Date: 26/04/21

Black Board

Design and Analysis of Algorithms

Topics:

- Greedy Algorithms I
  - 0/1 and Fractional Knapsack
  - The Activity Selection Problem

The 0/1 Knapsack Problem

- **Input:** A set I of n items, each with a value v_i and a weight w_i, where $1 \le i \le n$; and a knapsack capacity W.

I

- **Output:** A selection of $k \le n$ items $S \subseteq \cancel{W}$, such that $W_S = \sum_{i=1}^{k} w_i \le W$ and $V_S = \sum_{i=1}^{k} v_i$ is maximal.
  - Constraint: Any item may be picked **whole** or not picked. That is, the choice is 0/1.
  - No item may be repeated.

Brute Force Algorithm

Item #	Value	Weight
1	$20	2
2	$30	5
3	$32	7
4	$10	3
5	$3	1
6	$4	2

W=9

0/1

0/1

0/1

Item #

$\frac{\text{Item \#}}{3}$

9

$O(2^n)$

$2 \times 2 \times 2 \cdots = 2^n$

Does Greedy Approach work?

Item #	Value	Weight	Value/Weight
1	$20	2	10
2	$30	5	6
3	$32	7	4.57
4	$10	3	3.33
5	$3	1	3
6	$4	2	2

W=9

Item # : 1, 2, 5 = 53

8

The Fractional Knapsack Problem

- **Input:** A set I of n items, each with a value v_i and a weight w_i, where $1 \leq i \leq n$; and a knapsack capacity W.
- **Output:** A selection of $k \leq n$ items $S \subseteq W$, such that $W_S = \sum_{i=1}^{k} w_i \leq W$ and $V_S = \sum_{i=1}^{k} v_i$ is maximal.
  - **Constraint: An item may be picked whole *or fractionally*.**
  - No item may be repeated once fully exhausted.

Does Greedy Approach work on fractional Knapsack?

Item #	Value	Weight	Value/Weight
1	\$20	2	10
2	\$30	5	6
3	\$32	7	4.57
4	\$10	3	3.33
5	\$3	1	3
6	\$4	2	2

W=9

Greedy Choice and the Greedy Choice Property

- A Greedy Choice is said to have the Greedy Choice Property if it leads to a global optimal solution.
  - So in order to show that a greedy choice has the greedy choice property we need to give a proof that this is the case.
  - This is the proof of correctness of greedy algorithm.

Greedy Choice Property for the Greedy Fractional Knapsack Algorithm

Item #	Value	Weight		Value/Weight
1	\$20	2	1+1	10
2	\$30	5	1+1+1+1+1	6
3	\$32	7	7*1	4.57
4	\$10	3	1+1+1	3.33
5	\$3	1	1	3
6	\$4	2	1+1	2

W=9

Consider the items as list of unit weight items.

→ 10, 10, 6, 6, 6, 6, 6, 4.57, 4.57, ...

W

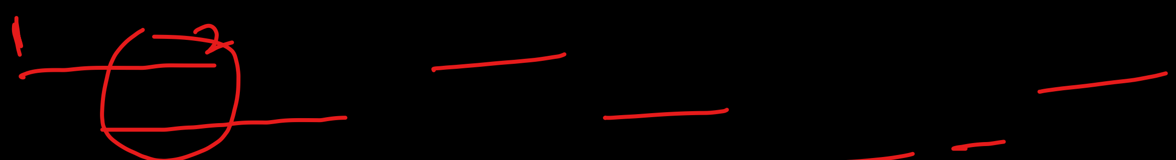
The Activity Selection Problem

Example 1: You are at a conference for a whole day with multiple talks scheduled in different halls. You can only either listen to a whole talk or skip it entirely. The talks are of varying durations. You wish to attend as many talks as possible (a case of quantity over quality!). You can obviously not take two talks at the same time. Which talks should you attend?

the maximal set of non-overlapping talks.

The Activity Selection Problem

Example 2: You work at the academic office of a university. The university is preparing a time table for a new semester. The classes have already been given time slots. The best room on campus is the Saminaninaar Hall. It would be best if Saminaninaar Hall is used to its fullest capacity, i.e. maximum possible classes may be held there. Which classes will you choose?



The Activity Selection Problem

Example 3: What if the classes were scheduled processes, and the Saminaninaar Hall was a computational resource such as a cloud server whose GPUs need to be fully utilized?

(What is the Saminaninaar Hall was a diligent programmer whose talent needed to be fully exploited by a cunning project manager?)

The Activity Selection Problem

$s[i]$, $f[i]$

- **Input:** A set A of size n containing $s[1 \ldots n]$ and $f[1 \ldots n]$: the start and finish times of n activities.
- **Output:** A selection $A' \subseteq A$ of non-overlapping activities, such that $|A|$ is maximal.

non overlap

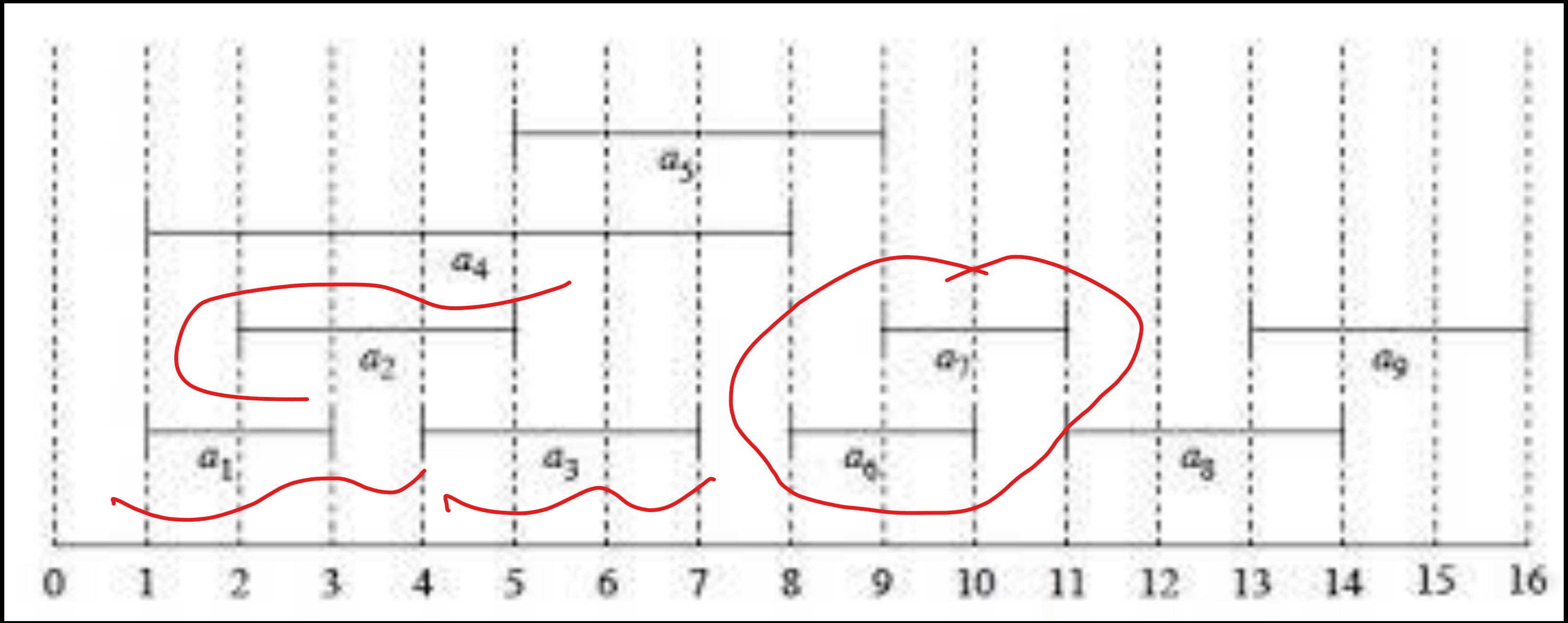
Note: two activities overlap iff the size of their intersection is more than 1.

$f_i \leq s_j$ or $f_j \leq s_i$

f_i

s_j

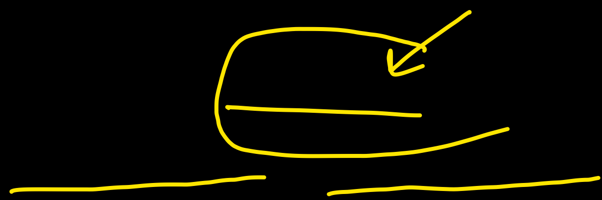
Visualizing Activities



What kind greed won't work?

1. Shortest activity first.

doesn't work

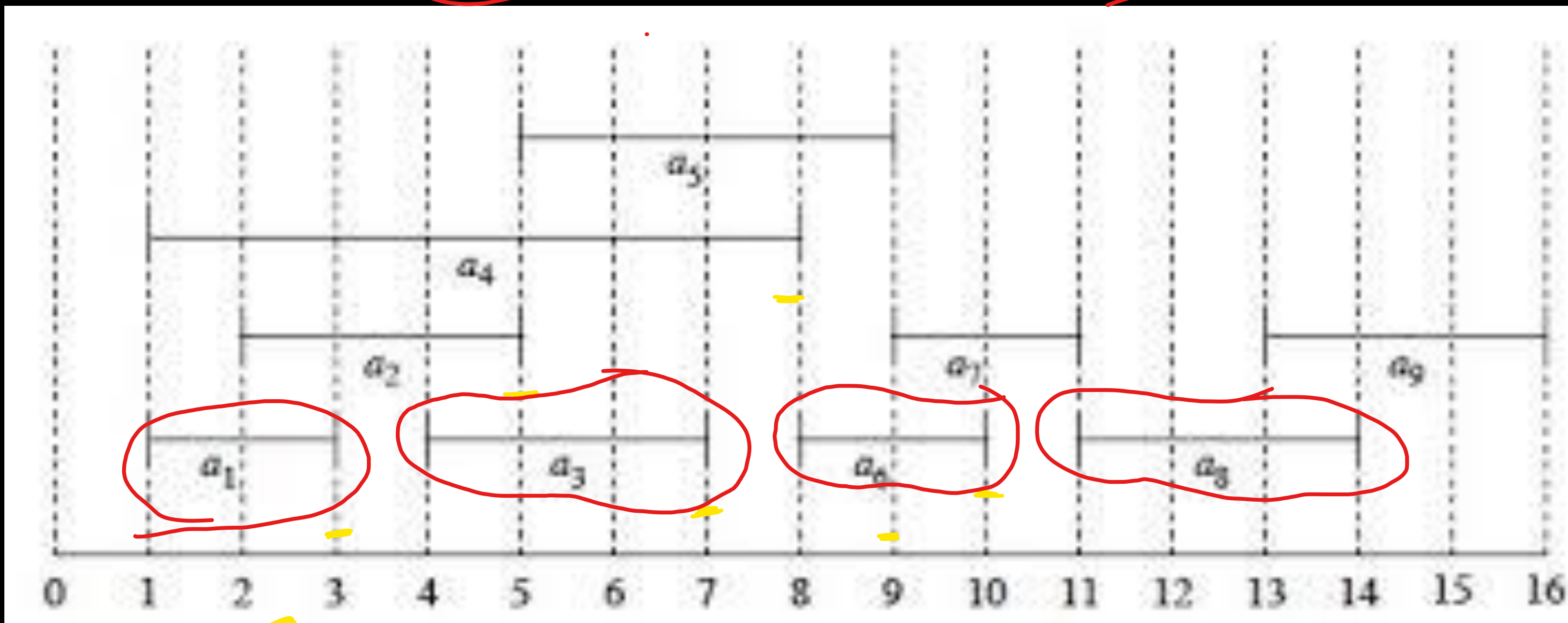


2. Sort activities on starting times

→ ~~Pick~~ sequentially. pick the next non-overlapping activity.

The Greed that works

a_1, a_3, a_6, a_8 optimal sol.



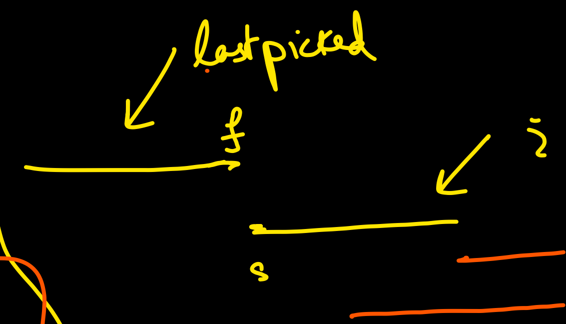
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

→ Sort activities on their finish times

The Greedy Algorithm

```
GreedyActSelect(A, n)
    SortOnFinishTimes(A, n)          → O(n lg n)
    X = insert(A[1])
    lastPicked ← 1
    For i = 2 to n
        IF(A[lastpicked].f ≤ A[i].s)
            X.insert(A[i])
            lastPicked ← i
    return X
```

$O(n)$



$T(n) = O(n \lg n)$

The Greedy Choice Property

a_1 α_2

α_3

a_2

α_4 . . .

α_k

$\cancel{a_1}$

a_2

a_3 . . . a_m