Algorithms Programming Project I
Greedy Algorithms

February 2023

1 Team Members

Yujie Wang: UFID-59913548
Haolan Xu: UFID-34326768

2 Greedy Strategies

2.1 Strategy 1

The first strategy is to iterate over each day starting from day 1 to day n. For each day, we consider all unpainted houses that are available to be painted on that day, and choose the house that has the earliest start day among them.

2.1.1 Analysis of algorithm:

1.The given houses have been in ascending order of start day.

2.Initialize the number of painted houses to 0 and a priority queue to store the end days of available houses.

3.Iterate each day from day 1 to day n.

4.For each day, among the unpainted houses that are available that day, paint the house that have the earliest start day, and if a house is painted, it will be marked which means it will not appear in later iterations for houses.

5.Keep track of the number of painted houses and return it as the answer.

Because the given houses are in ascending order of start day, we don't need to sort them, and just need to iterate n days, which means this algorithm's running time is θ(n).

2.1.2 The example of algorithm:

Consider the following input:

Table 1: S1 instance

n	m
3	4
StartDay	EndDay
1	3
1	4
2	4
3	4

Sort the houses as follows: [(1, 3), (1, 4), (2, 4), (3, 4)]

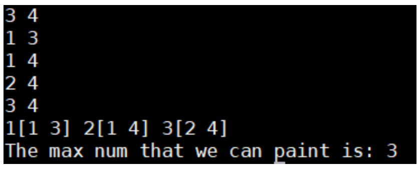
Initialize ans = 0 and a priority queue.

On day 1, available houses are (1, 3) and (1, 4). Choose (1, 3) and update ans = 1.

On day 2, available houses are (1, 4) and (2, 4). Choose (1, 4) since its start day is earlier and update ans = 2.

On day 3, available houses are (2, 4), and (3, 4). Choose (2, 4) since its start day is earlier and update ans = 3.

The final answer is 3, which is the maximum number of houses that can be painted.
The output in our code:



```
3 4
1 3
1 4
2 4
3 4
1[1 3] 2[1 4] 3[2 4]
The max num that we can paint is: 3
```

Figure 1: S1 Output

2.1.3 The counter example:

Consider the following input:

Table 2: S1 counter example

n	m
5	4
StartDay	EndDay
1	2
2	3
2	4
3	3

Sort the houses as follows: [(1, 2), (2, 3), (2, 4), (3, 3)]

After strategy 1 algorithm, we can get the result that ans = 3 with (1, 2), (2, 3), (2, 4). But the optimal solution is that ans = 4 with (1, 2), (2, 3), (3, 3), (2, 4). So the strategy 1 can't always get the optimal solution.

2.2 Strategy 2

The second strategy is to iterate over each day starting from day 1 to day n. For each day, we consider all unpainted houses that are available to be painted on that day, and choose the house that has the latest end day among them.

2.2.1 Analysis of algorithm:

1.Sort the houses based on their end day in descending order.

2.Initialize a priority queue to keep track of available end days of the painted houses.

3.Iterate over each day starting from day 1 to n.

4.For each day, among the unpainted houses that are available that day, paint the house that have the latest end day, and if a house is painted, it will be marked which means it will not appear in later iterations for houses.

5.Keep track of the number of painted houses and return it as the answer.

The time complexity of this algorithm is θ(m log m + n), where m is the number of houses and n is the number of days. The O(m log m) term is due to the sorting step, and the θ(n) term is due to iterating over n days.

2.2.2 The example of algorithm:

Consider the following input:

Table 3: S2 instance

n	m
3	4
StartDay	EndDay
1	3
1	4
2	3
3	4

Sort the houses as follows: [(1, 4), (3, 4), (1, 3), (2, 3)]

Initialize ans = 0 and a priority queue.

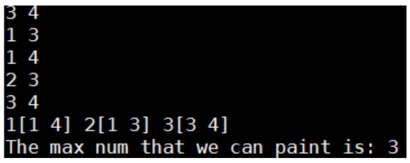
On day 1, available houses are (1, 3) and (1, 4). Choose (1, 4) since its end day is later and update ans = 1.

On day 2, available houses are (2, 3) and (1, 3). Choose (1, 3) and update ans = 2.

On day 3, available houses are (3, 4), and (2, 3). Choose (3, 4) since its end day is later and update ans = 3.

The final answer is 3, which is the maximum number of houses that can be painted.

The output in our code:



```
3 4
1 3
1 4
2 3
3 4
1[1 4] 2[1 3] 3[3 4]
The max num that we can paint is: 3
```

Figure 2: S2 Output

2.2.3 The counter example

Consider the following input:

Table 4: S2 counter example

n	m
3	3
StartDay	EndDay
1	1
1	3
2	2

Sort the houses as follows: [(1, 3), (2, 2), (1, 1)]

On day 1, available houses are (1, 1) and (1, 3). Choose (1, 3) since its end day is later.

On day 2, available houses are (2, 2). Choose (2, 2).

On day 3, no available houses.

The final answer is 2, but the optimal answer is 3. So the strategy 2 cannot always get the optimal solution.

2.3 Strategy 3

The third strategy is to iterate over each day starting from day 1 to day n. For each day, we consider all unpainted houses that are available to be painted on that day, and choose the house that has the shortest duration among them.

2.3.1 Analysis of algorithm:

1.Sort the houses based on their duration in ascending order.

2.Initialize a priority queue to keep track of available end days of the painted houses.

3.Iterate over each day starting from day 1 to n.

4.For each day, among the unpainted houses that are available that day, paint the house that have the shortest duration, and if a house is painted, it will be marked which means it will not appear in later iterations for houses.

5.Keep track of the number of painted houses and return it as the answer.

The time complexity of this algorithm is θ(m log m + n), where m is the number of houses and n is the number of days. The O(m log m) term is due to the sorting step, and the θ(n) term is due to iterating over n days.

2.3.2 The example of algorithm:

Consider the following input:

Table 5: S3 instance

n	m
3	4
StartDay	EndDay
1	3
1	4
2	4
3	4

Sort the houses as follows: [(3, 4), (2, 4), (1, 3), (1, 4)]

Initialize ans = 0 and a priority queue.

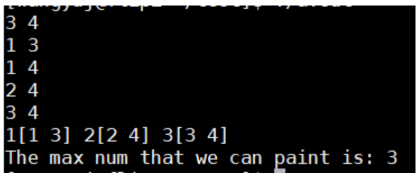
On day 1, available houses are (1, 3) and (1, 4). Choose (1, 3) since it has the shorter duration and update ans = 1.

On day 2, available houses are (2, 4) and (1, 4). Choose (2, 4) since it has the shorter duration and update ans = 2.

On day 3, available houses are (3, 4), and (1, 4). Choose (3, 4) since it has the shorter duration and update ans = 3.

The final answer is 3, which is the maximum number of houses that can be painted.

The output in our code:



```
3 4
1 3
1 4
2 4
3 4
1[1 3] 2[2 4] 3[3 4]
The max num that we can paint is: 3
```

Figure 3: S3 Output

2.3.3 The counter example:

Consider the following input:

Table 6: S3 counter example

n	m
4	4
StartDay	EndDay
1	2
1	3
2	3
3	4

Sort the houses as follows: [(1, 2), (3, 4), (2, 3), (1, 3)]

After strategy 3 algorithm, we can get the result that ans = 3 with (1, 2), (2, 3), (3, 4). But the optimal solution is that ans = 4 with (1, 2), (2, 3), (1, 3), (3, 4). So the strategy 3 can't always get the optimal solution.

2.4 Strategy 4

The fourth strategy is to iterate over each day starting from day 1 to day n. For each day, we consider all unpainted houses that are available to be painted on that day, and choose the house that has the earliest end day among them.

2.4.1 Analysis of algorithm:

1.Sort the houses based on their end day in ascending order.
2.Initialize a priority queue to keep track of available end days of the painted houses.
3.Iterate over each day starting from day 1 to n.
4.For each day, among the unpainted houses that are available that day, paint the house that have the earliest end day, and if a house is painted, it will be marked which means it will not appear in later iterations for houses.
5.Keep track of the number of painted houses and return it as the answer.

The time complexity of this algorithm is θ(m log m + n), where m is the number of houses and n is the number of days. The O(m log m) term is due to the sorting step, and the θ(n) term is due to iterating over n days.

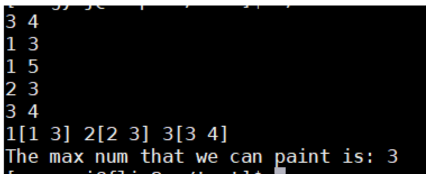
2.4.2 The example of algorithm:

Consider the following input:

Table 7: S4 instance

n	m
3	4
StartDay	EndDay
1	3
1	5
2	3
3	4

Sort the houses as follows: [(1, 3), (2, 3), (3, 4), (1, 5)]
Initialize ans = 0 and a priority queue.
On day 1, available houses are (1, 3) and (1, 5). Choose (1, 3) since its end day is earlier and update ans = 1.
On day 2, available houses are (2, 3) and (1, 5). Choose (2, 3) since its end day is earlier and update ans = 2.
On day 3, available houses are (3, 4), and (1, 5). Choose (3, 4) since its end day is earlier and update ans = 3.
The final answer is 3, which is the maximum number of houses that can be painted.
The output in our code:



```
3 4
1 3
1 5
2 3
3 4
1[1 3] 2[2 3] 3[3 4]
The max num that we can paint is: 3
```

Figure 4: S4 Output

2.4.3 The correctness of the strategy

To prove the correctness of this strategy, we can use contradiction. Assume that there is a better strategy called strategy X that paints more houses than strategy 4.

Now suppose the the last house painted by 4 is named G, and X has an additional house H painted in addition to G

Let's call the end day of H is named dH, and the end day of G is named dG

Now, in strategy 4, since it prefers to painting the house with earlier end day, which means dG < dH, but because of dG < dH, it is possible that strategy 4 can paint H. So the number of houses that strategy 4 paints equals the number of houses that strategy X paints, which contradicts our assumption.

Therefore, our assumption that strategy X paints more houses than strategy 4 must be false, and strategy 4 is optimal.

3 Experimental Comparative Study

3.1 The approach

To test the optimality of the different implementations, we generate randomly generated input files of various sizes, where each file specifies the startDay and endDay of a set of houses. And we use a random number generator to generate the startDay and endDay values for each house within a specified range (e.g., startDay between 1 and n-1, endDay between startDay and n). And we repeat this process to generate multiple input files for each input size.

Here is our input size:

Table 8: Experimental comparative study input size

n	m
100	80
500	400
1000	800
2000	1800
3000	2900

We compare the results of each implementation for each input size, and plot a grouped histogram with the number of houses painted (y-axis) against the input size (x-axis), grouping together all implementations for a given input size.

To ensure that the results are consistent and reproducible, we run each implementation three times on each input file and take the average of the results. And we also use a random seed to ensure that the same input files are generated each time the experiments are run.

By analyzing the results and comparing the performance of each implementation, we determine which strategy is most effective for solving the house painting problem for different input sizes.

3.2 The result

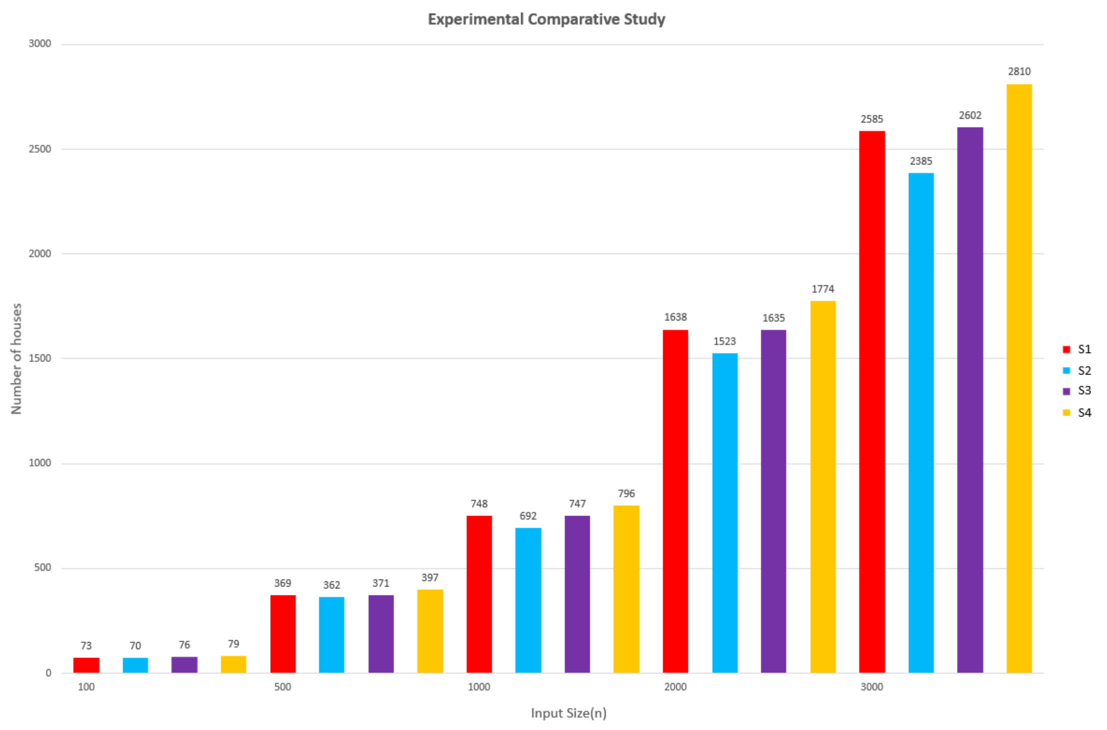


Figure 5: Experimental comparative study result

4 Conclusion

Overall, this project assignment involved implementing different strategies to solve a problem of selecting houses to be painted given their availability periods. It was a good learning experience as it required a good understanding of the problem, algorithm design, and implementation using a programming language.

For the task of implementing the strategy of iterating over each day and selecting the house that started being available the earliest, the algorithm was relatively straightforward to implement. One potential challenge was ensuring that the implementation had a running time of θ(n), where n is the number of days. This required that the given houses have been sorted by their startDay and endDay.

For the task of implementing the other strategies, the algorithms were also relatively straightforward to implement. The main challenge was to prove or disprove the optimality of each strategy, which required analyzing and testing the strategies with different input sizes and types. After our analysis, we found that s2 and s3 weren't optimal with the counter examples but s4 was optimal using the approach in the book to prove it.

For the task of generating random input files in C++, the implementation was relatively easy. It involved generating random numbers within a given number and writing them to a file. One potential challenge was ensuring that the generated input files were valid, i.e., the startDay of a house should be less than or equal to its endDay.

Overall, this project assignment provided a good opportunity to learn and practice algorithm design and implementation using a programming language, and also to think critically about the optimality of different strategies.

5 Bonus

5.1 Analysis of algorithm:

After analyzing the four strategies, we found that s4 was optimal. So we begin to give an θ(m log(m)) implementation of s4.

Considering that the running time is θ(m log m), we begin to iterate the houses after sorting the houses in ascending order of endDay.

We just can choose the compatible houses to maximize the houses to be painted since we can't iterate days which means we can't compare the houses' startDay and endDay with the n days to choose the suitable houses. But in this implementation, we can't get the optimal solution in strategy 4.

Here is the counter example:

Table 9: Counter example for strategy 5

n	m
4	4
StartDay	EndDay
1	2
2	3
3	4
3	5

In strategy 5, we can get the solution that (1, 2), (3, 4), the number of painted houses is 2. But the optimal solution is that (1, 2), (2, 3), (3, 4), (3, 5), the max number of painted houses is 4. So the s5 is not optimal.

5.2 Experimental Study

5.2.1 The approach

In this experimental study, we use the same input size in 3.1 to strategy 5.

5.2.2 The result

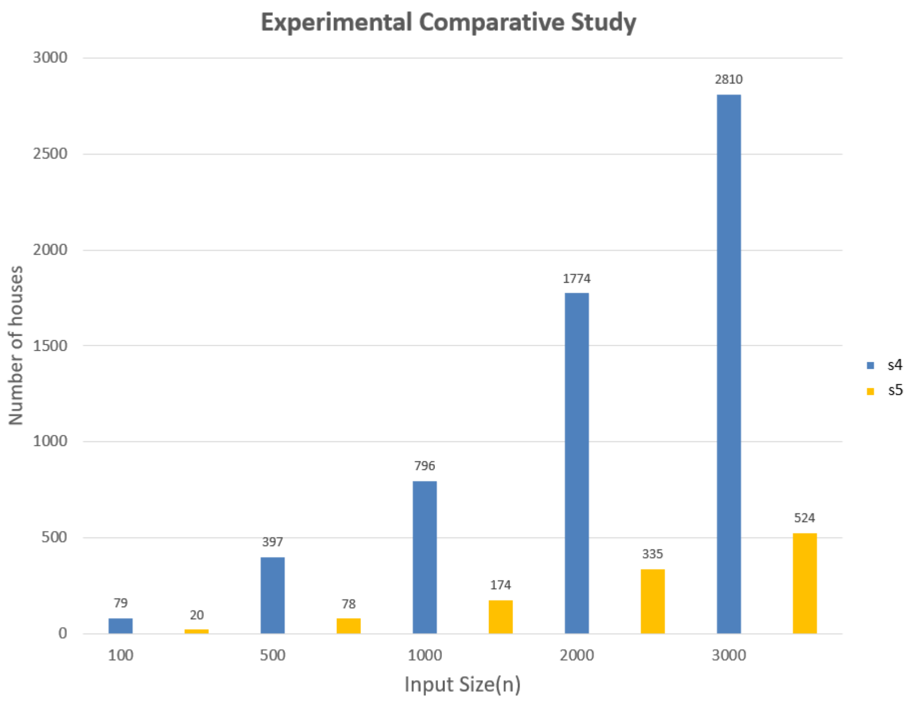


Figure 6: Experimental comparative study result for s4 and s5