PART 4: Path planning

Exercises for the path planning part. It is recommended to print the graphs used for these exercises and draw on them to run through the algorithms by hand. These drawings can be handed in as solutions. The graphs can be found at the end of part 4.

Exercise 4.1: Depth-first search (DFS) algorithm

Figure 2 shows a map of cities connected with roads. Each circle on the map is a node which represents a city, and the connections between them are edges representing roads. You need to get from city s0 to city s23. Use the DFS algorithm to find a route from city s0 to city s23.

You can read more about the DFS algorithm here `https://en.wikipedia.org/wiki/Depth-first_search` and in the book "Introduction to Algorithms" by Cormen, Leierson, Rivest and Stein. 3rd edition.", page 603-612, which can be found here: `https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf`.

1:
Start by stacking the lowest of the s numbers when it is possible to stack multiple. Start with s0 as the first frontier in the stack.

Write down the steps taken by the algorithm. In which order the nodes are expanded and what frontiers are queued at that point.

When the goal is reached draw the found route to the goal or write down the route from s0 to the goal, draw the layer numbers on each node or specify in which layer each node belong. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

2:
Try to start by stacking the highest of the s numbers when it is possible to stack multiple. Start with s0 as the first frontier in the stack.

Write down the steps taken by the algorithm. In which order the nodes are expanded and what frontiers are queued at that point.

When the goal is reached draw the found route to the goal or write down the route from s0 to the goal, draw the layer numbers on each node or specify in which layer each node belong. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

3:
Notice any difference in the paths obtained using the two different DFS methods? Why is one route longer than the other?

Exercise 4.2: Breadth-first search (BFS) algorithm

Figure 2 once more shows a map of cities connected with roads. This time the roads are toll roads. Each of the roads have the same fee. Use the BFS algorithm to find a route city s0 to city s23, where you have to pay the least amount of money.

You can read more about the DFS algorithm here `https://en.wikipedia.org/wiki/Breadth-first_search` and in the book "Introduction to Algorithms" by Cormen, Leierson, Rivest and Stein. 3rd edition.", page 594-603, which can be found here: `https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf`.

1:
Start by queuing the lowest of the s numbers when it is possible to queue multiple. Start with s0 as the first frontier in the queue.

Write down the steps taken by the algorithm. In which order the nodes are expanded and what frontiers are queued at that point.

When the goal is reached draw the found route to the goal or write down the route from s0 to the goal, draw the layer numbers on each node or specify in which layer each node belong. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

2:
Comment on the advantages and disadvantages of using BFS vs DFS

Which type of algorithm is preferred for robots?

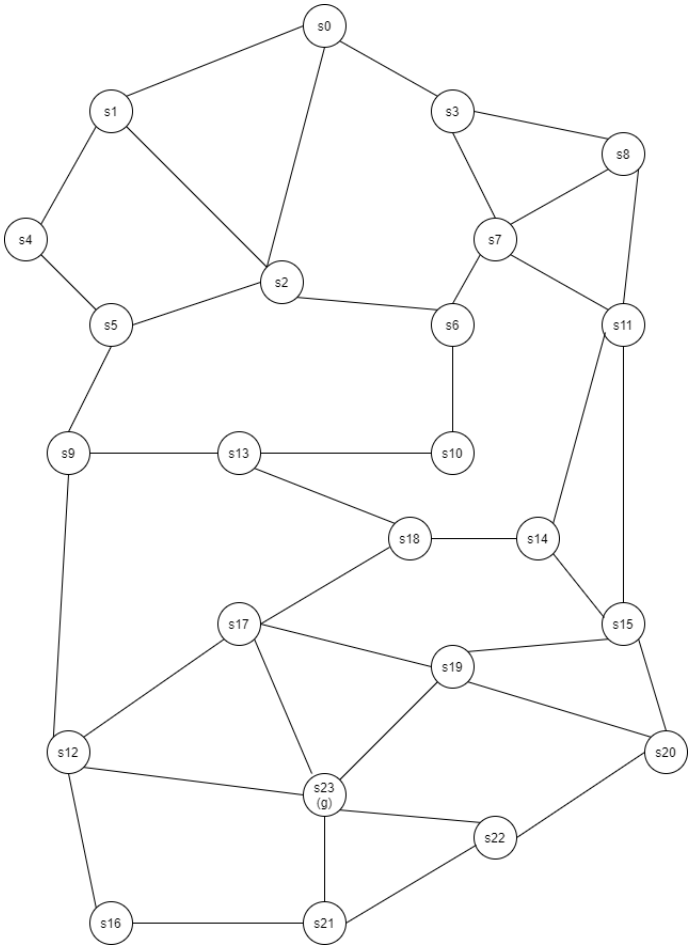

Figure 2: The graph used for Exercise 4.1 and 4.2. A larger picture can be found in ... (Appendix or end of Document?)

Exercise 4.3: Dijkstra's algorithm

Figure 3 shows the same city as Figure 2, but now the length of the roads have been added to the map. You once more want to get from city s0 to city s23, but this time you are in a rush and want to get to city s23 as fast as possible. You are allowed to drive the same speed on all roads. Use Dijkstra's algorithm to find the fastest route from city s0 to city s23.

You can read more about the Dijkstra algorithm here `https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm` and in the book "Introduction to Algorithms" by Cormen, Leierson, Rivest and Stein. 3rd edition.", page 658-664, which can be found here: `https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf`.

Write down the steps taken by the algorithm. In which order the nodes are expanded and what frontiers are generated at which point with what distance to it. Specify if a frontier is updated and write the new distance.

When the goal is reached draw the found route to the goal or write down the route from s0

to the goal, draw the distance to each node from s0 or specify the distance to each node from s0. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

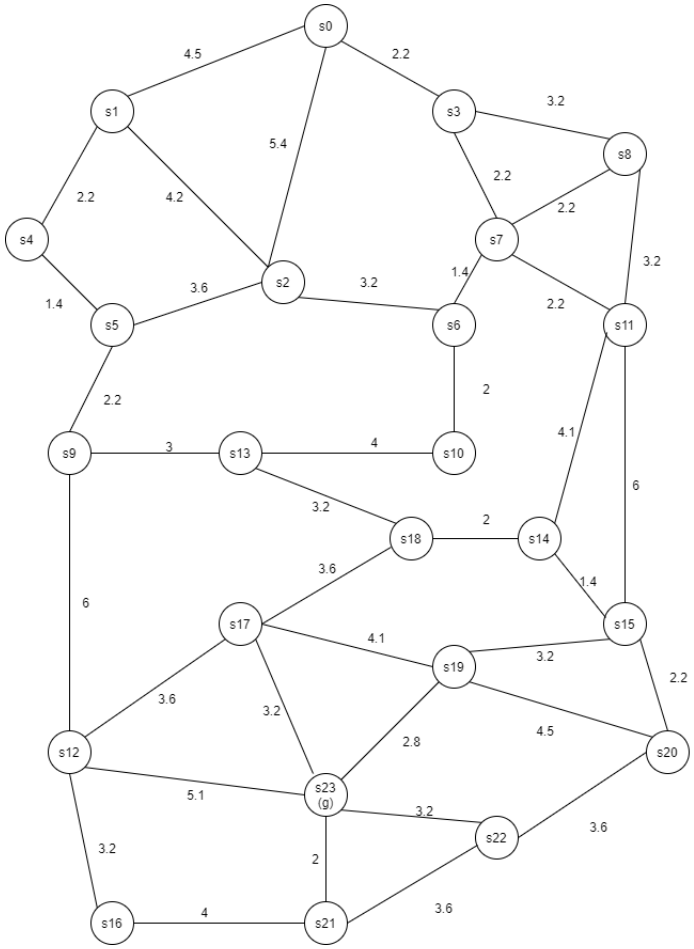

Figure 3: The graph used for Exercise 4.3. A larger picture can be found in ... (Appendix or end of Document?)

Exercise 4.4: Greedy best-first search algorithm

Figure 4 now shows the map of the cities with the distance from the cities to the goal city added in red. Use the Greedy best-first search algorithm to find a route from city s0 to city s23,

You can read more about the Greedy best-first search algorithm here `https://www.javatpoint.com/ai-informed-search-algorithms` and in the book "Artificial Intelligence Modern Approach" by S. Russell and P. Norvig.", page 92-96, which can be found here: `https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf`.

Write down the steps taken by the algorithm. In which order the nodes are expanded and

what frontiers are generated at which point with what distance to it. Specify if a frontier is updated and write the new distance.

When the goal is reached draw the found route to the goal or write down the route from s0 to the goal, draw the distance to each node from s0 or specify the distance to each node from s0. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

Exercise 4.5: A* search algorithm⋆

Now use both the direct distance and the length of the roads to get the shortest route from city s0 to city s23. Use the A* algorithm to obtain the shortest route.

You can read more about the A* algorithm here `https://en.wikipedia.org/wiki/A*_search_algorithm` and here `https://www.javatpoint.com/ai-informed-search-algorithms` and in the book "Artificial Intelligence Modern Approach" by S. Russell and P. Norvig.", page 96-101, which can be found here: `https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-9780131038059.25368.pdf`..

Write down the steps taken by the algorithm. In which order the nodes are expanded and what frontiers are generated at which point with what distance to it. Specify if a frontier is updated and write the new distance.

When the goal is reached draw the found route to the goal or write down the route from s0 to the goal, draw the distance to each node from s0 or specify the distance to each node from s0. Specify the total number of expanded nodes and the total number of frontiers generated. Show the path, as well as, the length of the path.

This route is the same as the one found by Dijkstra's algorithm. What is the advantage of using A*?

Exercise 4.6: Advantages and Disadvantages of Greedy best-first search and A* search algorithms

1:
What are the advantages and disadvantages of Greedy best-first search algorithm?

2:
What are the advantages and disadvantages of A* search algorithm?

3:
Which algorithm is best for aerial robots and why?

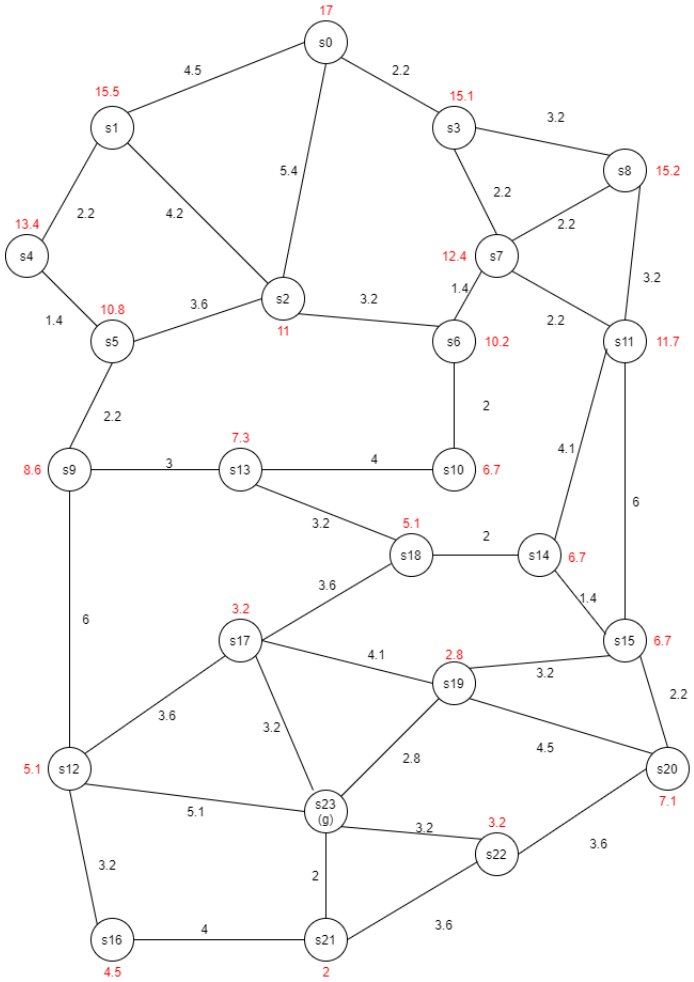

Figure 4: The graph used for Exercise 4.4 and 4.5. A larger picture can be found in ... (Appendix or end of Document?)

Exercise 4.7: Upgrade the Greedy best-first search from 2D to 3D

A search algorithm can be used to navigate through a grid based map. Such a map have been implemented in the file called *map_script.m*.

Open the file called *map_script.m*. The first part of the the file defines the map, including start and end position. The middle part runs the function called *greedy_2d*, and the last part draws the map and found route.

Run the Matlab script and watch the algorithm find the solution.

The *greedy_2d* function is an implementation of the Greedy best-first search algorithm in Matlab. Open the *greedy_2d.m* file and read through the function. Try to understand the implementation.

In order for a drone to navigate in 3D, the grid needs to be in 3D, and the algorithm

therefore needs to be upgraded to work in 3D.

Open the file called *map_script_3d.m*. This file is the same as *map_script.m* but it is in 3D.

Upgrade the greedy_2d function from 2D to 3D, and use it to solve the maze.

Hint: Before the nodes expanded in a square pattern, now it needs to expand in a cubic pattern.

Exercise 4.8: Modify the implementation of the Greedy best-first search algorithm into a A* search algorithm

The Greedy best-first search algorithm does not always find the optimal solution. To do this an A* algorithm is needed.

Modify the implementation of the Greedy best-first search algorithm to be a A* search algorithm

What changes have to be made?

Did it affect the running time of the algorithm?

Hint: The cost is now as $f(n) = h(n) + g(n)$ where $h(n)$ is the distance to the goal, and $g(n)$ is the cost between nodes.

What algorithm would be implemented if the formula was $f(n) = g(n)$ instead?

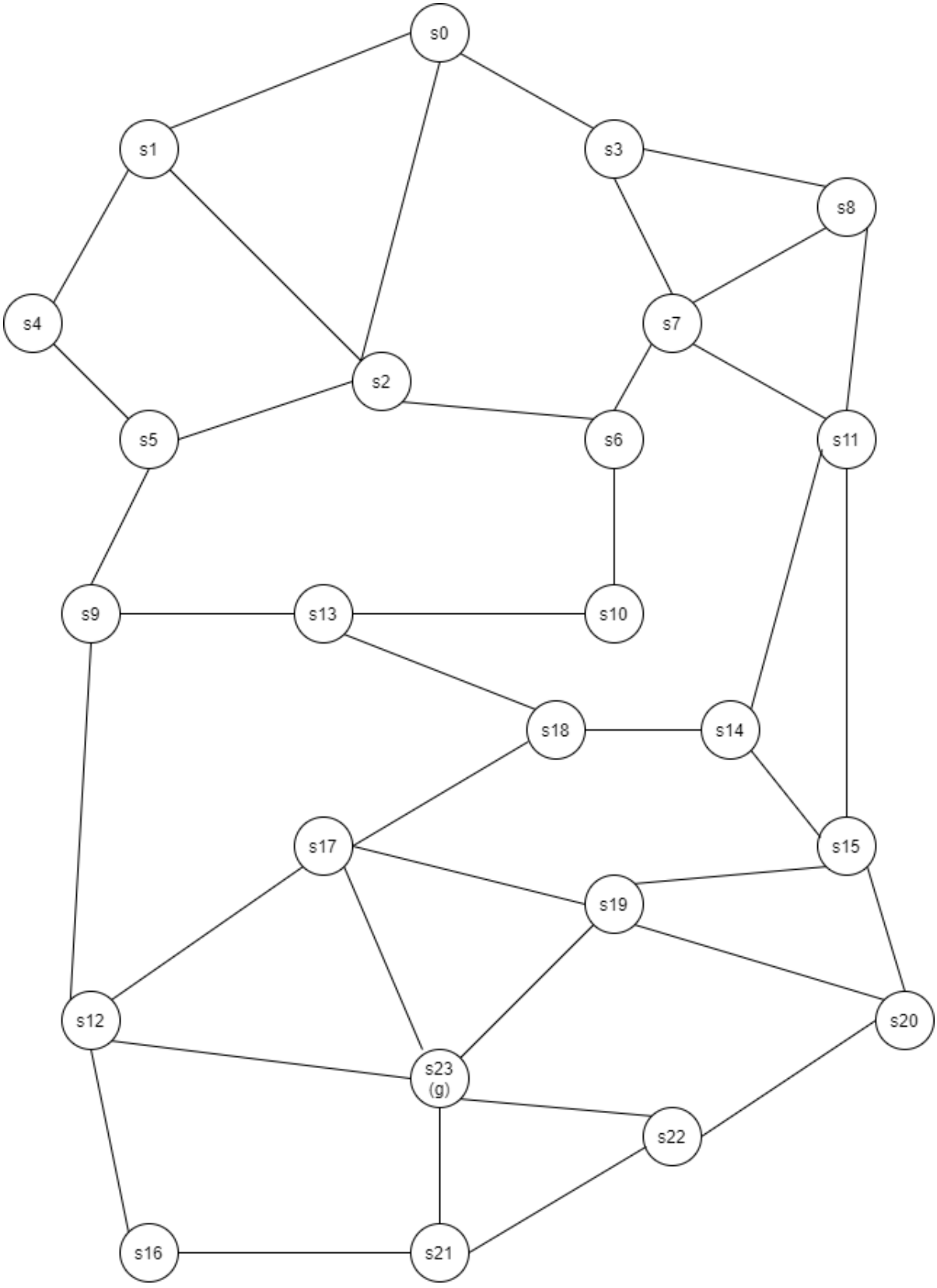

Figure 5: The graph used for Exercise 4.1 and 4.2

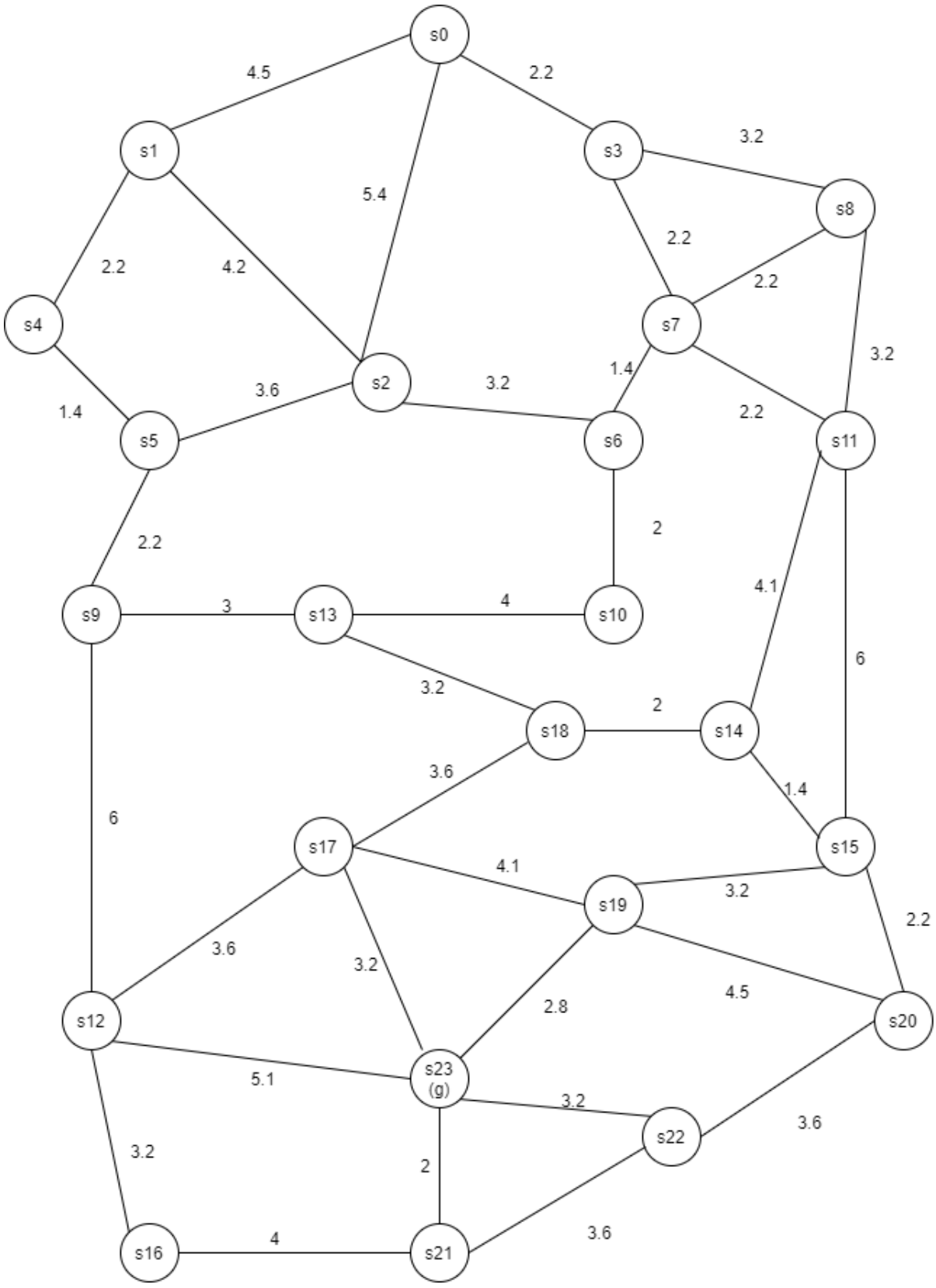

Figure 6: The graph used for Exercise 4.3

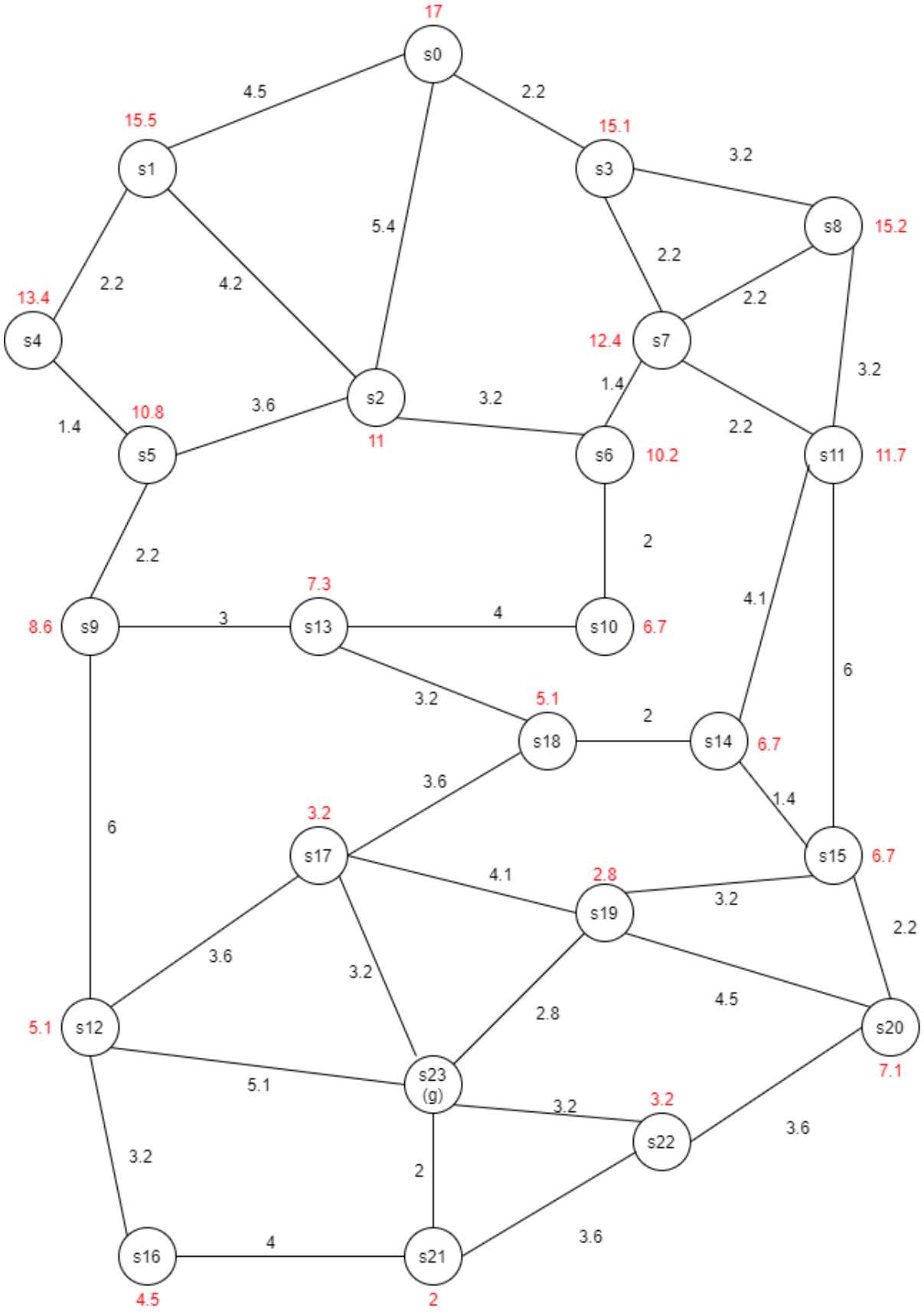

Figure 7: The graph used for Exercise 4.4 and 4.5