Activity Selection Problem

Greedy Algorithm

An Activity Selection Problem (Class Scheduling Problem)

- **Input: A set of activities S = {$a_1, a_2, \ldots, a_n$}**
- Each activity i has start time (s_i) and a finish time (f_i)
- *Duration of activity* a_i=[f_i - s_i]
- Two activities i and j are compatible if and only if their interval does not overlap ($s_i \geq f_j$ or $s_j \geq f_i$)
- **Output: a maximum-size subset of mutually compatible activities**

The Activity Selection Problem

- Here are a set of start and finish times

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

- What is the maximum number of activities that can be completed?
  - $\{a_3, a_9, a_{11}\}$ can be completed
  - But so can $\{a_1, a_4, a_8, a_{11}\}$ which is a larger set
  - But it is not unique, consider $\{a_2, a_4, a_9, a_{11}\}$

Lets try some greedy strategy (schedule shortest class first)

1. Sort by duration of each class in increasing order
2. Select and schedule shortest class
3. Select next shortest class whose start time is greater or equal to finish time of last scheduled class
4. Repeat Step 3 until activities finish

Is this greedy Algorithm correct ?

The Activity Selection Problem

- Here are a set of start and finish times

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

- Select shortest duration first
  - Greedy solution = {a_2, a_4, a_8, a_{11} }

The shortest duration greedy choice is not correct

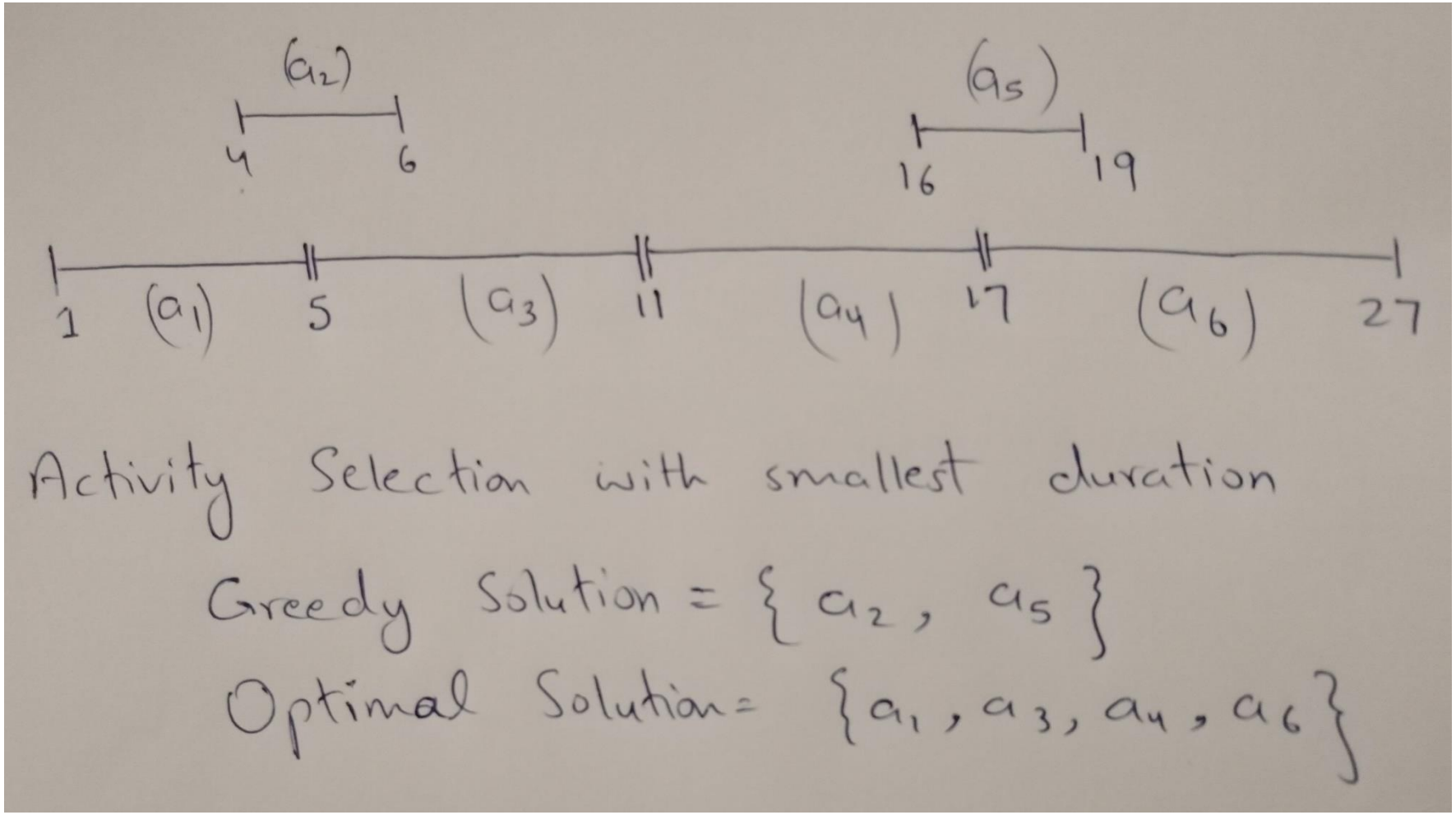
Counter example

Activity i	1	2	3	4	5	6
Start time$_i$	1	4	5	11	16	17
Finish time$_i$	5	6	11	17	19	27
duration	4	2	6	6	3	10

Greedy: Sort by duration: selected activities = {a2, a5}

Best Solution = 4 activities = {a1, a3, a4, a6}

The shortest duration greedy choice is not correct

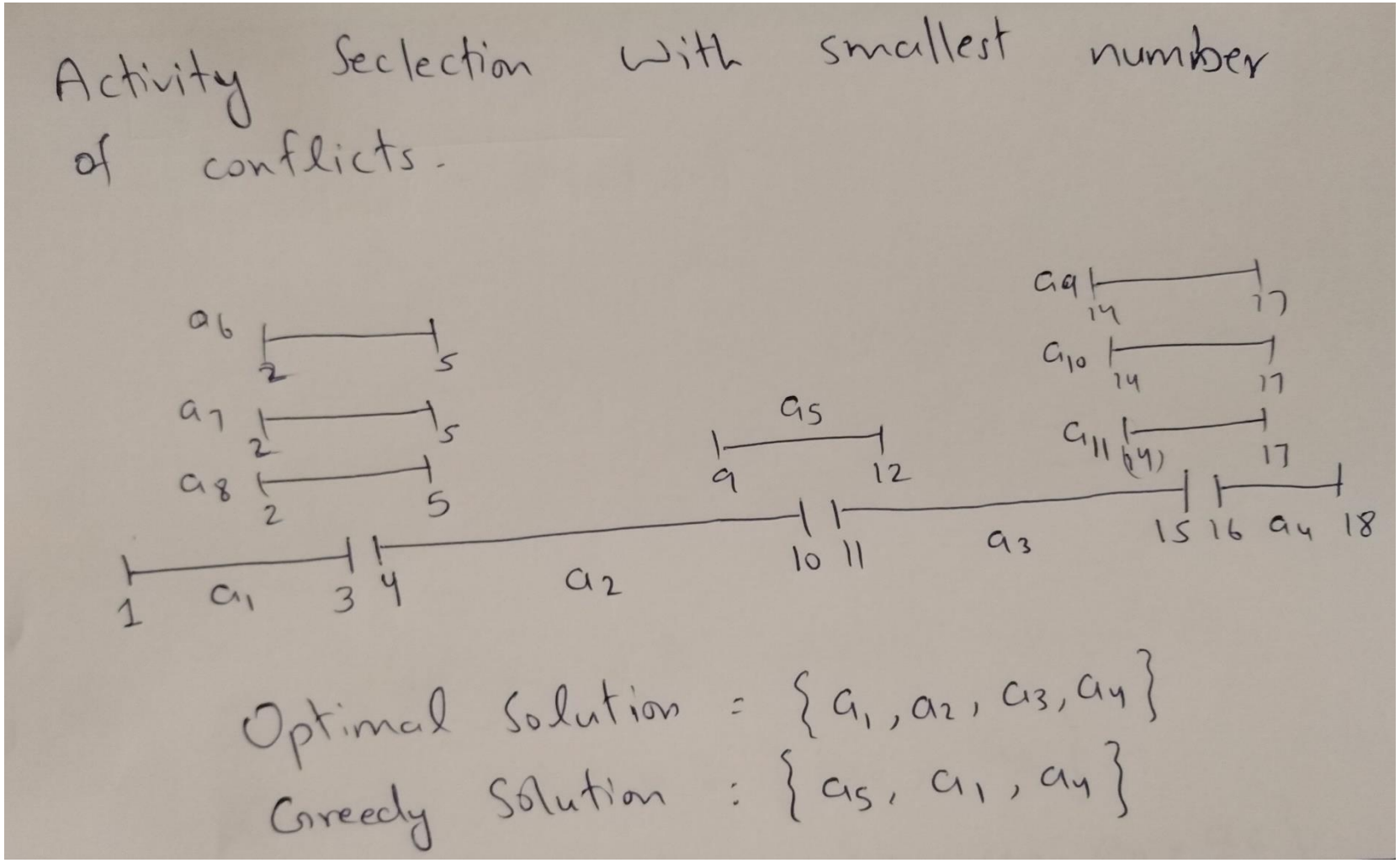


Activity Selection (another greedy algorithm)

- Select the activity with smallest number of conflicts with other activities

Is this greedy algorithm correct?

Activity Selection (another greedy algorithm)



Early Finish Greedy Approach

- Select the activity with the earliest finish
- Eliminate the activities that could not be scheduled
- Repeat!

Recursive Greedy Algorithm

It assumes that the input activities are ordered by monotonically increasing finish time (sorted)

```
RECURSIVE-ACTIVITY-SELECTOR(s, f, k, n)
1  m = k + 1
2  while m ≤ n and s[m] < f[k]        // find the first activity in S_k to finish
3      m = m + 1
4  if m ≤ n
5      return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
6  else return ∅
```

Iterative Greedy Algorithm

It also assumes that the input activities are ordered by monotonically increasing finish time (sorted)

```
GREEDY-ACTIVITY-SELECTOR(s, f)
1  n = s.length
2  A = {a_1}
3  k = 1
4  for m = 2 to n
5      if s[m] ≥ f[k]
6          A = A ∪ {a_m}
7          k = m
8  return A
```

Time Complexity of Greedy Activity Selection Algorithm

- O(n lg n) time for sorting, O(n) time for scheduling activities
- Overall time = O (n lg n)

Greedy Algorithm (Dry Run)

Activities are sorted by finish time

k	s_k	f_k	
0	-	0	
1	1	4	k = 1
2	3	5	
3	0	6	
4	5	7	k = 4
5	3	8	
6	5	9	
7	6	10	
8	8	11	k = 8
9	8	12	
10	2	13	
11	12	14	k = 11

GREEDY-ACTIVITY-SELECTOR(s, f)

```
1  n = s.length
2  A = {a_1}
3  k = 1
4  for m = 2 to n
5      if s[m] ≥ f[k]
6          A = A ∪ {a_m}
7          k = m
8  return A
```

Selected activities = $\{a_1, a_4, a_8, a_{11}\}$

Why it is Greedy?

- Greedy in the sense that it leaves as much opportunity as possible for the remaining activities to be scheduled
- The greedy choice is the one that maximizes the amount of unscheduled time remaining