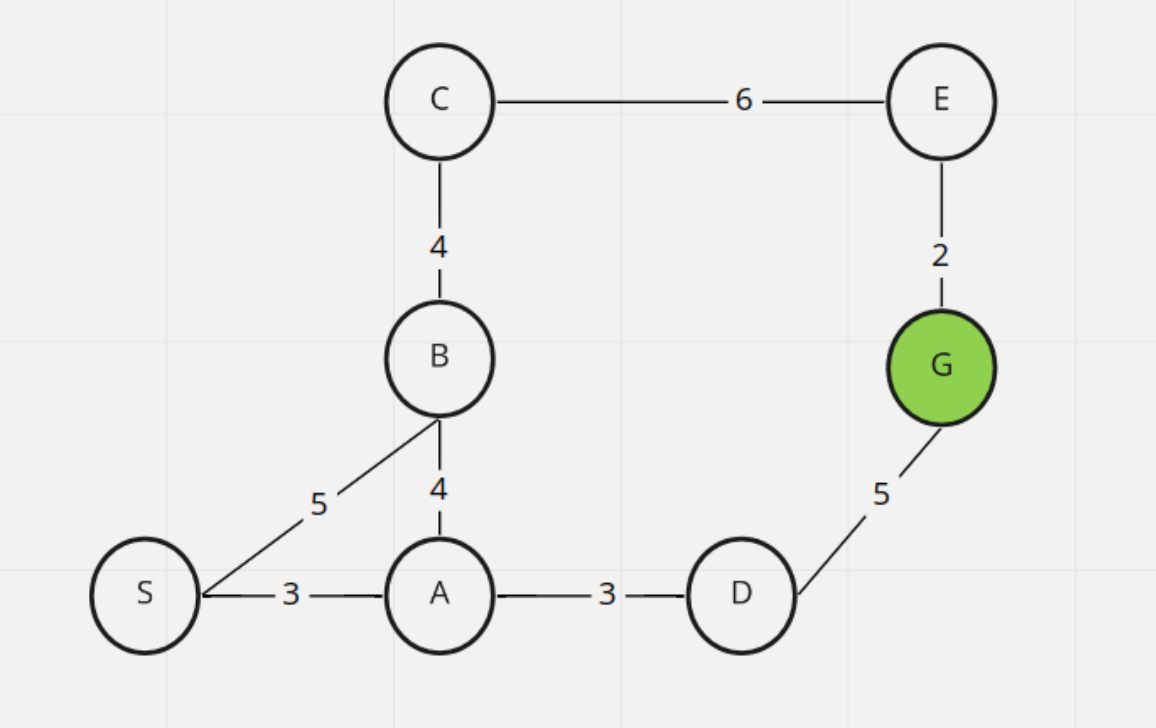
CA318
Labsheet #2

Question 1

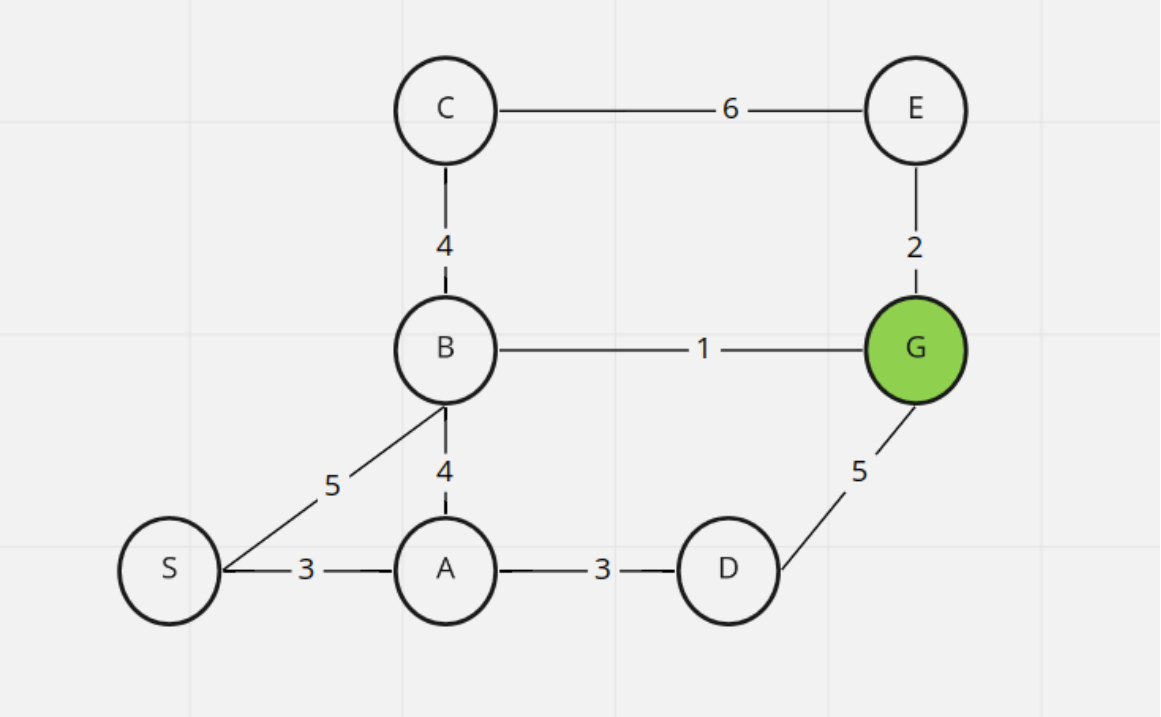
Using Branch and Bound, expand the search trees for the following maps and find the shortest path to G

(a)



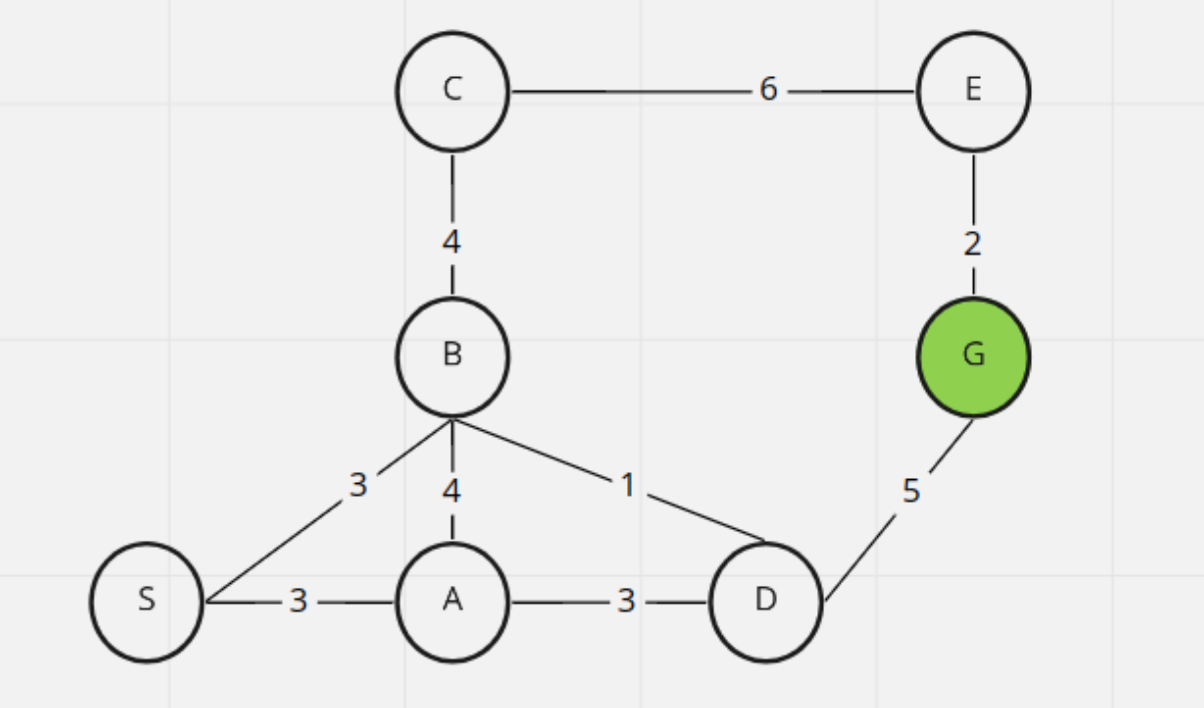
Step	Current Paths under consideration	Action
1	S-A (3) S-B(5)	Expand A
2	S-A-D (6) S-B (5)	Expand B
3	S-A-D (6) S-B-A (9) S-B-C (9)	Expand D
4	S-A-D-G (11) S-B-A (9) S-B-C (9)	Expand A
5	**S-A-D-G (11)** ~~S-B-A-D (12)~~ S-B-C (9)	Expand C
6	**S-A-D-G (11)** ~~S-B-A-D (12)~~ ~~S-B-C-D (15)~~	STOP
7	STOP, solution is S-A-D-G (11)	

(b)



Step	Current Paths under consideration	Action
1	S-A (3) S-B (5)	Expand A
2	S-A-D (6) S-B (5)	Expand B
3	~~S-A-D (6) S-B-A (9) S-B-C (9)~~ **S-B-G (6)**	STOP
4	STOP solution is **S-B-G (6)**	

(c)



Step	Current Paths under consideration	Action
1	S-A (3) S-B(3)	Expand A
2	S-A-D (6) S-B(3)	Expand B
3	S-A-D (6) S-B-A(7) S-B-C(7) S-B-D (4)	Expand D
4	S-A-D (6) S-B-A(7) S-B-C(7) S-B-D-G (9)	Expand D
5	~~S-A-D-G (11)~~ S-B-A(7) S-B-C(7) **S-B-D-G (9)**	Expand A
6	~~S-A-D-G (11)~~ ~~S-B-A-D(10)~~ S-B-C(7) **S-B-D-G (9)**	Expand C
7	~~S-A-D-G (11)~~ ~~S-B-A-D(10)~~ ~~S-B-C-E(13)~~ **S-B-D-G (9)**	STOP
8	STOP solution is S-B-D-G (9)	

Question 2

We mentioned in the lectures that Branch and Bound has a Sort Queue step. In practice this is not optimal.

1. Why is the Sort Queue step **not optimal**? Give a clear example

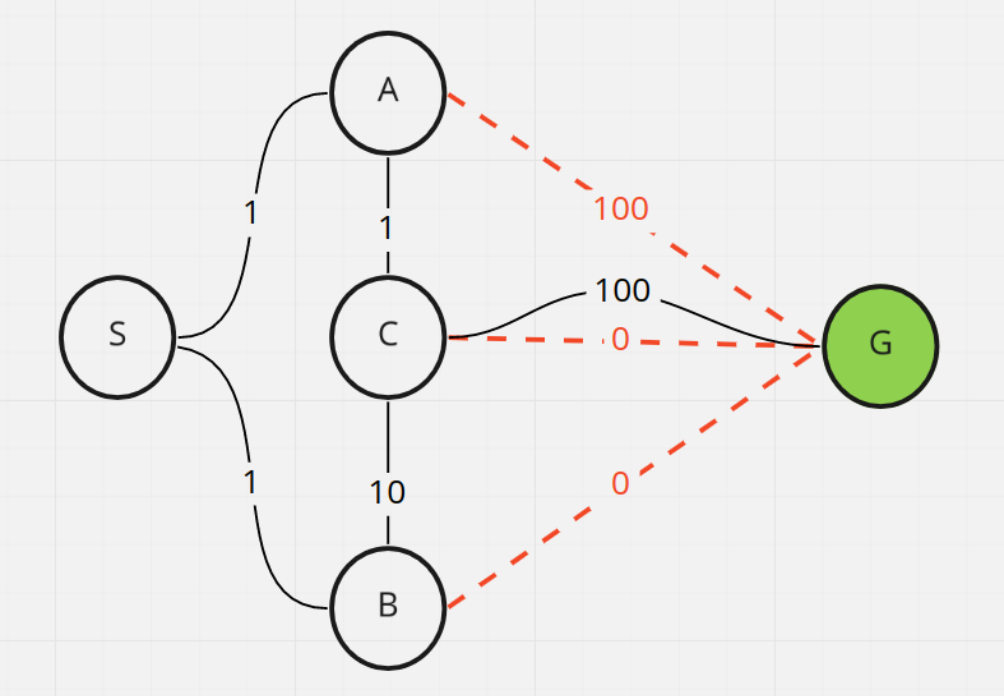
Sorting is an O(n^2) or O(n log n) overhead.

2. What could it be replaced with? Give a clear example

Whereas `min()` is O(n)

How would the change from step 2. affect the algorithm?
It would reduce the running time of the algorithm. Note also, because we only ever take the first value from the queue, replacing it with `min()` would be perfectly fine.

Question 3



Consider the graph above, with distance to the goal heuristics shown in red. Answer the following questions

1. Are the heuristics in red admissible? Explain your answer
    a. Yes as they underestimate the actual distance to the goal
2. What is the final state of the search graph when Branch and Bound completes?
    a. It fails to find the shortest path, instead choosing S-B-C-G
3. Do you have any comments on the way Branch and Bound performed?
    a. It performed incorrectly because the admissible heuristics were not **consistent**

Bonus Question

You have been asked to analyze how a BFS algorithm will perform, for a new game called "WeGo".

After a little analysis you arrive at the following data for WeGo:

- Level by level branching: [7,8,9,6,7,7,7,5,7]
- The WeGo game tree always has a depth of 10.

Answer the following questions, state any assumptions you make.

1. What size will the **open** list be?
    a. If the solution to WeGo is found at the last level & **last** element,
    b. If the solution to WeGo is found at the last level & **first** element
2. Worst case, how many **comparisons** will BFS have to perform:
    a. If the solution to WeGo is found at the last level & **last** element?
    b. If the solution to WeGo is found at the level & **first** element
3. If a **closed** list is also being maintained, worst case, how many elements will be stored in the **open+closed** lists if the solution to WeGo is found at the last level / last element?
4. If the amount of memory required for each element stored is 1KB, how much physical memory will be occupied:
    a. If a **closed** list is also being maintained, and the solution to WeGo is found at the last level / last element?
    b. If **no closed** list is maintained, and the solution to WeGo is found at the last level / last element?
5. What are the dangers of not maintaining a closed list?
6. What are the dangers of maintaining a closed list?
7. Are there any rules of thumb for deciding to maintain a closed list? What are they?
8. What are your general thoughts on BFS?

Assumptions: an average branching factor of mean(c(7,8,9,6,7,7,7,5,7)) = 7

1.
    a. 7^9 = 40353607
    b. 7^8 = 5764801
2.
    a. x <- c(0:9), sum(7^x) = 47079208
    b. x <- c(0:8), sum(7^x) = 6725601
3. x <- c(0:9), sum(7^x) = 47079208
4.
    a. sum(7^x) * 1024 = 48,209,108,992 (48GB)
    b. 7^9 * 1024 = 41,322,093,568 (41GB)
5. Exhausts memory faster than no closed list (41 v 48GB)
6. Yes, if each node has exactly one distinct path to it then we don't need to maintain the closed list
7. Infeasible except for very small problems