# Q3. [9 pts] Search

Suppose we have a connected graph with $N$ nodes, where $N$ is finite but large. Assume that every node in the graph has exactly $D$ neighbors. **All edges are undirected**. We have exactly one start node, $S$, and exactly one goal node, $G$.

Suppose we know that the shortest path in the graph from $S$ to $G$ has length $L$. That is, it takes at least $L$ edge-traversals to get from $S$ to $G$ or from $G$ to $S$ (and perhaps there are other, longer paths).

We'll consider various algorithms for searching for paths from $S$ to $G$.

**(a)** [2 pts] Uninformed Search
Using the information above, give the tightest possible bounds, using big $\mathcal{O}$ notation, on **both the absolute best case and the absolute worst case number of node expansions** for each algorithm. Your answer should be a function in terms of variables from the set $\{N, D, L\}$. You may not need to use every variable.

**(i)** [1 pt] DFS **Graph** Search

Best case: ____________________ Worst case: ____________________

**(ii)** [1 pt] BFS **Tree** Search

Best case: ____________________ Worst case: ____________________

**(b)** [2 pts] Bidirectional Search
Notice that because the graph is undirected, finding a path from $S$ to $G$ is equivalent to finding a path from $G$ to $S$, since reversing a path gives us a path from the other direction of the same length.

This fact inspired **bidirectional search**. As the name implies, bidirectional search consists of two simultaneous searches which both use the same algorithm; one from $S$ towards $G$, and another from $G$ towards $S$. When these searches meet in the middle, they can construct a path from $S$ to $G$.

More concretely, in bidirectional search:

- We start Search 1 from $S$ and Search 2 from $G$.
- The searches take turns popping nodes off of their separate fringes. First Search 1 expands a node, then Search 2 expands a node, then Search 1 again, etc.
- This continues until one of the searches expands some node $X$ which the other search has also expanded.
- At that point, Search 1 knows a path from $S$ to $X$, and Search 2 knows a path from $G$ to $X$, which provides us with a path from $X$ to $G$. We concatenate those two paths and return our path from $S$ to $G$.

Don't stress about further implementation details here!

Repeat part (a) with the bidirectional versions of the algorithms from before. Give the tightest possible bounds, using big $\mathcal{O}$ notation, on both the absolute best and worst case number of node expansions by the bidirectional search algorithm. Your bound should still be a function of variables from the set $\{N, D, L\}$.

**(i)** [1 pt] Bidirectional DFS **Graph** Search

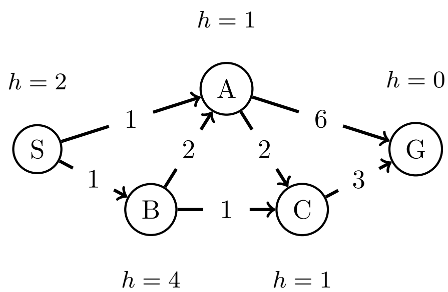
Best case: ____________________ Worst case: ____________________

**(ii)** [1 pt] Bidirectional BFS **Tree** Search

Best case: ____________________ Worst case: ____________________

In parts (c)-(e) below, consider the following graph, with start state $S$ and goal state $G$. Edge costs are labeled on the edges, and heuristic values are given by the $h$ values next to each state.

In the search procedures below, break any ties alphabetically, so that if nodes on your fringe are tied in values, the state that comes first alphabetically is expanded first.




**(c)** [1 pt] Greedy Graph Search

What is the path returned by greedy graph search, using the given heuristic?

- ○ $S \to A \to G$
- ○ $S \to A \to C \to G$
- ○ $S \to B \to A \to C \to G$
- ○ $S \to B \to A \to G$
- ○ $S \to B \to C \to G$

**(d)** A* Graph Search

**(i)** [1 pt] List the nodes in the order they are expanded by A* graph search:

Order: ____________________

**(ii)** [1 pt] What is the path returned by A* graph search?

- ○ $S \to A \to G$
- ○ $S \to A \to C \to G$
- ○ $S \to B \to A \to C \to G$
- ○ $S \to B \to A \to G$
- ○ $S \to B \to C \to G$

**(e)** Heuristic Properties

**(i)** [1 pt] Is this heuristic *admissible*? If so, mark *Already admissible*. If not, find a minimal set of nodes that would need to have their values changed to make the heuristic admissible, and mark them below.

○ Already admissible

☐ Change $h(S)$ ☐ Change $h(A)$ ☐ Change $h(B)$

☐ Change $h(C)$ ☐ Change $h(D)$ ☐ Change $h(G)$

**(ii)** [1 pt] Is this heuristic *consistent*? If so, mark *Already consistent*. If not, find the minimal set of nodes that would need to have their values changed to make the heuristic consistent, and mark them below.

○ Already consistent

☐ Change $h(S)$ ☐ Change $h(A)$ ☐ Change $h(B)$

☐ Change $h(C)$ ☐ Change $h(D)$ ☐ Change $h(G)$

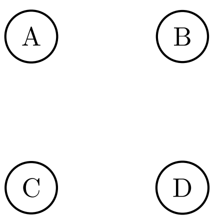
# Q4. [8 pts] CSPs

Four people, A, B, C, and D, are all looking to rent space in an apartment building. There are three floors in the building, 1, 2, and 3 (where 1 is the lowest floor and 3 is the highest). Each person must be assigned to some floor, but it's ok if more than one person is living on a floor. We have the following constraints on assignments:

- $A$ and $B$ must not live together on the same floor.
- If $A$ and $C$ live on *the same* floor, they must both be living on floor 2.
- If $A$ and $C$ live on *different* floors, one of them must be living on floor 3.
- $D$ must not live on the same floor as anyone else.
- $D$ must live on a higher floor than $C$.

We will formulate this as a CSP, where each person has a variable and the variable values are floors.

**(a)** [1 pt] Draw the edges for the constraint graph representing this problem. Use binary constraints only. You do not need to label the edges.




**(b)** [2 pts] Suppose we have assigned C = 2. Apply forward checking to the CSP, filling in the boxes next to the values for each variable that are eliminated:

| | | | |
|---|---|---|---|
| A | ☐ 1 | ☐ 2 | ☐ 3 |
| B | ☐ 1 | ☐ 2 | ☐ 3 |
| C | | ☐ 2 | |
| D | ☐ 1 | ☐ 2 | ☐ 3 |

**(c)** [3 pts] Starting from the original CSP with full domains (i.e. without assigning any variables or doing the forward checking in the previous part), enforce arc consistency for the entire CSP graph, filling in the boxes next to the values that are eliminated for each variable:

| | | | |
|---|---|---|---|
| A | ☐ 1 | ☐ 2 | ☐ 3 |
| B | ☐ 1 | ☐ 2 | ☐ 3 |
| C | ☐ 1 | ☐ 2 | ☐ 3 |
| D | ☐ 1 | ☐ 2 | ☐ 3 |

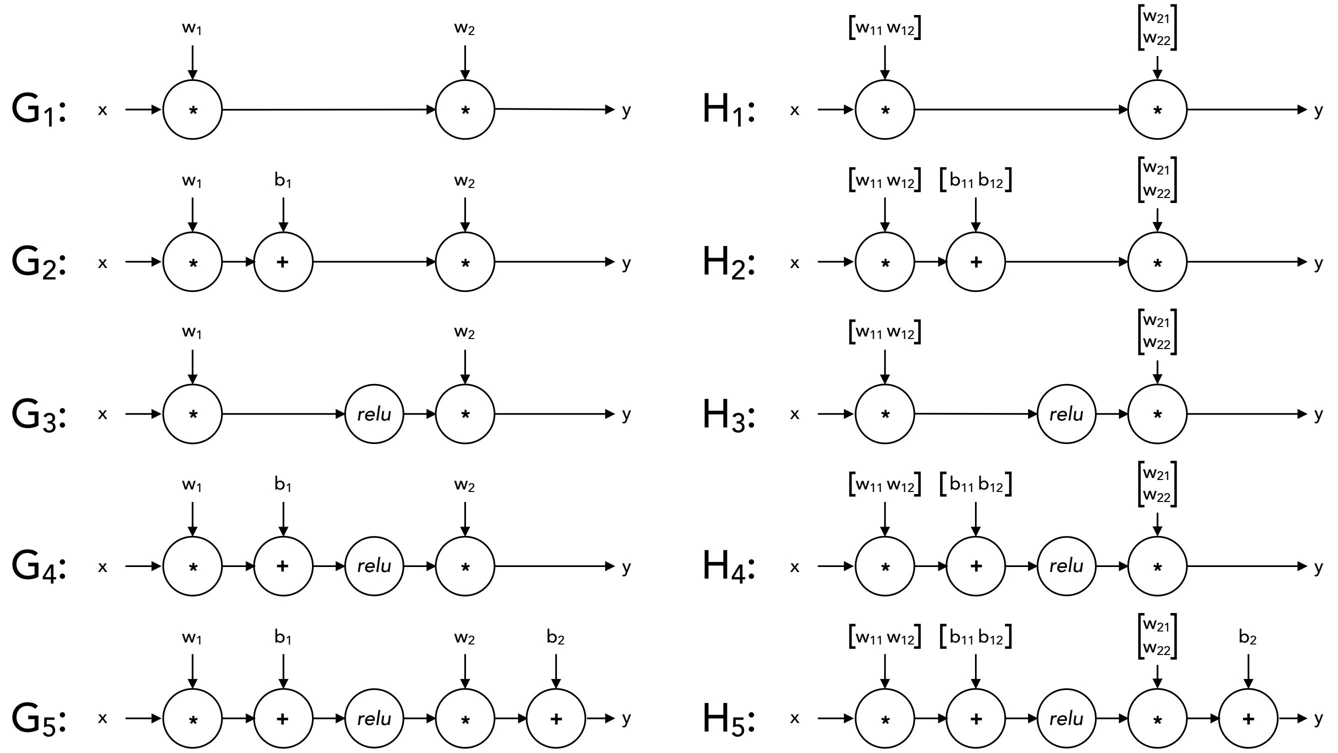
**(d)** [2 pts] Suppose that we were running local search with the min-conflicts algorithm for this CSP, and currently have the following variable assignments.

| | |
|---|---|
| A | 3 |
| B | 1 |
| C | 2 |
| D | 3 |

Which variable would be reassigned, and which value would it be reassigned to? Assume that any ties are broken alphabetically for variables and in numerical order for values.

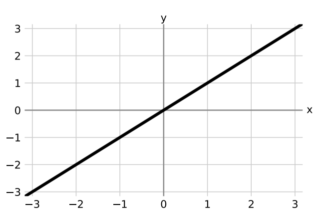
The variable ○ A will be assigned the new value ○ 1
○ B ○ 2
○ C ○ 3
○ D

# Q10. [15 pts] Neural Networks: Representation

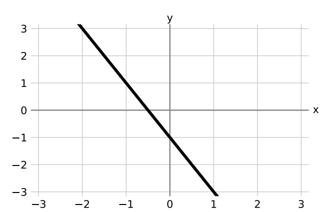



For each of the piecewise-linear functions below, mark all networks from the list above that can represent the function **exactly** on the range $x \in (-\infty, \infty)$. In the networks above, $relu$ denotes the element-wise ReLU nonlinearity: $relu(z) = max(0, z)$. The networks $G_i$ use 1-dimensional layers, while the networks $H_i$ have some 2-dimensional intermediate layers.

**(a)** [5 pts]

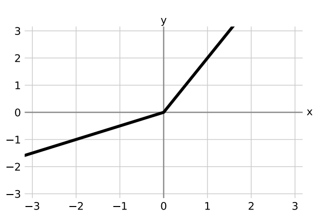


☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above

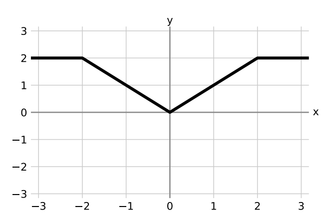


☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above

**(b)** [5 pts]

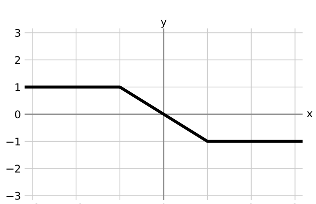


☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above



☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above

**(c)** [5 pts]



☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above

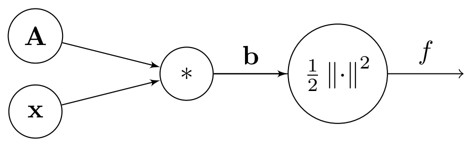
☐ $G_1$ ☐ $H_1$
☐ $G_2$ ☐ $H_2$
☐ $G_3$ ☐ $H_3$
☐ $G_4$ ☐ $H_4$
☐ $G_5$ ☐ $H_5$
○ None of the above

# Q11. [9 pts] Backpropagation

In this question we will perform the backward pass algorithm on the formula

$$f = \frac{1}{2}\left\|\mathbf{Ax}\right\|^2$$

Here, $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{b} = \mathbf{Ax} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, and $f = \frac{1}{2}\left\|\mathbf{b}\right\|^2 = \frac{1}{2}\left(b_1^2 + b_2^2\right)$ is a scalar.




**(a)** [1 pt] Calculate the following partial derivatives of $f$.

**(i)** [1 pt] Find $\frac{\partial f}{\partial \mathbf{b}} = \begin{bmatrix} \frac{\partial f}{\partial b_1} \\ \frac{\partial f}{\partial b_2} \end{bmatrix}$.

○ $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ○ $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ ○ $\begin{bmatrix} b_2 \\ b_1 \end{bmatrix}$ ○ $\begin{bmatrix} f(b_1) \\ f(b_2) \end{bmatrix}$ ○ $\begin{bmatrix} A_{11} \\ A_{22} \end{bmatrix}$ ○ $\begin{bmatrix} b_1 + b_2 \\ b_1 - b_2 \end{bmatrix}$

**(b)** [3 pts] Calculate the following partial derivatives of $b_1$.

**(i)** [1 pt] $\left(\frac{\partial b_1}{\partial A_{11}}, \frac{\partial b_1}{\partial A_{12}}\right)$

○ $(A_{11}, A_{12})$ ○ $(0, 0)$ ○ $(x_2, x_1)$ ○ $(A_{11}x_1, A_{12}x_2)$ ○ $(x_1, x_2)$

**(ii)** [1 pt] $\left(\frac{\partial b_1}{\partial A_{21}}, \frac{\partial b_1}{\partial A_{22}}\right)$

○ $(A_{21}, A_{22})$ ○ $(x_1, x_2)$ ○ $(1, 1)$ ○ $(0, 0)$ ○ $(A_{21}x_1, A_{22}x_2)$

**(iii)** [1 pt] $\left(\frac{\partial b_1}{\partial x_1}, \frac{\partial b_1}{\partial x_2}\right)$

○ $(A_{11}, A_{12})$ ○ $(A_{21}, A_{22})$ ○ $(0, 0)$ ○ $(b_1, b_2)$ ○ $(A_{21}x_1, A_{22}x_2)$

**(c)** [3 pts] Calculate the following partial derivatives of $f$.

**(i)** [1 pt] $\left(\frac{\partial f}{\partial A_{11}}, \frac{\partial f}{\partial A_{12}}\right)$

○ $(A_{11}, A_{12})$ ○ $(A_{11}b_1, A_{12}b_2)$ ○ $(A_{11}x_1, A_{12}x_2)$
○ $(x_1b_1, x_2b_1)$ ○ $(x_1b_2, x_2b_2)$ ○ $(x_1b_1, x_2b_2)$

**(ii)** [1 pt] $\left(\frac{\partial f}{\partial A_{21}}, \frac{\partial f}{\partial A_{22}}\right)$

○ $(A_{21}, A_{22})$ ○ $(A_{21}b_1, A_{22}b_2)$ ○ $(A_{21}x_1, A_{22}x_2)$
○ $(x_1b_1, x_2b_1)$ ○ $(x_1b_2, x_2b_2)$ ○ $(x_1b_1, x_2b_2)$

**(iii)** [1 pt] $\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)$

○ $(A_{11}b_1 + A_{12}b_2, A_{21}b_1 + A_{22}b_2)$ ○ $(A_{11}b_1 + A_{21}b_2, A_{12}b_1 + A_{22}b_2)$
○ $(A_{11}b_1 + A_{12}b_1, A_{21}b_2 + A_{22}b_2)$ ○ $(A_{11}b_1 + A_{21}b_1, A_{12}b_2 + A_{22}b_2)$

**(d)** [2 pts] Now we consider the general case where $\mathbf{A}$ is an $n \times d$ matrix, and $\mathbf{x}$ is a $d \times 1$ vector. As before, $f = \frac{1}{2}\left\|\mathbf{Ax}\right\|^2$.

**(i)** [1 pt] Find $\frac{\partial f}{\partial \mathbf{A}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

○ $\mathbf{x}^\top\mathbf{A}^\top\mathbf{Ax}$ ○ $\mathbf{Axx}^\top$ ○ $\mathbf{A}\left(\mathbf{A}^\top\mathbf{A}\right)^{-1}$ ○ $\mathbf{AA}^\top\mathbf{Ax}$ ○ $\mathbf{A}$

**(ii)** [1 pt] Find $\frac{\partial f}{\partial \mathbf{x}}$ in terms of $\mathbf{A}$ and $\mathbf{x}$ only.

○ $\mathbf{x}$ ○ $\left(\mathbf{A}^\top\mathbf{A}\right)^{-1}\mathbf{x}$ ○ $\mathbf{xx}^\top\mathbf{x}$ ○ $\mathbf{x}^\top\mathbf{A}^\top\mathbf{Ax}$ ○ $\mathbf{A}^\top\mathbf{Ax}$