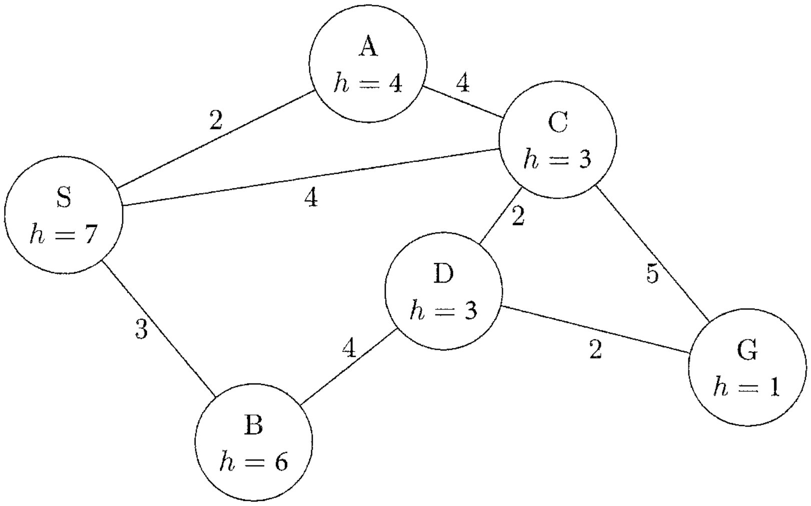
2. (20 points) Search Algorithms

Consider the following search problem with initial state — S, goal state — G, path costs indicated along edges, and a heuristic given.



For each of the following **graph search** strategies, report (i) the order in which states are expanded and (ii) the path returned by the *graph search.* Assume all ties (node orderings) use alphabetical ordering to select what is expanded first, e.g., S-A-D will be expanded before S-B-C. *Note, this requests using the graph search method (a state should only be expanded once).*

Search Method	States Expanded	Path Returned
Example	S, A, B, C, D, G	S-B-D-G
(a) Depth-first search	S,A,C,D,B,G	S,A,C,D,G
(b) Breadth-first search	S,A,B, C,D G	S,C, G
(c) Uniform cost search	S,A,B,C,D,G	S, C, G
(d) Greedy search using h	S, C, G	S, C,G
(e) A*using h	S, A,C,B,D,G	S,C,D,G

3. (5 points) Word Search

Consider a word search puzzle. Given a starting word, and ending word, generate a sequence of steps between the two by changing a single letter each time. All intermediate words must be real (in the dictionary). For example, with the start and end word pair of (`boat`, `gold`), a possible sequence is:

`boat` → `coat` → `colt` → `cold` → `gold`.

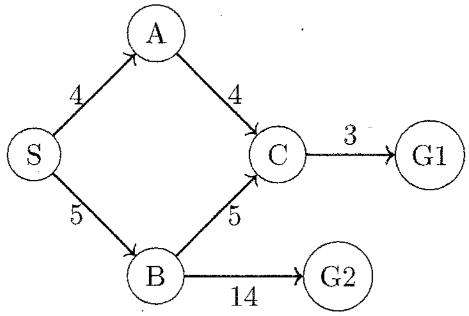
Which search strategy would be better for this problem: depth-first search or breadth-first search? Why? Explain your answer in terms of branching factor and the size of the total state space. *Hint: It may help to draw part of the search space.*

For a word with length L, the total state space should be 25^L, because there are 25 replacement of each letter in the word. The branch factor should be 25. But there is a constraint that the intermediate words must be real, so we can get the branching factor will be less than 25. Even though the total space count will be less than 25^L, it is still very big count for computing. In this situation we should choose BFS, which can find a relative short path to solutions.

In the given example, if we use DFS, ~~we should~~ imagine that we get the a-start word firstly, we will seach all the cases of a-start words, ~~and~~ then turn to search c-start (we should excpet the b-start word because it is the original letter). This will cost 3^{25} times for all the a-start word. In worst situations, it can cost lots of resources. In contrast, BFS cost $25 + 2^{25} + 3^{25} + 4^{25}$ in the worst situation, but it would be much better in common situations.

4. (13 points) A*and Heuristics

Consider the search state graph shown below, where S is a start state and $G1$ and $G2$ are goal states.



5

4 4 5

	S	A	B	C	G1	G2
h_1	8	5	3	2	1	0
h_2	0	0	0	0	0	0
h_3	7	4	4	1	0	0

3

14

(a) (3 pts) Which heuristics are admissible (or write *none*)?

h_1, h_2, h_3

(b) (3 pts) Which heuristics are consistent (or write *none*)?

h_1, h_2, h_3

(c) (3 pts) For heuristic h_1, what order will nodes (paths) be added to the fringe in A*graph search? Present info as ((S), $g + h = f$), ((S-A-C), 12+1=13), etc.

~~{(S), 0+8=8~~ ① {(S), 0+8=8 }

② {(S,A), 4+5=9 ; (S, B), 5+3=8}

③ {(S,A), 4+5=9 ; (S,B,C)= 5+5+2=12 ; (S,B,G2)= 5+14+0=19}

④ {(S,A,C) = 8+2=10 , (S,B,C) = 5+5+2=12 ; (S,B,G2) = 5+14+0 =19}

⑤ {(S,A,C,G1) = 11+1=12, (S,B,C) = 10+2=12 ; (S,B,G2) = 19+0=19 }

We get the G1

(d) (2 pts) For heuristic h_1, what order will states be added to the closed/explored set in A*graph?

{ S, B, A, C, G1}

(e) (2 pts) For heuristic h_1, what path with A*graph search return?

S, A, C, G1