Lab4 Solution

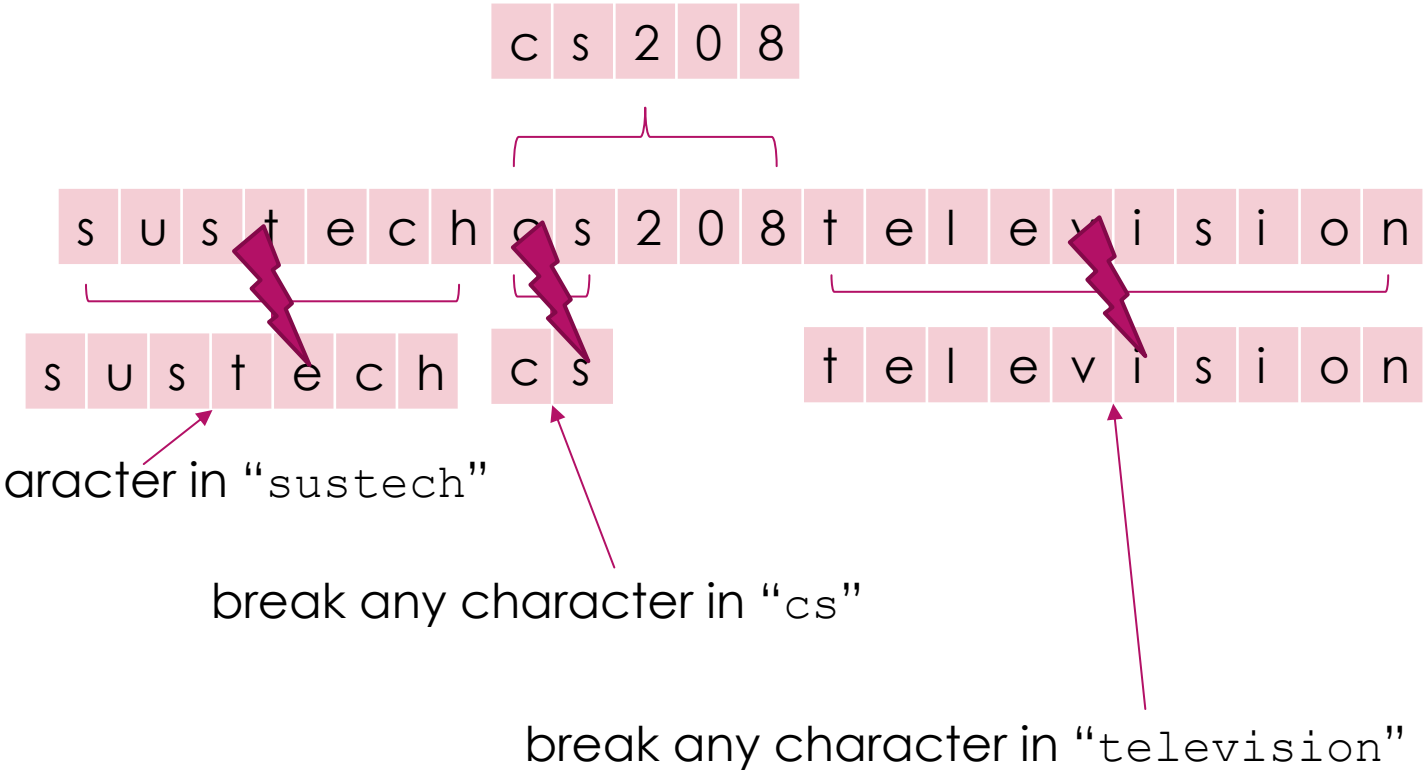
YAO ZHAO

Lab4.A TV Breaker

- Recently, Bob bought a TV. The TV can be considered as a string S. **A TV is called wonderful if its string contains at least one amazing feature.** There are n amazing features in total, and each one of the features is a string f_i. Alice is very jealous of Bob's TV. So one day Alice goes to Bob's home and starts breaking the TV.
- Alice can only do one hitting for one time, each hitting she can choose one character of the TV string and make it become space. Now what she wants is to make Bob's TV become no wonderful. That is, after all her hittings, the TV will not contain any one of the amazing features.
- Alice doesn't want to be very tired, so please help her find the minimum times of hittings to achieve her goal.
- The alphabet of the strings consists of characters and the value of ASCII in decimal is from 33 to 126.

Input:

```
sustechcs208television
4
cs
television
cs208
sustech
```

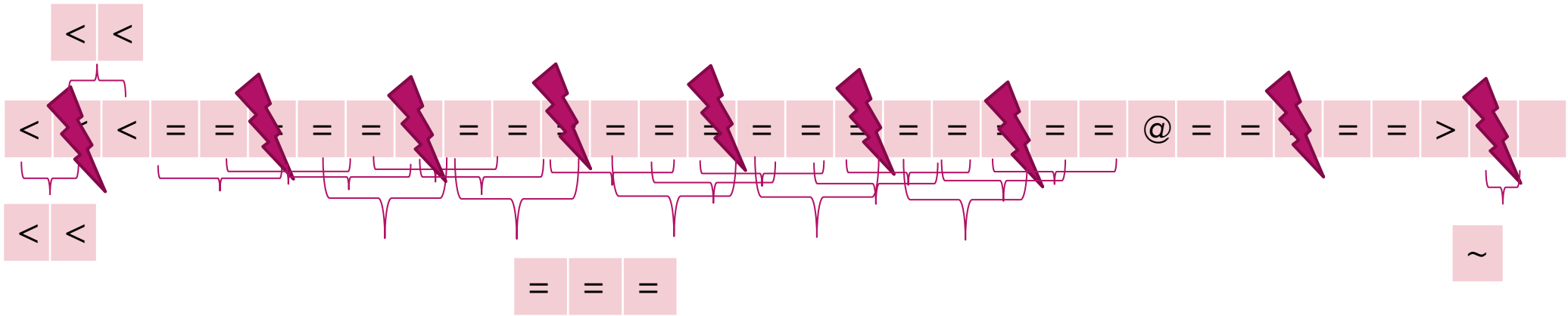


Output:

3

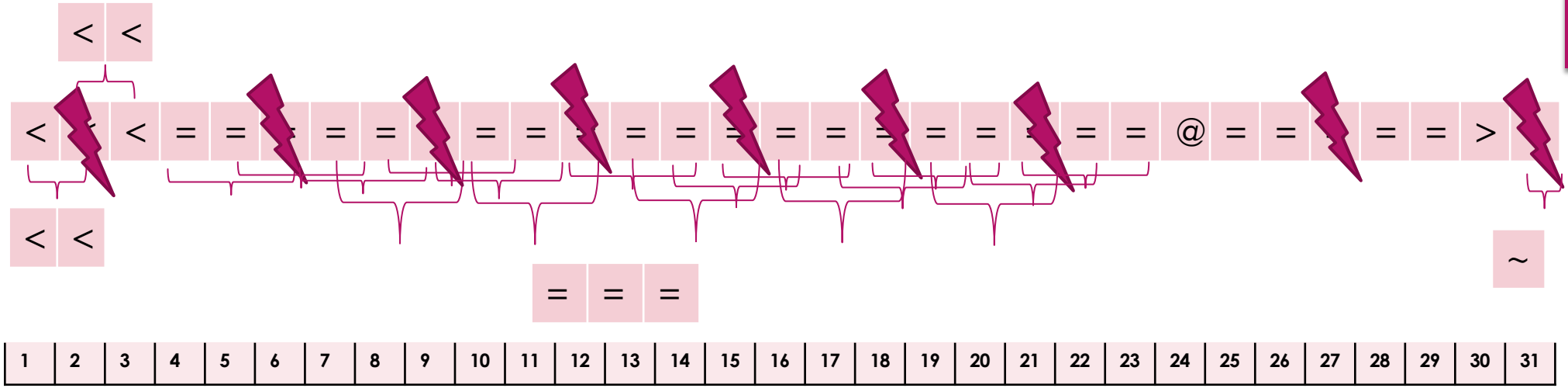
Input:

```
<<<====================@=====>~
3
<<
~
===
```



Output:

9



First: find all intervals that match the feathers (**KMP**)

<< ⟶ [1,2], [2,3]

~ ⟶ [31,31]

=== ⟶ [4,6],[5,7],[6,8],[7,9],[8,10],[9,11],[10,12],[11,13],[12,14],[13,15],[14,16],
[15,17],[16,18],[17,19],[18,20],[19,21],[20,22],[21,23],[25,27],[26,28],[27,29]

Then, this problem is transformed into an interval coverage problem:

Given n intervals [li,ri] in axis, find the minimum points in axis that each interval contains at least one point.

That's what Practice 5 asks you to do.

Practice 5: Interval Coverage Problem

output points set ← ø
p ← -∞
Sort the intervals by endpoint, and process them in endpoint order
For each interval
    If the start point of the current interval is larger than p
        add the endpoint of the current interval to the output points set
        p ← the endpoint of the current interval
    End If
End For

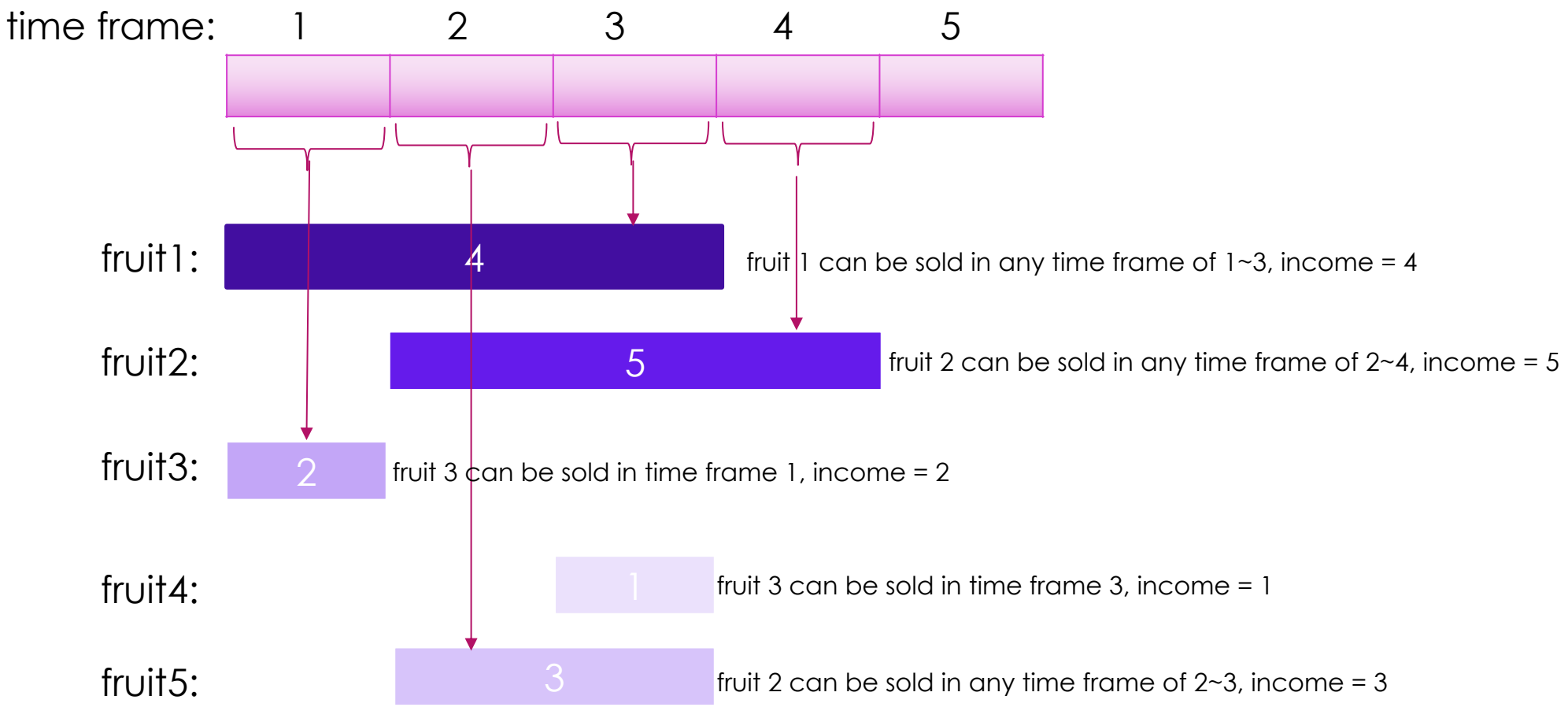
Lab4.B Fruit

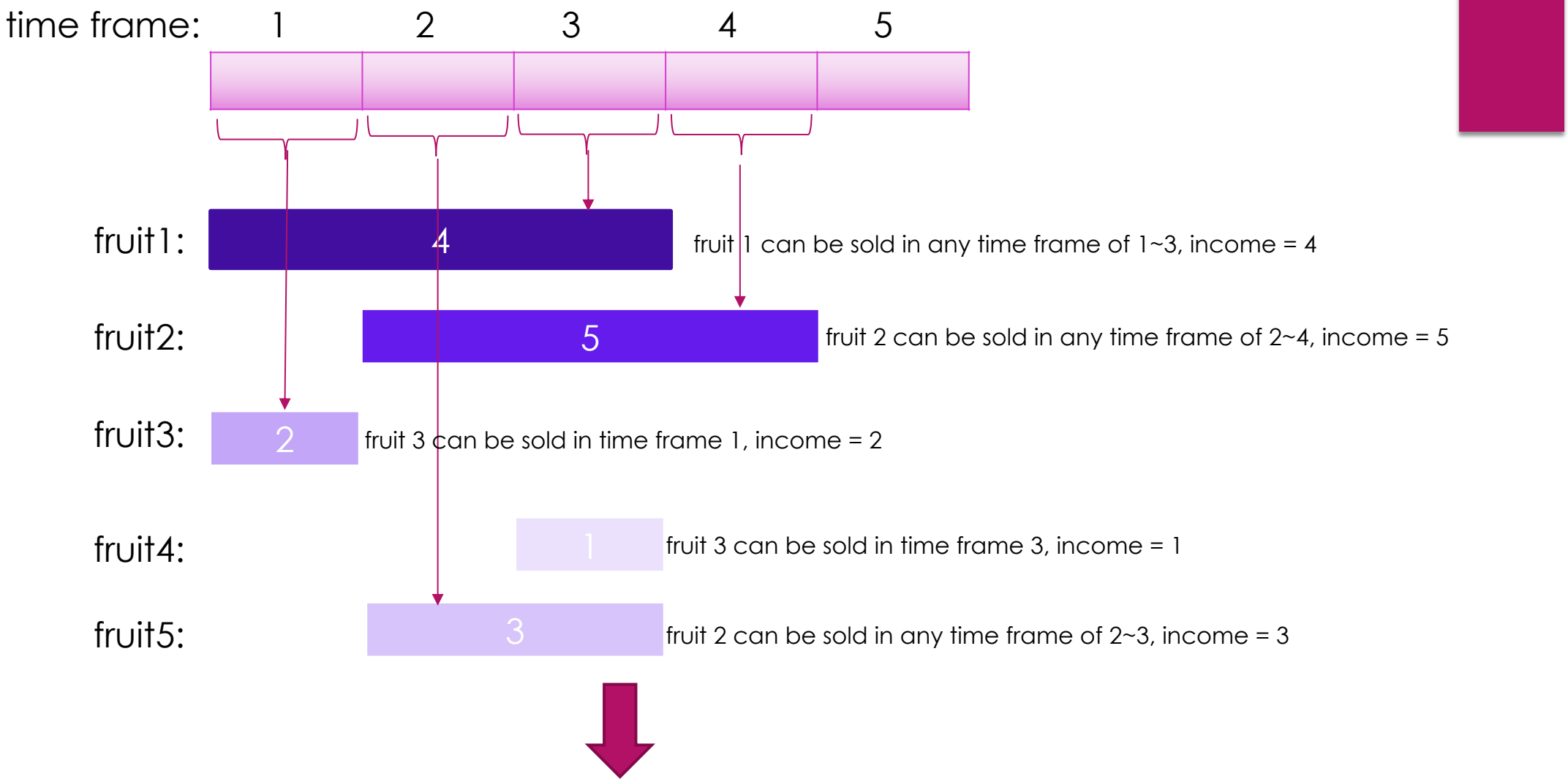
- Little Liu has a plantation where he grows n types of fruits. Each type of fruit can be sold for a certain price v_i, but only within a specific time frame from l_i to r_i. Little Liu can only sell one type of fruit at a time. Your task is to calculate the maximum amount of money he can earn from his plantation.

Note: n≤5000，$1 \le l_i \le r_i \le 10^9$，$1 \le v_i \le 10^9$

Input:

```
5
1 3 4
2 4 5
1 1 2
3 3 1
2 3 3
```




time frame: 1 2 3 4 5
fruit1: 4
fruit 1 can be sold in any time frame of 1~3, income = 4
fruit2: 5
fruit 2 can be sold in any time frame of 2~4, income = 5
fruit3: 2
fruit 3 can be sold in time frame 1, income = 2
fruit4: 1
fruit 3 can be sold in time frame 3, income = 1
fruit5: 3
fruit 2 can be sold in any time frame of 2~3, income = 3
time frame: 1 2 3 4 5
fruit3 fruit5 fruit1 fruit2
2 + 3 + 4 + 5 = 14

Output:

14

Judgment: Greedy algorithm can be used to solve this problem?

Claim 1 and proof

Claim 1: Sort fruits by price so that $v_1 \geq v_2 \geq \ldots \geq v_n$, consider each fruit in turn.
Let the previously selected fruits set be S. If the current fruit a_i can be added to S to ensure that every fruit matches a timeslot successfully, then select a_i, otherwise do not select a_i. This greedy strategy is bound to result in the maximum profit.

Proof:

1. Let the i^{th} fruit be a_i and the optimal fruit set of the previous i-1 fruits be S.
2. If a_i can be added to S to ensure that every fruit in S matches a timeslot, then the i^{th} fruit must be selected, otherwise, the total profit will not be optimal.
3. If after joining a_i, some fruits in S incur conflicts, that means at least one fruit in S can't be selected. As $v_1 \geq v_2 \geq \ldots \geq v_i$, a_i has the least profit, and the original fruits in S have no conflict, so do not select a_i.
4. Every selection can guarantee that the obtained set S is optimal, so the result is optimal.

Claim 1 and proof

Claim 1: Sort fruits by price so that $v_1 \geq v_2 \geq \ldots \geq v_n$, consider each fruit in turn.
Let the previously selected fruits set be S. If the current fruit a_i can be added to S to ensure that every fruit matches a timeslot successfully, then select a_i, otherwise do not select a_i. This greedy strategy is bound to result in the maximum profit.

Proof:

1. Let the i^{th} fruit be a_i and the optimal fruit set of the previous i-1 fruits be S.
2. If a_i can be added to S to ensure that every fruit in S matches a timeslot, then the i^{th} fruit must be selected, otherwise, the total profit will not be optimal.
3. If after joining a_i, some fruits in S incur conflicts, that means at least one fruit in S can't be selected. As $v_1 \geq v_2 \geq \ldots \geq v_i$, a_i has the least profit, and the original fruits in S have no conflict, so do not select a_i.
4. Every selection can guarantee that the obtained set S is optimal, so the result is optimal.

which timeslot would you place the elected fruits in?

$1 \leq l_i \leq r_i \leq 10^9$

How to choose the timeslot since the time range is too large?

Active Timeslots

Initially, all time slots are marked black. Then, for each fruit a_i (l_i, r_i, v_i), find the smallest k, where $k \geq l_i$ and the time slot k is black, and then mark the time slot k as white. Finally, we get exactly n time slots marked white, and these n time slots are useful time slots. Call them "active timeslots" .

Algorithm of Finding all Active Timeslots

S ← ø
Sort the fruits by start time so that $l_1 \leq l_2 \leq l_3 \ldots \leq l_n$
x ← 0
For i=1 to n {
    x ← max(x+1, l_i);
    add x to S
}

Claim 2 and proof

Claim 2. Suppose a timeslot set T and a fruit set S, if each fruit in S can schedule no conflict on T, there must be such a greedy strategy: scan each timeslot in T from front to rear, for each timeslot, if there are some fruits whose $l_i \leq$ the timeslot $\leq r_i$ and haven't be scheduled, select the fruit with the smallest r_i.

Proof: Suppose that there is no conflict match M between S and T. Now scan the first timeslot ts_1 in T:

(1) If ts_1 is idle and if there are some fruits whose $l_i \leq ts_1 \leq r_i$, we can match the fruit with the smallest r to ts_1. The operation will not affect the other match relations.

(2) If ts_1 is idle and no fruits whose $l_i \leq ts_1 \leq r_i$, make no change.

(3) If ts_1 is occupied by a fruit a_x but a_y with the smallest r for ts_1, swap the fruit a_x and a_y, assume the timeslot of a_y is ts_y,

We have:

$l_y \leq ts_y \leq r_y \quad l_x \leq ts_1 \leq r_x \quad r_y < r_x$

So, $l_x \leq ts1 < ts_y \leq r_y < r_x$, the fruit a_x can be arrange to timeslot ts_y.

Repeat the swap operations on the remaining time slots, the greedy strategy can generate a no-conflict match between S and T.

Claim 3 and proof

Claim 3. The maximum profit of scheduling the fruits only on the "active timeslots" is the same as that scheduling the fruits on all timeslots.

Proof: by induction

- Base： let $n = 1$, for the fruit (l_1, r_1, v_1), according to the algorithm of p.14, the active timeslot is l_1, can sell the fruit at l_1, obtain the maximum profit v_1.
- Induction：Suppose that $n = m$, for the first m fruits, according to the algorithm of p.14, get m "active timeslots"(let the m "active timeslots" set T), let fruits set S to be the optimal fruit set on T, and the S also the optimal fruit set on all timeslots.

When $n = m + 1$, now consider the fruit a_{m+1} , there are 3 cases:

- The optimal fruit set of the first $m + 1$ fruits doesn't include the fruit a_{m+1}
- Need to delete a fruit in S to free its timeslot for the fruit a_{m+1}
- Add the fruit a_{m+1} to the optimal fruit set S without deleting the previously selected fruits

Claim 3 proof: case 1

- Case 1: The optimal fruit set of the first $m + 1$ fruits doesn't include the fruit a_{m+1}

Since there are $m + 1$ fruits, only one newly added active timeslot and no previous active timeslots will be affected. The optimal solution of the first $m + 1$ fruits is equal to the optimal solution of the first m fruits. So, the optimal solution of the first $m + 1$ fruits can only be related to the first $m + 1$ active timeslots.

Claim 3 proof: case 2

- Need to delete a fruit in S to free its timeslot for the fruit a_{m+1}

the fruit a_{m+1} oy replaces the fruit in the **previous active timeslot**, **the newly added active timeslot does not affect the previous active timeslot**. Therefore, The selected active timeslots of the first $m+1$ fruits are equal to that of the first m fruits. So, the optimal solution of the first $m+1$ fruits is only related to the first $m+1$ active timeslots.

Claim 3 proof: case 3

- Add the fruit a_{m+1} to the optimal fruit set S without deleting the previously selected fruits

1. Know that fruits set S is the optimal fruit set on the m active timeslots set T. According to the greedy strategy of claim 2, get the matching relation R of the fruits in S and the first m active points.
2. Under the current matching relation R, add the $m+1$ **active timeslot**.
3. Try to add the fruit a_{m+1}, according to the greedy strategy of claim 2, try to find a timeslot for the fruit a_{m+1}.

$a \leftarrow a_{m+1}$

Enumerate each timeslot ts_i from the l_{m+1} to ∞{

If ts_i is idle, a can match the ts_i, return ts_i

Else if $ts_i \leftarrow\rightarrow a_k${

if $T_k > T_{m+1}$ {

swap a_{m+1} and a_k

$a \leftarrow a_k$

}

}

}

From above finding algorithm, the returned ts must be the l_{m+1} or the first idle timeslot $l_{m+1} + j$(j depends on the number of conflicts).

The returned ts must be either an active timeslot in T, or the $m+1$ **active timeslot**, So, the optimal solution of the first $m+1$ fruits is only related to the first $m+1$ active timeslots.

Solution: Greedy + Linear Match

According to **Claim 1 and Claim 2**, consider the following algorithm:

Sort fruits by v_i so that $v1 \geq v2 \geq v3 \ldots \geq vn$

$S \leftarrow \emptyset$

$T \leftarrow$ n active timeslots and $ts_1 \leq ts_2 \leq ts_3 \ldots \leq ts_n$

for each fruit a_i {

    if (LinearMatch(a_i,l_i) is true){

        add a_i to S

    }

}

How to Linear Match

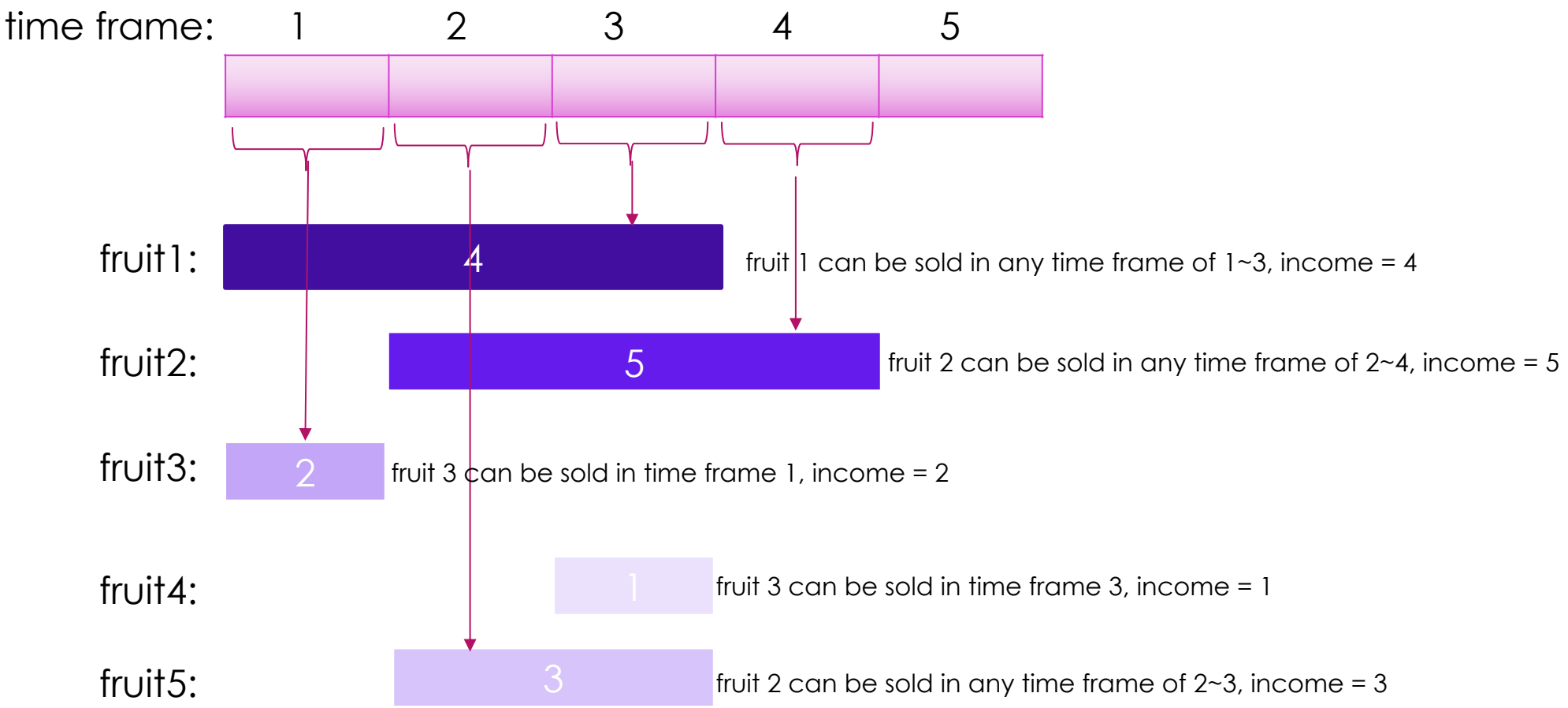
LinearMatch(a_i, x){
    if (x>r_i) return false
    if x is idle{
        x matches a_i
        return true
    }
    a_j = getfruitOf(x)
    if (Ti > Tj){
        return LinearMatch(a_i, next active timeslot of x)
    }else{
        if LinearMatch(a_j, next active timeslot of x){
            x matches a_i
            return true
        }
    }
    return false

}

Demo

Input:

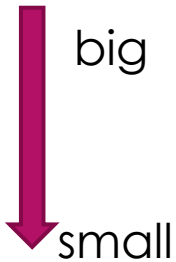
```
5
1 3 4
2 4 5
1 1 2
3 3 1
2 3 3
```



Sort fruits by v_i so that $v1 \geq v2 \geq v3 \ldots \geq vn$

fruit1:	1	3	4
fruit2:	2	4	5
fruit3:	1	1	2
fruit4:	3	3	1
fruit5:	2	3	3

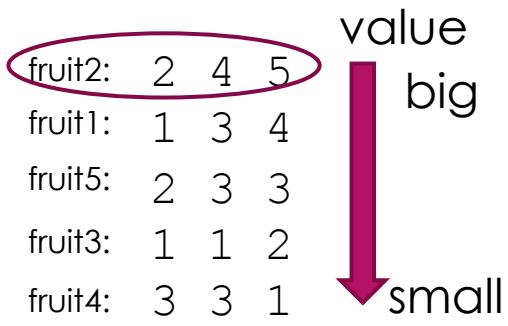
fruit2:	2	4	5
fruit1:	1	3	4
fruit5:	2	3	3
fruit3:	1	1	2
fruit4:	3	3	1



Algorithm of Finding all Active Timeslots

S ← ø
Sort the fruit by start time so that $l_1 \leq l_2 \leq l_3 \ldots \leq l_n$
x ← 0
For i=1 to n {
    x ← max(x+1, l_i);
    add x to S
}

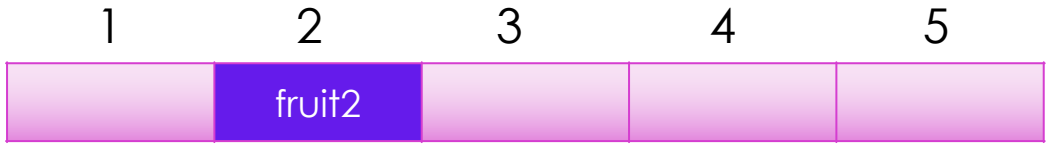
Active Timeslots

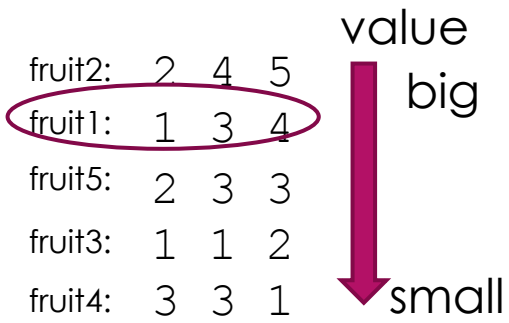


fruit2, I =2, try ts2, 2 is idle

Result:
fruit2(2,4,5)←→ ts2

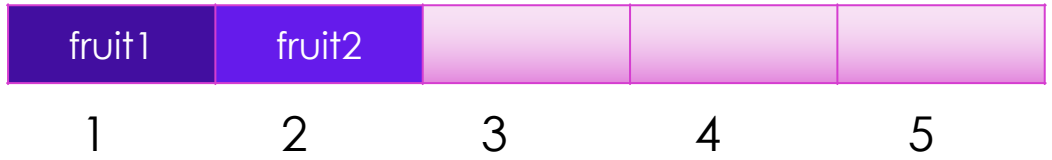
Active Timeslots

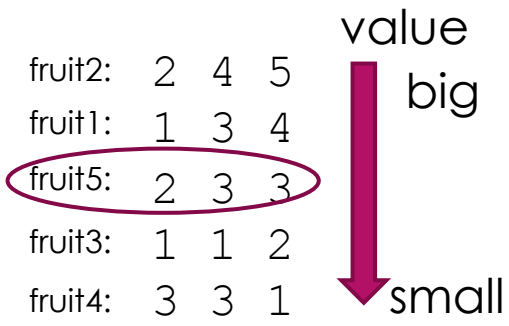




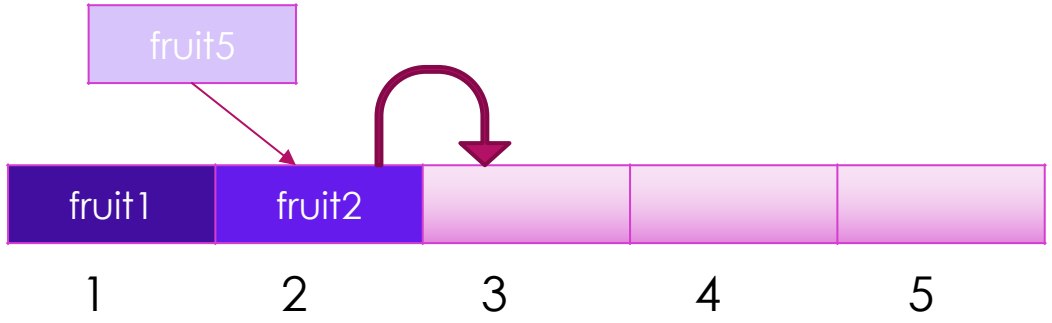
fruit1, l =1, try ts1, ts1 is idle

Result:
fruit1(1,3,4)←→ ts1

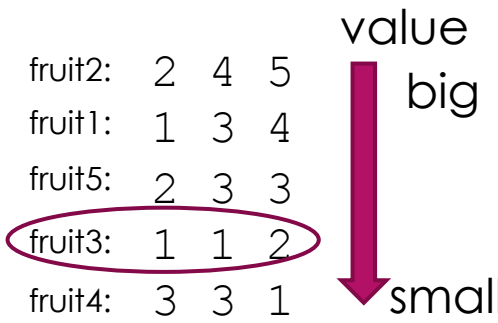




fruit5.l = 2, try ts2
but ts2 matches fruit2, conflict.
fruit2.r> fruit5.r, let fruit2 try the next: ts3,
ts3 is idle, fruit2 moves to ts3, fruit 5 matches ts2



Result:
fruit2(2,4,5)←→ ts3
fruit5(2,3,3)←→ ts2
fruit1(1,3,4)←→ ts1



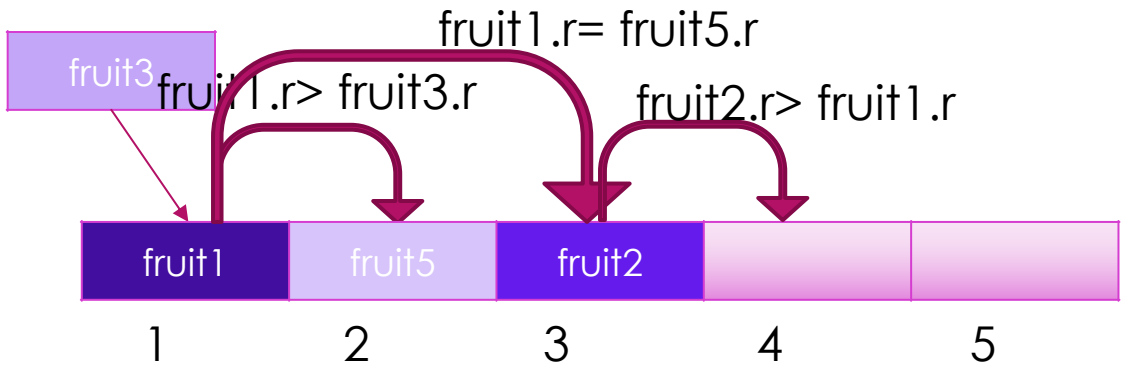
fruit3.l =1, try active timeslot 1
but ts1 matches fruit1, conflict.
fruit1.r> fruit3.r, let fruit1 try the next: ts2,

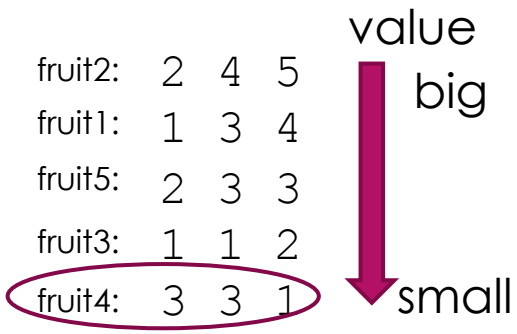
ts2 matches fruit5, conflict, fruit5.r == fruit1.r,
let fruit1 try the next: ts3

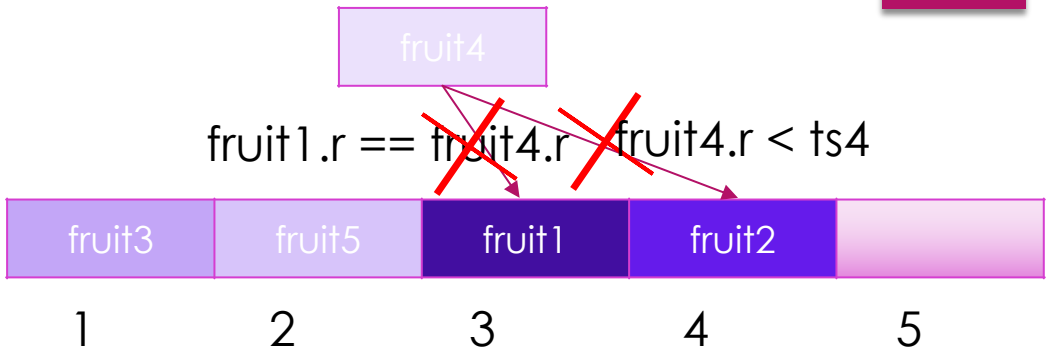
ts3 matches fruit2, conflict, fruit2.r > fruit1.r,
let fruit2 try the next： ts4

ts4 is idle, ts4 matches fruit2.r return true
ts3 assigned to fruit1 return true
ts1 assigned to fruit3 return true

Result:
fruit3(1,1,2)←→ ts1
fruit5(2,3,3)←→ ts2
fruit1(1,3,4)←→ ts3
fruit2(2,4,5)←→ ts4







fruit4, l =3, try ts3, but ts3 matches fruit1, conflict.
fruit1.r == fruit4.r, let fruit4 try next: ts4
fruit4.r < ts4, fail, return false