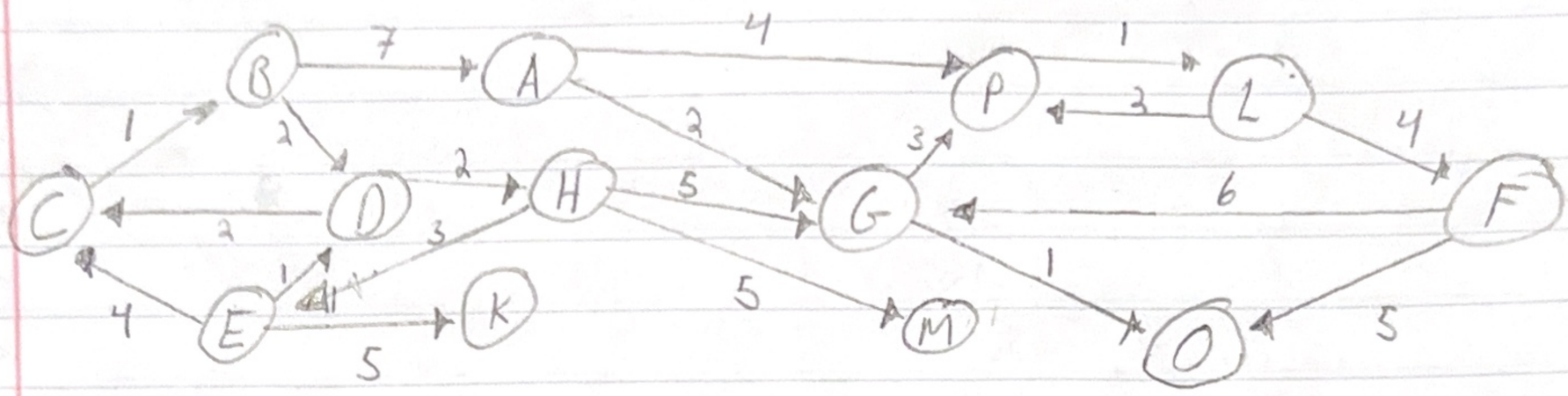
Jeremiah McDonald
Intro to Algorithms
25 April 2025

1. Dijkstra: Find the shortest path from E to all other nodes using Dijkstras Algorithm.



Step	Current	Visited SET	Updates
0	Node with lowest	[]	Initialize All distances to ∞, E = 0
1	E	[E]	C = 4, O = 1, K = 5 ; P[D]=E, P[C]=E, P[K]=E
2	D	[E, D]	H = 3, C = 3 update! ; P[C]=D, P[H]=D
3	H	[E, D, H]	G = 8, M = 8, E >No update P[M]=H, P[G]=H
4	C	[E, D, H, C]	B = 4 ; P[B]=C
5	B	[E, D, H, C, B]	A = 9, O >No update ; P[A]=B
6	K	[E, D, H, C, B, K]	
7	M	[E, D, H, C, B, K, M]	
8	G	[E, D, H, C, B, K, M, G]	O = 9, P = 11 P[O]=G, P[P]=G
9	A	[E, D, H, C, B, K, M, G, A]	P >No Update, G >No Update
10	O	[E, D, H, C, B, K, M, G, A, O]	
11	P	[E, D, H, C, B, K, M, G, A, O, P]	L = 12 P[L]=P
12	L	[E, D, H, C, B, K, M, G, A, O, P, L]	P >No update, F = 16 P[F]=L
13	F	[E, D, H, C, B, K, M, G, A, O, P, L, F]	

Final Shortest Path: E, O, H, G, P, L, F = 16

Priority Queue at each step: PQ = [E], PQ = [D, C, K],
PQ = [H, C, K], PQ = [C, K, G, M], PQ = [B, K, G, M],
PQ = [K, G, M, A], PQ = [G, M, A], PQ = [G, A],
PQ = [A, O, P], PQ = [O, P], PQ = [P],
PQ = [L], PQ = [F]

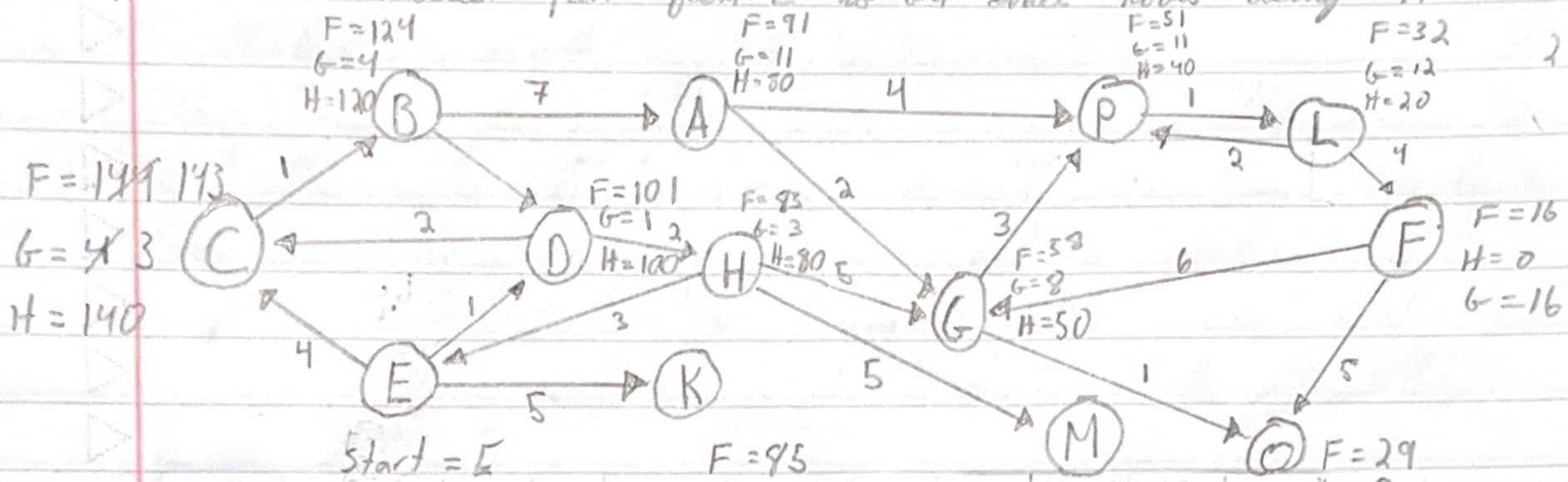
Node	Prev
F	L
L	P
P	G
O	G
G	H
M	H
A	B
B	C
C	D
H	D
D	E

[E, null]

Graphs

Jeremiah McDonald
Intro to Algorithms
25 April 2025

2. Find the shortest path from E to all other nodes using A^*



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

X_1 = current node, X_2 = max

$H = X_2 - X_1$, F = total cost (G+H) , G = cost from start

Priority Queue: PQ=[E] , PQ=[K, D, C] , PQ=[D, C] , PQ=[H, C], PQ=[M, G, C] , PQ=[G, C] , PQ=[O, P, C] , PQ=[P, C], PQ=[L, C] , PQ=[F, C] , PQ=[C] , PQ=[B] , PQ=[A]

Visited: [E, K, D, H, M, G, O, P, L, F, C, B, A]

3. Which algorithm found the shortest path to H in less iterations Dijkstra or A^* Algorithm In this graph to find the shortest path to every node is the exact same for both algorithms. This is because your must visit every node in the graph in both cases which takes the same amount of time.

If we were looking at which one found H fastest, because I chose a poor heuristic Dijkstra actually was faster (by one iteration)

Jeremiah McDonald
Intro to Algorithms
25 April 2025

7. Matrix represents - directed graph.
This program takes input n x n. The inner loop runs n times
and the outer loop runs n times. ∴ Time Complexity = $O(n^2)$
Space Complexity: input is given so no additional
space needed for matrix. Only used variables i + j inside loop ∴
∴ Space Complexity = $O(n^2)$

Every Path:
Time Complexity: n = number of vertices, m = number of edges
In DFS, you could search all neighbors (edges). Prevent cycles but
number of simple paths could be exponential. 1→2, 1→3, 1→4, 1→5
☆ Time Complexity = 2^n ☆ Because at each vertex there are 2 choices: include path or not
Space Complexity: Visited set = O(n), path list = O(n), Stack = O(n)
graph = O(n+m) ⇒ O(n+n+n+n x m) = O(4n x m)
☆ O(n x n) ☆

Draw Graph: Time Complexity: O(n) where n is the number of vertices
Space Complexity: O(n) because we require n spaces in memory