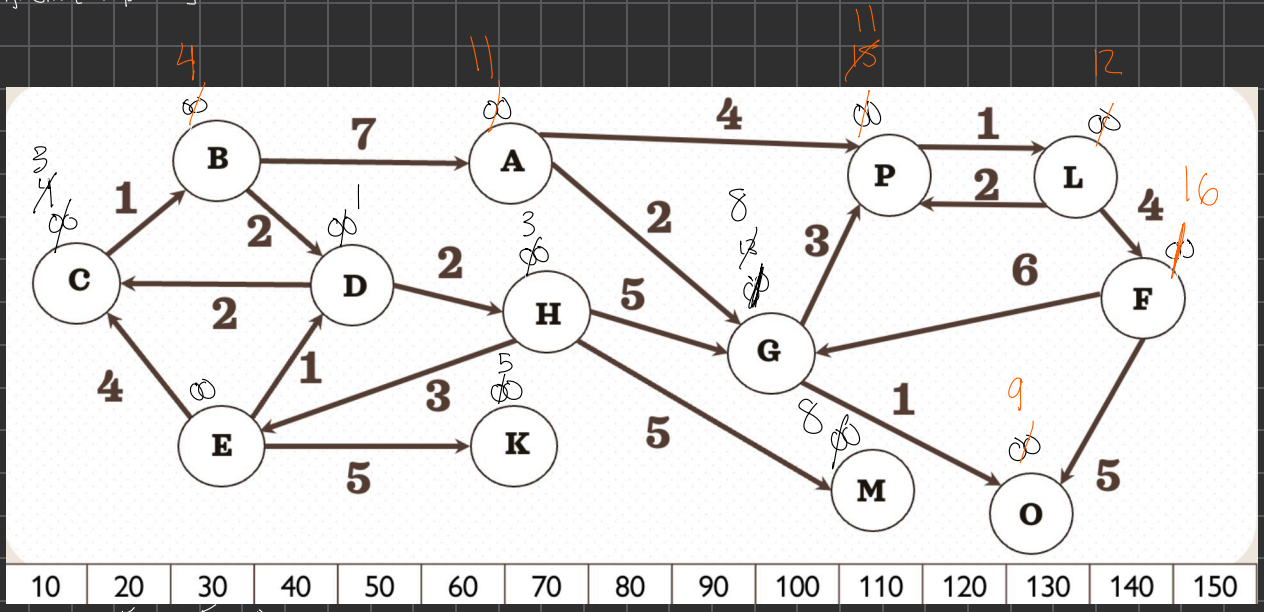
Assignment 7 - Graphs

Due: April 23 2025

Jaydin Cruz

1. (text) Dijkstra [15 points]



Visited: [ ]

[E] E → E : 0
E → C : 4
E → K : 5
E → D : 1

* set all to ∞
* go to cheapest cost

[ED]
ED → H : 3
ED → C : 3

[EDC]
EDC → B : 4

[EDCB]
EDCB → A : 11

[EDH]
EDH → G : 8
EDH → M : 8

[EDCBA]
EDCBA → P : 15
EDCBA → G : 8

[EDHG]
EDHG → P : 11
EDHG → O : 9

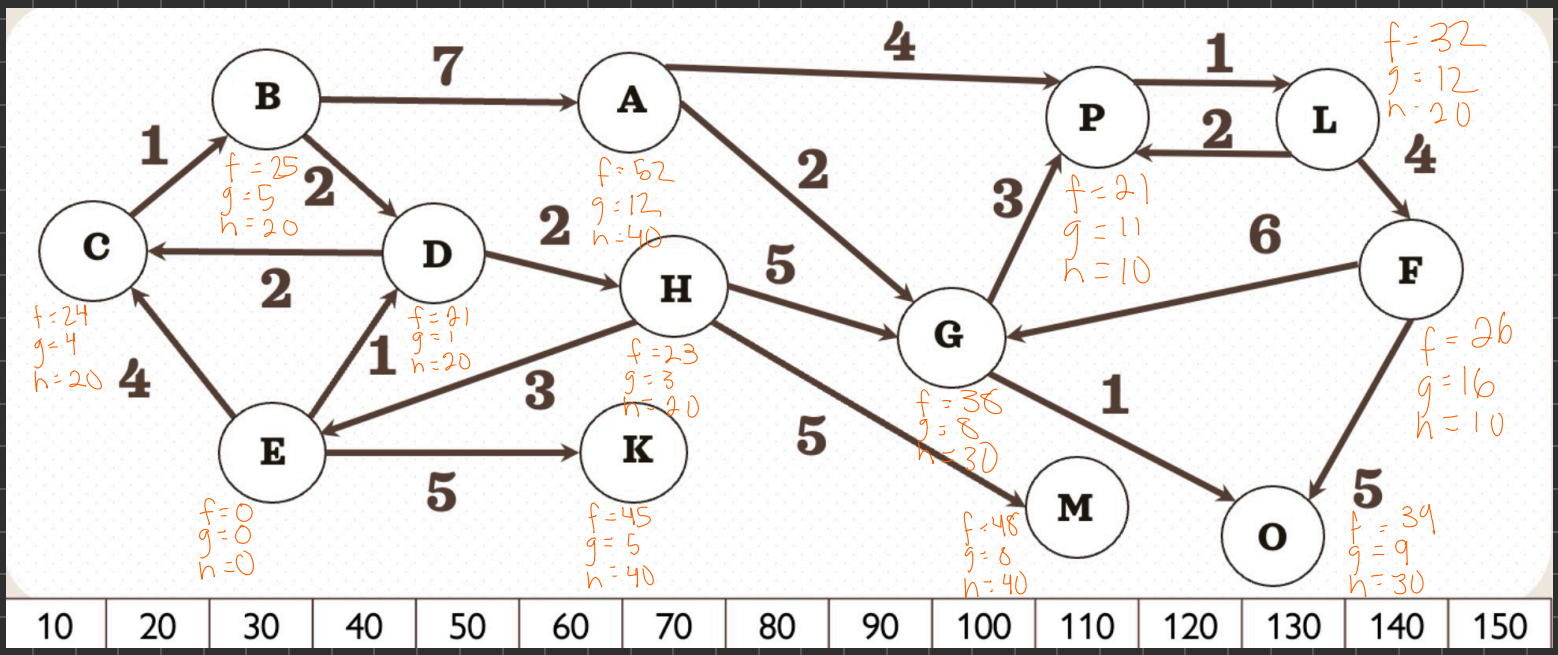
[EDHGO]
EDHGO : deadend

[EDHGP]
EDHGP → L : 12

[EDHGPL]
EDHGPL → F : 16

Shortest Path
E → E : 0
E → C : 3 (E, D, C)
E → K : 5
E → D : 1
E → H : 3 (E, D, H)
E → B : 4 (E, D, C, B)
E → G : 8 (E, D, H, G)
E → M : 8 (E, D, H, M)
E → P : 11 (E, D, H, G, P)
E → O : 9 (E, D, H, G, O)
E → L : 12 (E, D, H, E P)
E → F : 16 (EDH E P L, F)
E → A : 11 (EDCB, A)

2. (text) A* [20 points]



h(n) = Abs. of distance

ex. E, is at 70 to go to C we go back 20, so cost is 20

3. (text) Comparison [5 points] (E → H)

Both Dijkstra and A* found node "H" from E in five iterations. Both Algorithms took the same # of iterations, both performed the same in terms of iterations to find shortest path to H.

7. (text) Algorithms Analysis [10 points]

is GraphDirected(Graph[][] matrix, n)

square matrix

Time Complexity: $O(n^2)$ where n is the number of vertices in the graph input is nxn. method check each pair $M_{[i,j]}$ in matrix, making n^2 comparisons in the worst case. Furthermore,

→ total comp $\sum_{i=1}^{n} i = \frac{n \cdot (n)}{2} = \frac{n^2}{2}$

→ $O(\frac{n^2}{2}) \rightarrow O(n^2)$

→ Two nested loops: outer: n times (every row)
inner: n time (every colum in row)
↳ it square so same..

Space Complexity: $O(n^2)$, where n is # of vertices bc → matrix input
↳ no extra space used expect loop variable; → const space

Continue on Next Page!

EveryPath

Time Complexity: $O(2^n)$, n = # of vertices) n = # vertices, E = # of edges DFS used to explore all possible path -> each vertex we visit up to n vertices worst case U to W

$O(2^n)$ bc each vertex leads to other neighbor, in the worst DFS can be exponential bc its recursive

Space Complexity: $O(n+e)$, n = # of vertices, e = number of edges

↳ path list -> $O(n)$ up to n vertices
↳ visited set -> $O(n)$ up to n vertices
call stack

$O(n)$ for path + visited + recursion

DrawGraph

Time Complexity: $O(n^2)$ where n is # of vertices -> bc it prints $n \times n$ (square) adjacency matrix which grows faster than the $O(n)$ time for parsing + building matrix $O(n^2 + n) \rightarrow O(n^2)$

Space Complexity: $O(n^2)$ bc of the storage required for the adjacency matrix

Node list $O(n)$ slower than adjacency matrix