Difference between Greedy Search and A* Search.

While Greedy search depends only on the heuristic value, A* search depends both on heuristic values, as well as the path cost to that node from the root node, both having the advantage of being a 'directed' search when compared to other searching algorithms such as UCS.

Greedy Search:
The advantage of Greedy search is that it is directed towards the goal and uses only heuristic values to compute the path to the goal, which is quite simple to implement, and also, the store space is required less.
The disadvantage of Greedy search is that it's main focus is to move towards a node which is closer(less heuristic value) to the goal node. Hence it might not be an optimal solution sometimes because, there might be a path in a single iteration, which might take you away from the goal, but in the next iterations, it is very much close to the goal, and hence would lead to shortest path. But this node is never reached by Greedy, because it always chooses a node with least heuristic value. This concludes that the heuristic values alone can't give the optimal solution all the time.

A* Search: g(x) + h(x)
The advantage of A* is that it is a combination of both USC and Greedy search. It takes the path cost to that node from the root, as well as the heuristic value to the goal node. This makes it a better search algorithm than Greedy Search because, it focuses on both
1. Keeping the cost less from earlier nodes. (minimize g(x))
2. Reach the goal early as possible. (minimize h(x))

The disadvantage of A* search is that it still is not optimal because it depends on the heuristic values. Also, it should keep track of the path cost to that node, more space is required.
Hence, it would find the lowest cost path, ONLY if:
1. h(x) < true cost.
2. h is never overestimated distance to goal.
3. h is optimistic
4. h is admissible.
This means, optimistic A* finds the lowest cost path.

Justification with Example from the given problem:
Each movie would have neighbors as follows:
1 ⟶ [3, 2, 4]
2 ⟶ [8, 1, 6, 5]
3 ⟶ [1, 5, 7, 6]
4 ⟶ [8, 7, 5, 1]
5 ⟶[3, 4, 9, 2]
6 ⟶[9, 2, 10, 3]
7 ⟶[4, 3, 9, 11, 12]
8 ⟶[2, 4, 10]
9 ⟶[6, 5, 7]
10⟶[6, 8, 12]
11⟶[7, 12]
12⟶[7, 11, 10]

Output Sample:
Greedy: m1, m3, m6.
A* : m1, m3, m4.

1. Expand: (1): Neighbors of (1) ⟶(2, 3, 4)
Greedy chooses: (3) as first child, as h(3) is least among h(3), h(2), h(4).
A* Chooses: (3) as first child, as g(3)+h(3) is least among g(2)+h(2), g(3)+h(3) and g(4)+h(4).
Both behavior is same.

However,
2. Expand: (3): Neighbors of (3) ⟶ (1, 5, 7, 6)
Greedy Chooses: (6) because heuristic of (6) is least among 5, 7 and 6.
A* Chooses: (4), even though it has a very high h(x) value. This is because, even if it has a high h(x) value, it is trying to minimize both h(x) and g(x), hence it's goal being to minimize the cost, as well as to stay focused to move towards the goal(g(x)+h(x)).