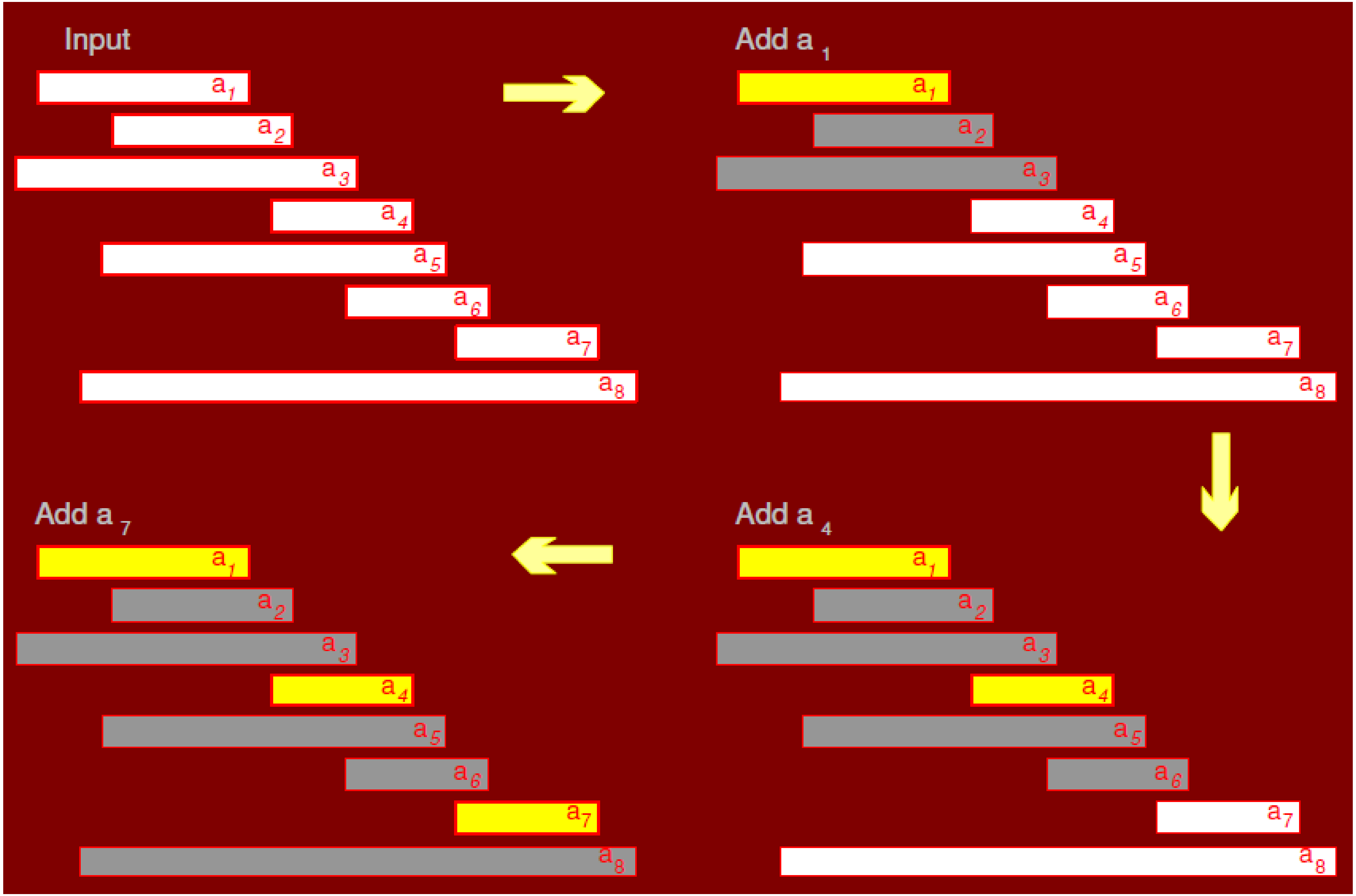
Greedy Algorithms
Activity Scheduling

(Class 26)

Greedy Algorithms - Activity Scheduling

- Here is a simple greedy algorithm that works:
- Sort the activities by their finish times.
- Select the activity that finishes first and schedule it.
- Then, among all activities that do not interfere with this first job, schedule the one that finishes first, and so on.

```
SCHEDULE(a[1..N])
sort a[1..N] by finish times
A ← {a[1]}             // schedule activity 1 first
prev ← 1               // most recently scheduled
for i ← 2 to N
  if (a[i].start ≥ a[prev].finish)
      then A ← A ∪ a[i]
      prev ← i
```



Input
a1
a2
a3
a4
a5
a6
a7
a8
Add a 1
Add a 4
Add a 7

- Figure shows an example of the activity scheduling algorithm.
- There are eight activities to be scheduled.
- Each is represented by a rectangle.
- The width of a rectangle indicates the duration of an activity.
- The eight activities are sorted by their finish times.

- The eight rectangles are arranged to show the sorted order.
  - Activity a_1 is scheduled first.
  - Activities a_2 and a_3 interfere with a_1 so they ar not selected.
  - The next to be selected is a_4.
  - Activities a_5 and a_6 interfere with a_4 so are not chosen.
  - The last one to be chosen is a_7.
  - Eventually, only three out of the eight are scheduled.

Running Time Analysis:

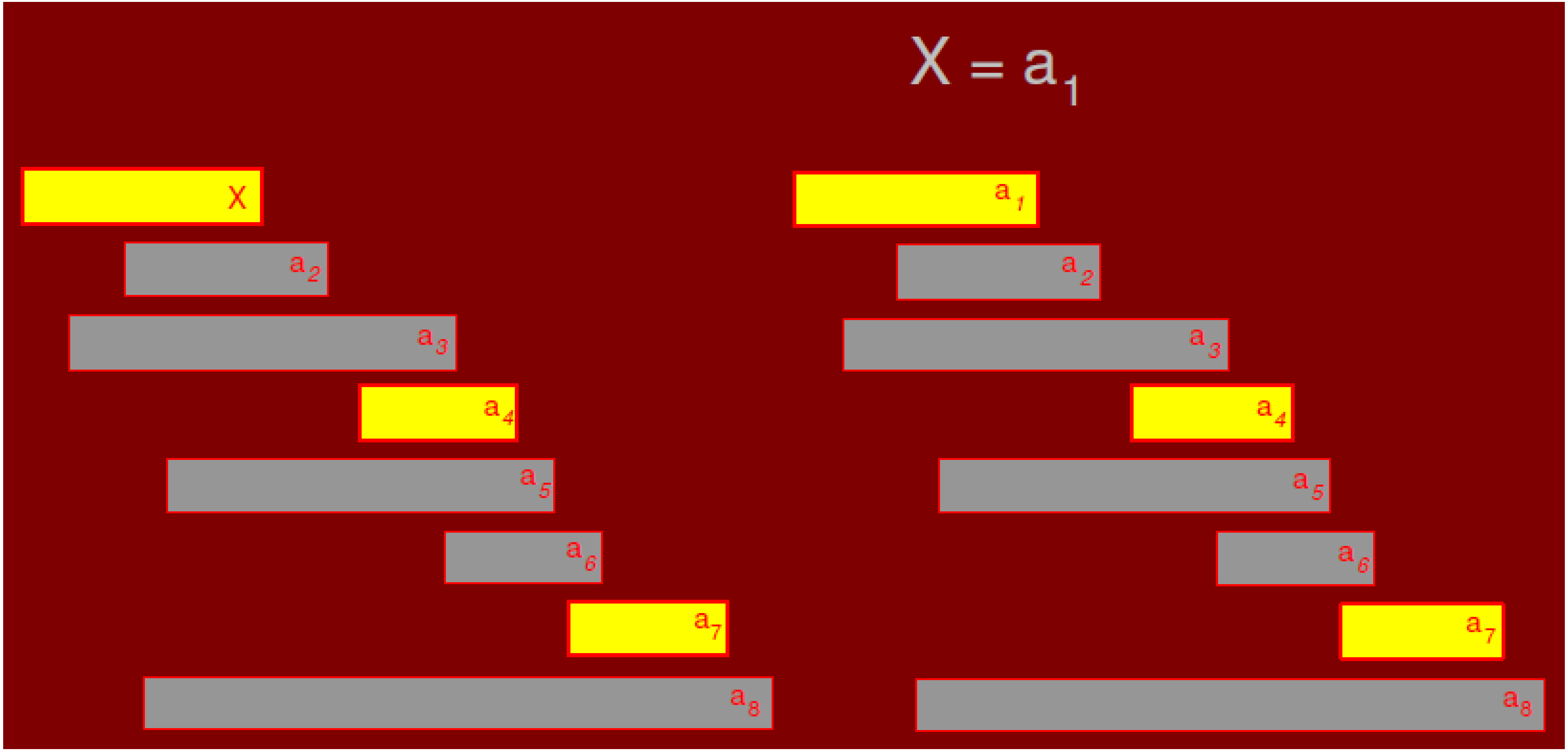
- Time is dominated by sorting of the activities by finish times.
- Thus, the complexity is:

$$O(n \log n)$$

Correctness of Greedy Activity Selection

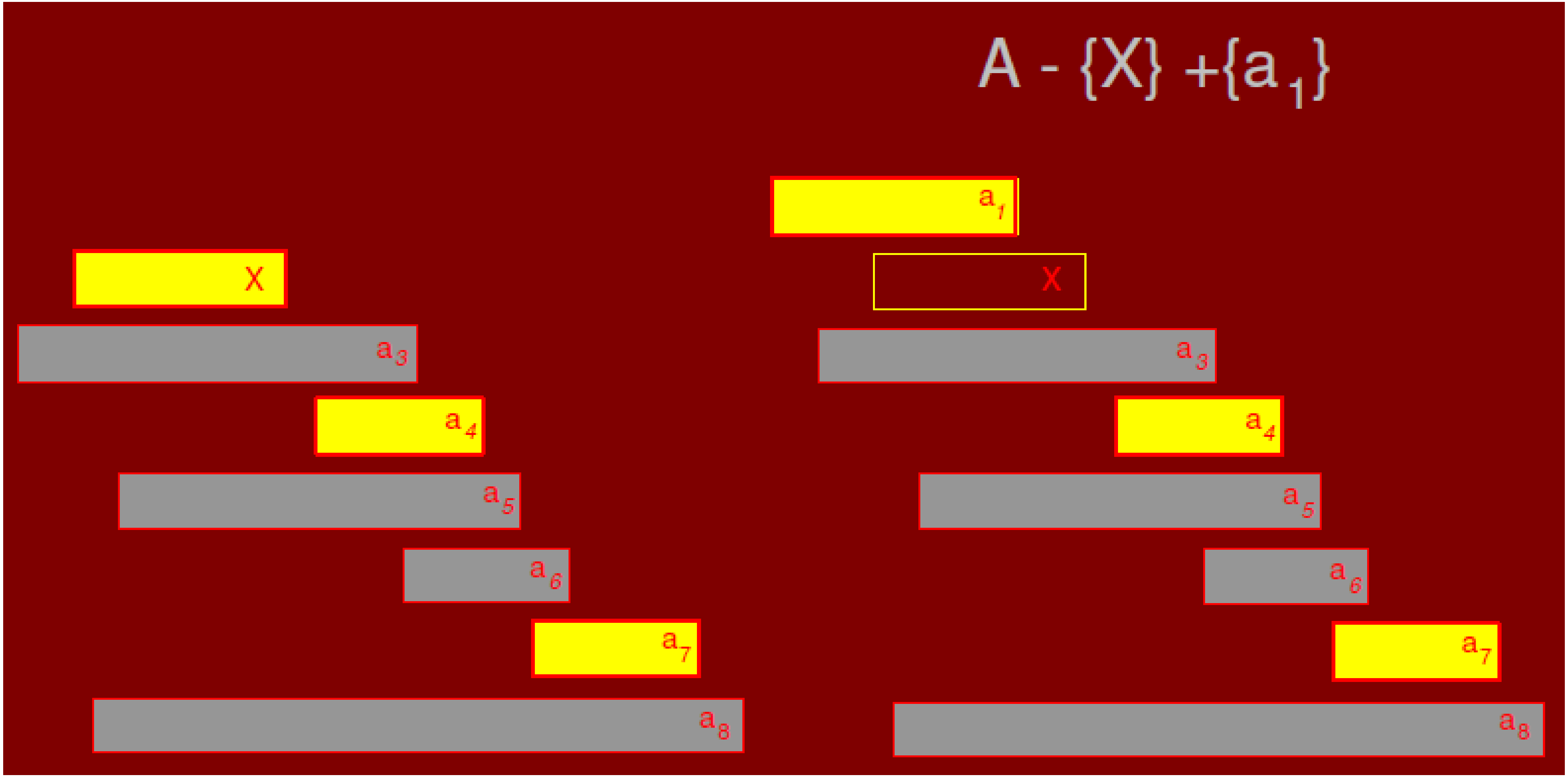
- Our proof of correctness is based on showing that the first choice made by the algorithm is the best possible.
- And then using induction to show that the algorithm is globally optimal.
- The proof structure is noteworthy because many greedy correctness proofs are based on the same idea:
  - Show that any other solution can be converted into the greedy solution without increasing the cost.

- **Claim:**
- Let $S = \{a_1, a_2, \ldots, a_n\}$ of n activities, sorted by increasing finish times, that are to be scheduled to use some resource.
- Then there is an optimal schedule in which activity a_1 is scheduled first.
- **Proof:**
- Let A be an optimal schedule.
- Let x be the activity in A with the smallest finish time.
- If $x = a_1$ then we are done.
- Otherwise, we form a new schedule A' by replacing x with activity a_1.



Activity $X = a_1$

- We claim that $A' = A - \{x\} \cup \{a1\}$ is a feasible schedule, i.e., it has no interfering activities.
- This because $A - \{x\}$ cannot have any other activities that start before x finishes, since otherwise, these activities will interfere with x.



New schedule A' by replacing x with ctivity a_1.

- Since a_1 is by definition the first activity to finish, it has an earlier finish time than x.
- Thus a_1 cannot interfere with any of the activities in $A - \{x\}$.
- Thus, A' is a feasible schedule.
- Clearly A and A' contain the same number of activities implying that A' is also optimal.

- **Claim:**
- The greedy algorithm gives an optimal solution to the activity scheduling problem.
- **Proof:**
- The proof is by induction on the number of activities.
- For the basis case, if there are no activities, then the greedy algorithm is trivially optimal.

- For the induction step, let us assume that the greedy algorithm is optimal on any set of activities of size strictly smaller than $|S|$ and we prove the result for S.
- Let S' be the set of activities that do not interfere with activity a_1, That is:

$$S' = \{\mathrm{a_i} \in S \mid s_i \geq \mathrm{f}_1\}$$

- Any solution for S' can be made into a solution for S by simply adding activity a_1, and vice versa.

- Activity a_1 is in the optimal schedule (by the above previous claim).
- It follows that to produce an optimal schedule for the overall problem, we should first schedule a_1 and then append the optimal schedule for S'.
- But by induction (since $|S'| < |S|$), this exactly what the greedy algorithm does.