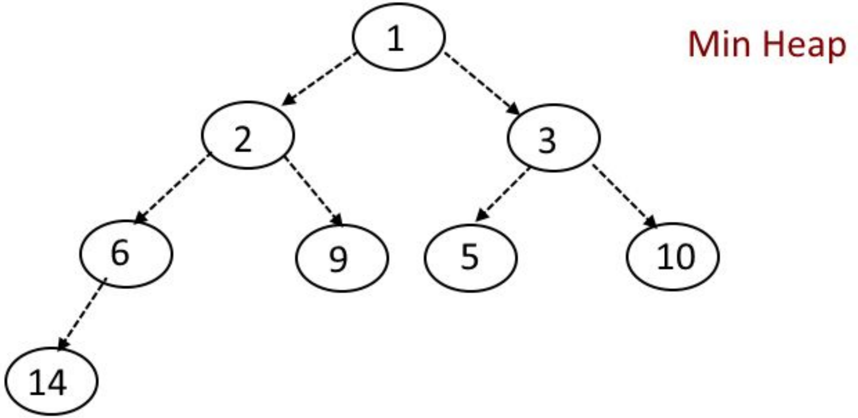
Adjacency List

- *A* is an adjacency list for graph *G*
- (c,j) is in $A[i]$ if $w_{i,j} = c$

1 →	(3, 2)	(1,3)	(2,4)
2 →	(3, 4)		
3 →	(4,5)		
4 →	(1,3)		
5 →			



Heap



0	1	2	3	4	5	6	7	8
	1	2	3	6	9	5	10	14

for Node at i : Left child will be 2i and right child will be at 2i+1 and parent node will be at [i/2].

Dijkstra's Algorithm - Shortest Path

```
Require: Weighted graph G = (V, E), vertex s ∈ V, vertex t ∈ V .
1: Q ← minHeap
2: Q.insert(0, s)
3: seen ← []
4: D[s] ← 0, all other D[v] ← ∞
4: while Q is not empty do
    5: cost,u ← Q.removeMin
    6: if u not in seen seen.append(u)
        7. If u == t then return cost
        8: for each outgoing edge (u, v) from u do
            9. If cost > 0 and D[v] > D[u] + c_uv
                10. D[v] =  D[u] + c_uv
                11. Q.append(D[v], v).
            12: end if
        13: end for
    14: end if
13: end while
14: return D[t]
```

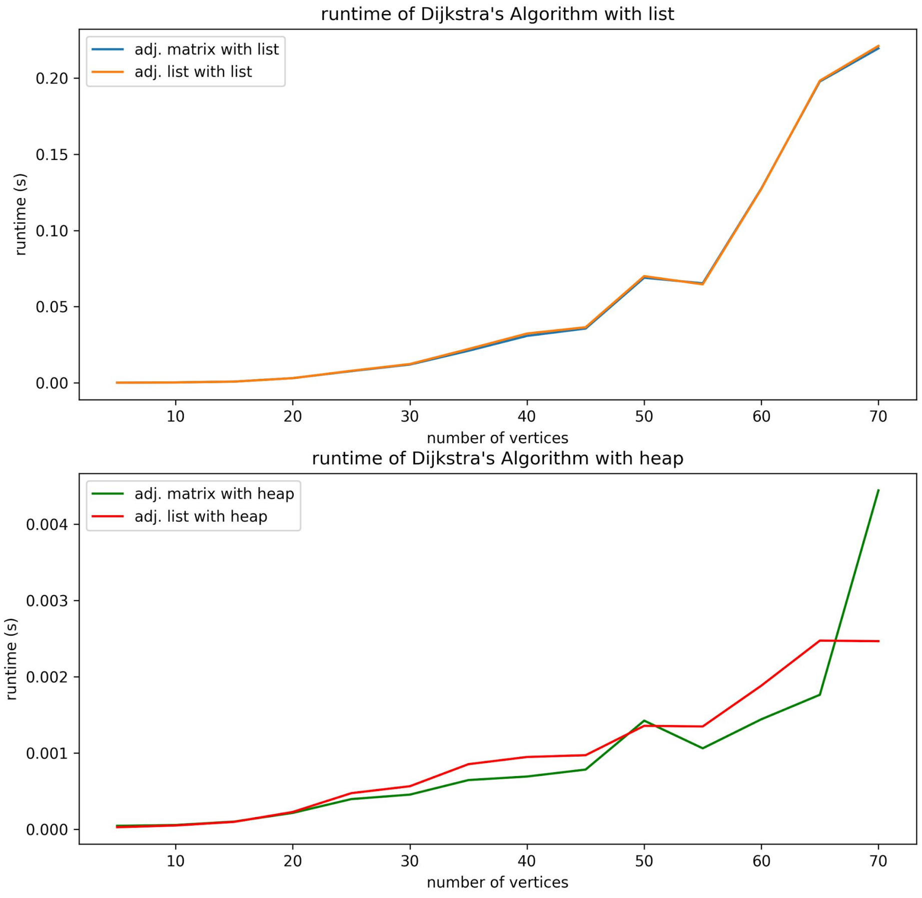
Runtime

- n = # of vertices
- m = # of edges
- What takes time?
    - Popping from Queue
        - At most n times
    - Inserting into Queue
        - At most m times
- Runtime: $O(n * T_{pop} + m * T_{insert}) = O(n \log n + m \log n) = O(m \log n)$ if $m > n$

https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html

Analyzing Data Structures

- Data Structures change runtimes of algorithms
- Heap vs. List
- Adjacency Matrix vs. Adjacency List


runtime of Dijkstra's Algorithm with list
adj. matrix with list
adj. list with list
runtime (s)
0.00
0.05
0.10
0.15
0.20
10
20
30
40
50
60
70
number of vertices
runtime of Dijkstra's Algorithm with heap
adj. matrix with heap
adj. list with heap
runtime (s)
0.000
0.001
0.002
0.003
0.004
10
20
30
40
50
60
70
number of vertices

Takeaways

- Data structures change efficiency
- Algorithmic thinking helps solve complex problems

Questions?

Negative Weights