\[ \{2, 3, 4, 5, 21, 5, 13, 8\} \]

\[ (A_1 \times A_2) (A_3 \times A_4) (A_5 \times A_6) \]

\[ 2 \times 3 \quad 3 \times 5 \quad 5 \times 8 \]

greedy approach
pick multiplication to get
rid of the biggest number
won't be optimal
Suppose we have a set \( S = \{ a_1, a_2, \ldots, a_n \} \) of proposed activities that wish to use a resource, such as a lecture hall, which can only accommodate one activity at a time. Each activity \( a_i \) has a start time \( s_i \) and a finish time \( f_i \), with \( f_i \geq s_{i-1} \) for all \( i \geq 2 \).

We shall see later the advantage that this assumption provides. (For example, consider the following set of activities:

\[
\begin{array}{cccccccc}
4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
14 & 12 & 11 & 10 & 9 & 6 & 5 & 2 \\
4 & 3 & 2 & 1 & 1 & 1 & 1 & 1
\end{array}
\]

\( \text{if } f_1 \geq f_2 \geq \cdots \geq f_3 \geq f_2 \geq f_1 \)

In monotonically increasing order of finish time:

In the activity-selection problem, we wish to select a maximum-size subset of mutually compatible activities. We assume that the activities are sorted by their finish times. If \( f_i \leq s_j \) for all \( j < i \), then activity \( a_i \) is mutually compatible with all other activities, and activity \( a_i \) can be chosen. If activity \( a_i \) overlaps with any other activity, it cannot be chosen. A feasible solution is a set of activities that are mutually compatible and have the maximum possible number of activities.

Suppose we have a set of activities \( a_1, a_2, \ldots, a_n \) and we wish to select a maximum-size subset of mutually compatible activities. We can use a greedy algorithm to solve this problem. The algorithm works as follows:

1. Sort the activities in decreasing order of finish times.
2. Select the first activity in the sorted list. This activity is mutually compatible with all activities that come after it.
3. For each subsequent activity, check if it is mutually compatible with the already selected activities. If it is, add it to the selected set.
4. Repeat step 3 until there are no more activities that can be added to the selected set.

The algorithm guarantees that the selected set of activities is a maximum-size subset of mutually compatible activities.
Knapsack Problem

Greedy Algorithms

For fractional knapsack, the setup is the same, but the thief can take item more than once.

In the fractional knapsack problem, the setup is the same, but the thief can take a fractional amount of an item or take an item if or leave it behind. He cannot take a fractional amount of an item or take an item if or leave it behind. He cannot take a fractional amount of an item or take an item if or leave it behind. He cannot take a fractional amount of an item or take an item if or leave it behind.
Huffman Algorithm

Huffman-Tree

\( \text{Huffman}(C) \)

Each character has a freq tree

Children in the tree

\( \text{freq} = x \cdot \text{freq} + y \cdot \text{freq} \)

\( \text{freq} \)

Allocate a new node \( z \)

\( \text{left} = x \)

\( \text{right} = y \)

\( \text{extract-min}(\emptyset) \)

\( \text{insert}(z, \emptyset) \)

Return the root of the tree