AOA Program Project I

Team 68: Zhenyu Wang (UFID: 6045-7013)   Heng Zhou (UFID: 8627-8332)

Main contributions

Zhenyu Wang:

a) Provide ideas for the code implementation of the algorithm.
b) Analyze the complexity of each algorithm.
c) Implement functions of the algorithm.
d) Write and test the makefile.

Heng Zhou:

a) Write pseudocode.
b) Analyze the complexity of each algorithm.
c) Implement functions of the algorithm, test it, and make diagrams.
d) Correctness Proof

Strategy 1 **2**
Algorithm 2
Pseudo-code 2
Running Time 2
Correctness Proof 2
Strategy 2 **3**
Algorithm 3
Pseudo-code 3
Running Time 3
Correctness Proof 3
Strategy 3 **4**
Algorithm 4
Pseudo-code 4
Running Time 4
Correctness Proof 5
Strategy 4 **5**
Algorithm 5
Pseudo-code 5
Running Time 5
Correctness Proof 6
STRAT 1-4 Experiments **6**
Strategy 5 **8**
Algorithm 8
Pseudo-code 8
Analysis 8
Experiments 8
Conclusion **9**

Strategy 1

Algorithm

Earliest-Start-Time-First: Iterate over each day starting from day 1, …, n. For each day, among the unpainted houses that are available to be painted on that day, paint the house that started being available the earliest.

Pseudo-code

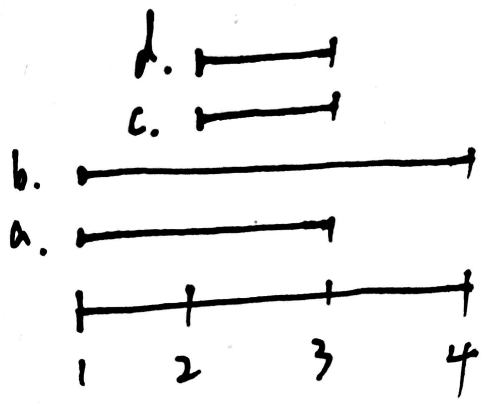
```
i = 0, j = 0
A = ∅
while (j < m) {
        if ($s_j$<=i<=$f_j$) {
                A = A U {j}
                j++
                break
        }
        else if ($s_j$ > i) {
                break
        }
        else {
        j++
        }
}
return A
```

Running Time

- Choose house to paint is O(1) and keep track of all houses is O(m)
- Total time is O(m)

Correctness Proof

Counterexample:



By Strategy 1, days [1, 2, 3, 4] will paint houses [a, b, c, null]. So, it is not an optimal algorithm.

Strategy 2

Algorithm

Latest-Start-Time-First: Iterate over each day starting from day 1, …, n. For each day, among the unpainted houses that are available that day, paint the house that started being available the latest.

Pseudo-code

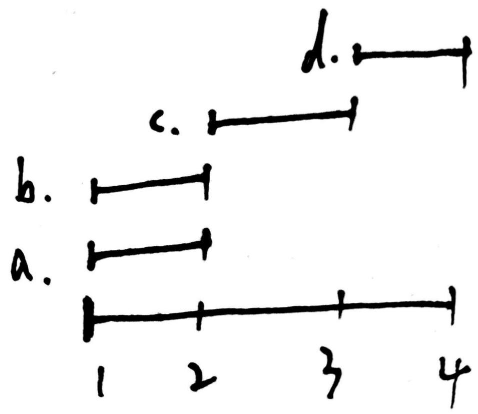
```
Initially, H is the set of all houses
i = 0
A = ∅
for (i in n) {
        choose j ∈ H that $s_j<=i<=f_j$ and $s_j$ is largest
        add j to A
}
return A
```

Running Time

- Add all houses to priority queue is O(mlogm)
- Choose latest start time house to paint is O(logm), for all houses it is O(mlogm)
- Keep track of all days is O(n)
- Total time is O(n+mlogm)

Correctness Proof

Counterexample:



By strategy 2, the days [1, 2, 3, 4] will paint houses [a, c, d, null]. So, it is not an optimal algorithm.

Strategy 3

Algorithm

Shortest-Duration-Time-First: Iterate over each day starting from day 1, …, n. For each day, among the unpainted houses available that day, paint the house available for the shortest duration.

Pseudo-code

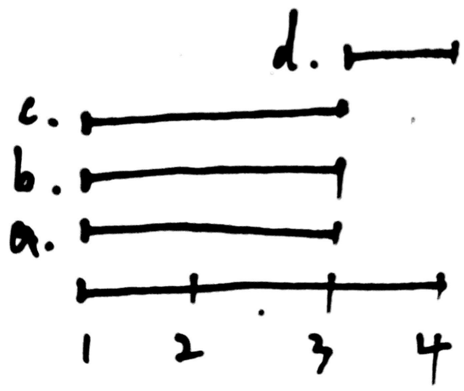
```
Initially H is the set of all houses
i = 0
A = ∅
$d_j = f_j - s_j$
for (i in n) {
        choose j ∈ H that $s_j <= i <= f_j$ and $d_j$ is smallest
        add j to A
}
return A
```

Running Time

- Add all houses in priority queue based on duration time is O(mlogm)
- Keep track of all days is O(n)
- Retrieve and eliminate the head of heap-based priority queue is O(logm), for all houses it is O(mlogm)
- Total time is O(n+mlogm)

Correctness Proof

Counterexample:



By Strategy 3, days [1, 2, 3, 4] will paint houses [a, b, d, null]. So, it is not an optimal algorithm.

Strategy 4

Algorithm

Earliest-End-Time-First: Iterate over each day starting from day 1, …, n. For each day, among the unpainted houses that are available that day, paint the house that will stop being available the earliest.

Pseudo-code

```
Initially, H is the set of all houses
i = 0
A = ∅
for (i in n) {
        choose j ∈ H that $s_j$<=i<=$f_j$ and $f_j$ is smallest
        add j to A
}
return A
```

Running Time

- Add all houses in priority queue based on finishing time is O(mlogm)
- Keep track of all days is O(n)
- Retrieve and eliminate the head of the heap-based priority queue is O(logm), for all houses the time complexity is O(mlogm)
- Total time is O(n+mlogm)

Correctness Proof

Let $g_1, \dots, g_k$ and $o_1, \dots, o_m$ be the sequence of houses painted by greedy and optimal algorithms respectively, ordered by increasing finishing time.

Lemma 1. For all i <= k, we have: $f_{gi} <= f_{oi}$.

Proof. By induction, we have $f_{g1}=f_{o1}$. Assume this holds for k-1, thus $f_{gk-1}<=f_{ok-1}$. For k-th house, note that $f_{ok-1}<=s_{ok}$. Using the inductive hypothesis, we have $f_{gk-1}<=f_{ok-1}<=s_{ok}$. The greedy algorithm picks earliest finish time among compatible houses, thus $f_{gk}<=f_{ok}$.

Lemma 2. The greedy algorithm returns an optimal set of houses S, that is, k=m.

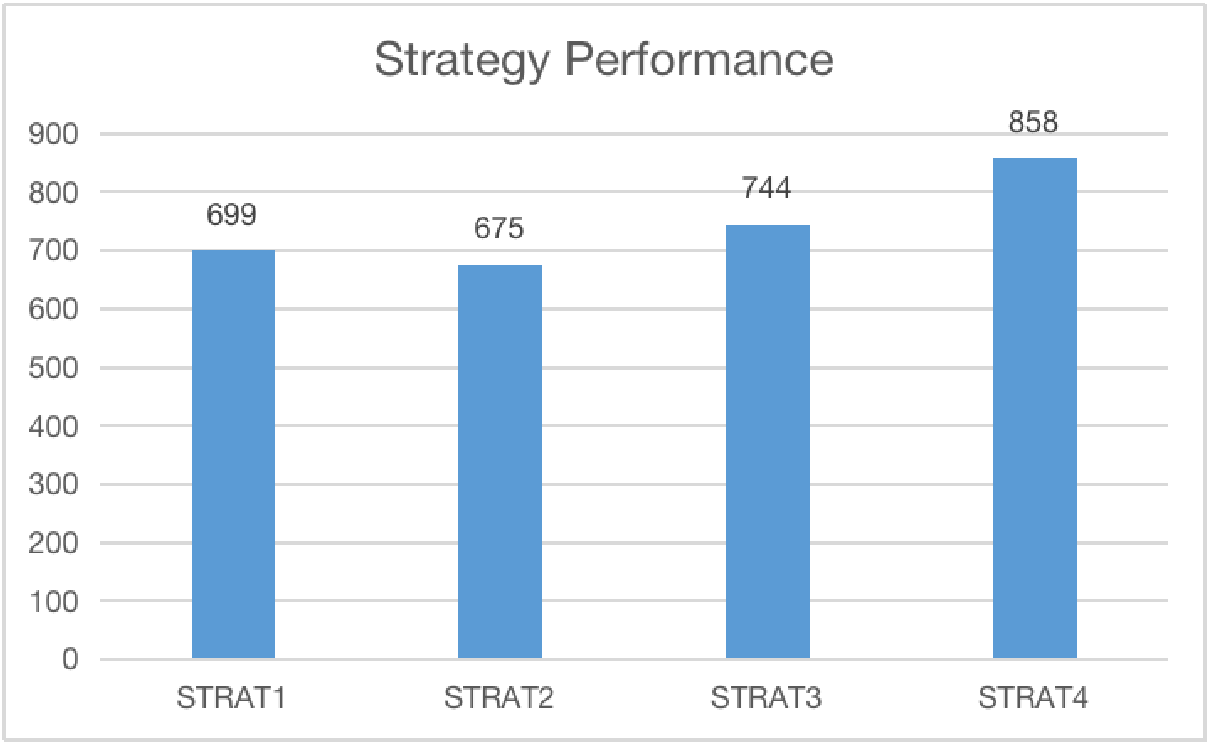
Proof. (By contradiction)

Suppose S is not optimal, then the optimal set O must paint more houses, that is, m>k. That means there is a house o_{k+1} that paints after o_k ends. Since $f_{gk}<=f_{ok}$, greedy will also paint house o_{k+1}.

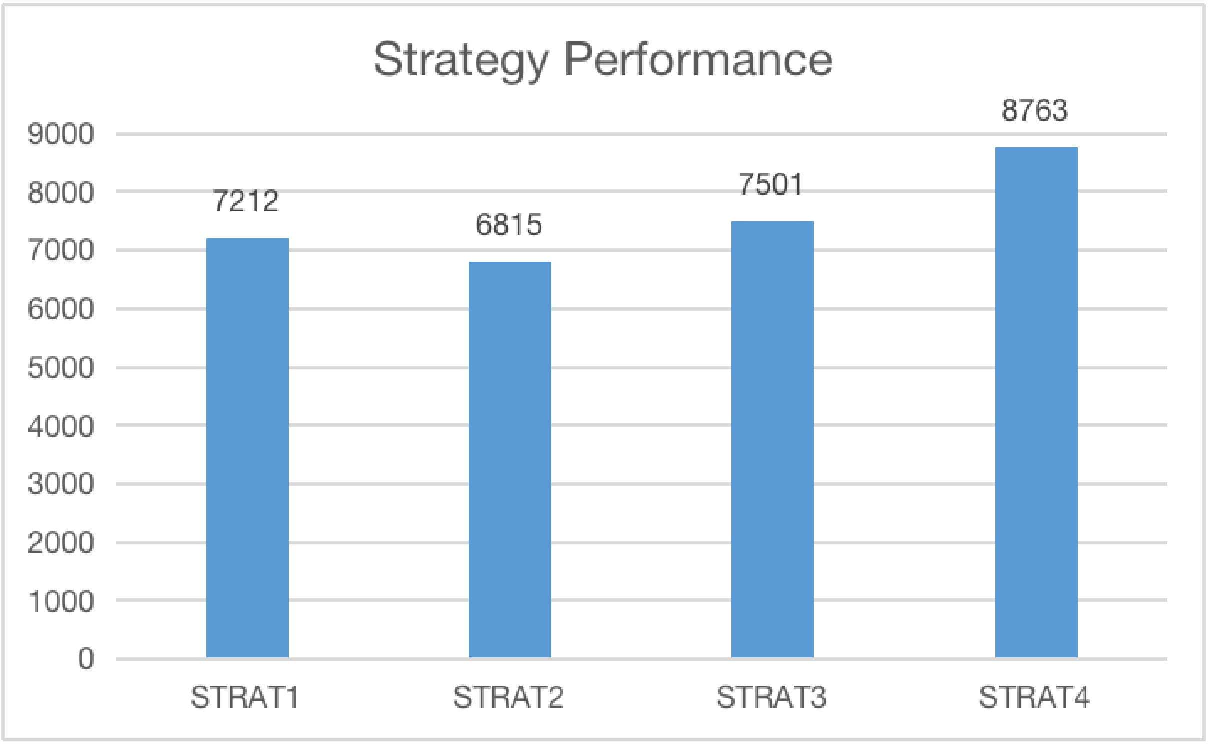
STRAT 1-4 Experiments

In order to verify the performance of different strategies, we set m and n to be [1000, 10000, 100000]. Among all these tests, the finish-time-first greedy algorithm STRAT4 can always find the most houses to paint. As we have proved above, STRAT 4 is the optimal greedy algorithm for this house painting problem.

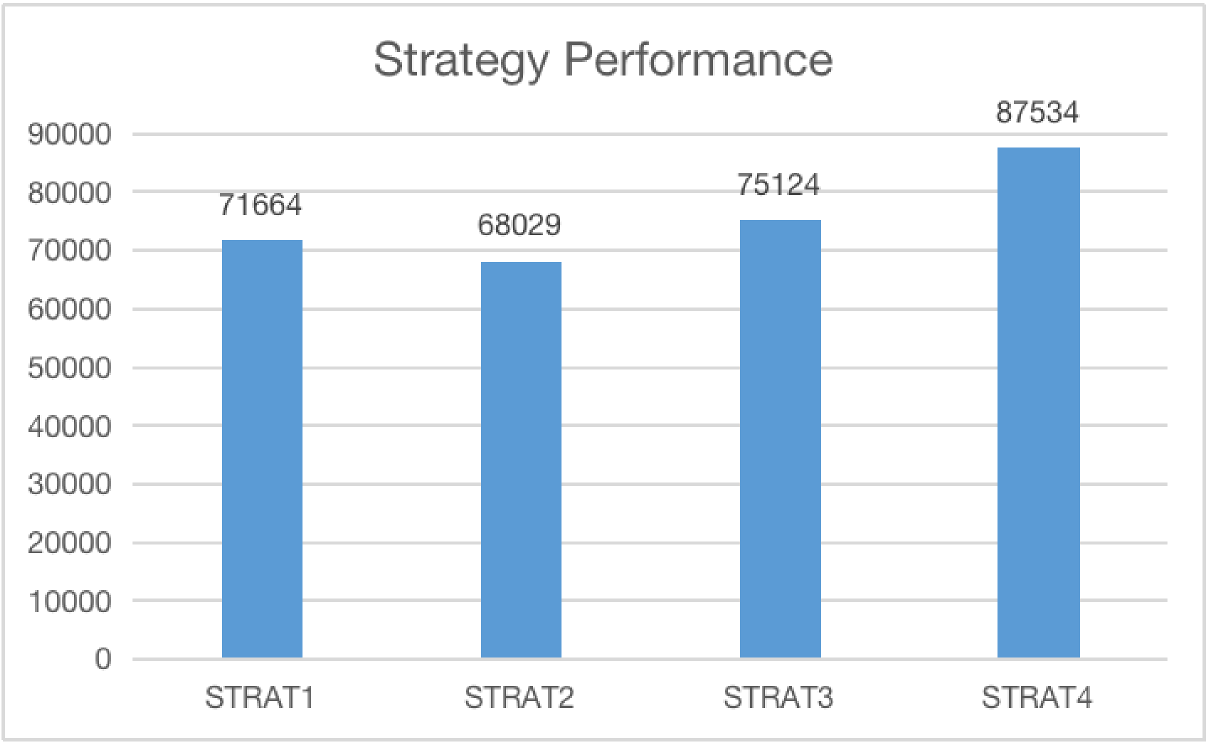
When m=1000, n=1000, the houses that can be painted are listed in the following.



When m=10,000, n=10,000, the houses that can be painted are listed in the following.



When m=100,000, n=100,000, the houses that can be painted are listed in the following.



Strategy 5

Algorithm

Iterate over each house by start day. For each start day, among the unpainted houses that are available that day, paint the house that will stop being available the earliest.

Pseudo-code

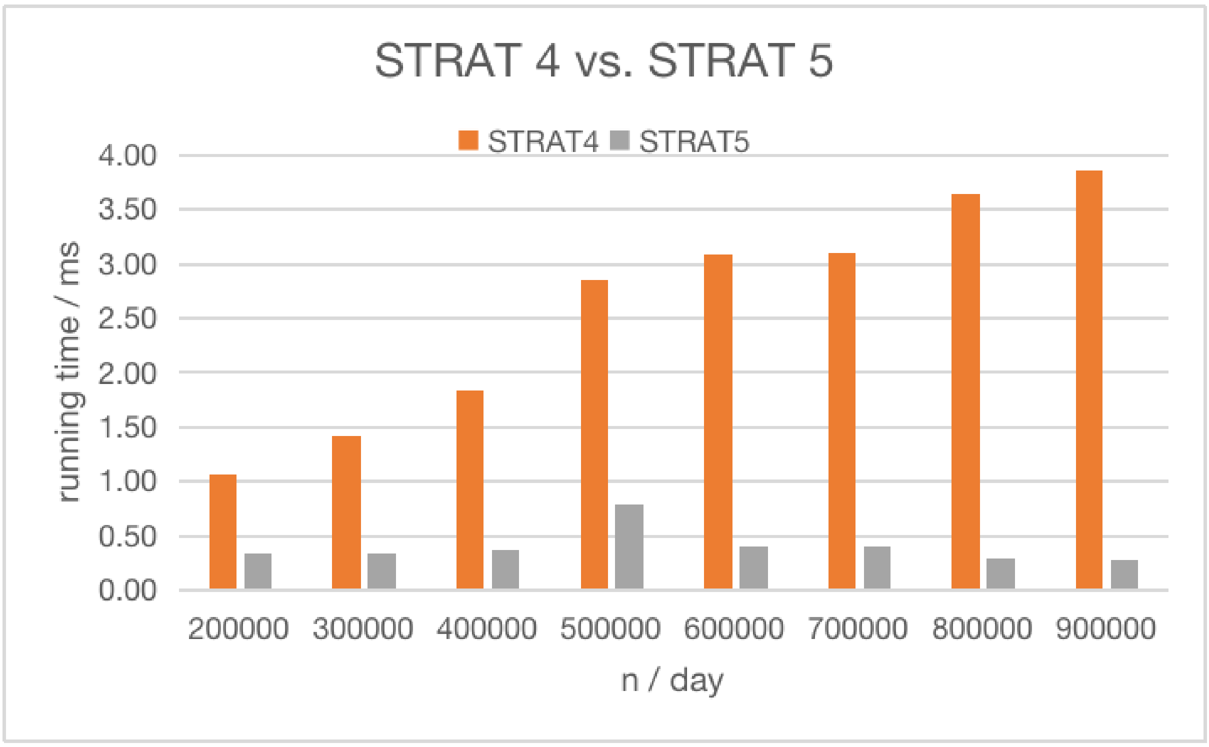
```
Initially, H is the set of all houses
A = $\varnothing$
for (day in start_day of H) {
        choose j $\in$ H that $f_j$>=i and $f_j$ is smallest
        add j to A
}
return A
```

Analysis

Time: Keep track of all houses in priority queue is O(mlogm).
Space: The space complexity of PriorityQueue and ArrayList is O(m).
Correctness: As proved in Strategy 4, the finish-time-first greedy algorithm is optimal.

Experiments

When m=1000, n is set to be [200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000]. The running time of STRAT 4 and 5 are shown in the following figure. As we have proved above, the time complexity of STRAT 4 is O(n + mlogm) and the time complexity of STRAT 5 is O(mlogm). The experimental results are the same as the theoretical analysis. Under a certain m, we can find that the running time of STRAT 4 is linear to n and the running time of STRAT 5 is a constant.



Conclusion

The primary research of this project is to set different priority queue sorting rules according to various sorting strategies and determine an appropriate time to put the house into the priority queue. Once it is determined when a house enters the priority queue, the only difference for strategy 2 strategy 3 strategy 4 is the priority queue sorting rules.

The main idea of the algorithm is to iterate over each day, and if startDay is equal to the current number of days when iterating, then put it inside the heap to ensure that there will not be an empty day without work. That is, we need to keep the startDay of the house inside the heap always earlier or equal to the current number of days. Now we get a heap. If the endDay of this house is less than the current number of days, then it means that this house is expired, so we will throw it away and continue to look for the next one in the heap to pop out; if the endDay is greater than the current number of days, that is, it is still within the construction period, and we can pop it out.

For the potential technical challenges, the main difficulty we encountered was how to customize the sorting rules for the priority queue. For each different strategy, we need to set a different comparator to satisfy the sorting rules. In addition, using LinkedList to keep the labels of the houses enables output of all the houses that can be painted relatively easily.