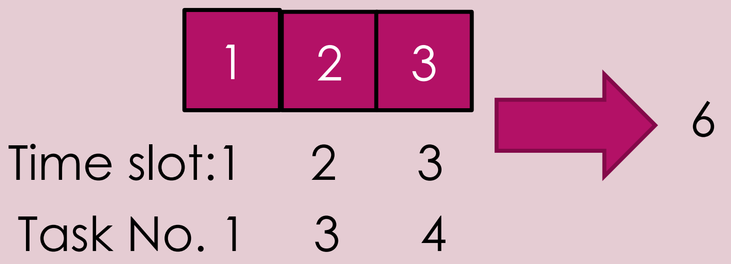
LAB8 Q2

YAO ZHAO

The Maximum Income

- Hong has N tasks for you. **Each task can be finished in a unit time, and each unit time can only do one task**. For the i-th task, you can finish it in a unit time in [Si,Ti], and you will be paid Vi.

  Hong wants to know the maximum income you can get.
- $N \le 5000$，$1 \le Si \le Ti \le 10^8$，$1 \le Vi \le 10^8$。

Input	Output	
4 1 1 2 2 2 2 1 2 3 1 3 1	6	

No *Si*

- Hong has *N* tasks for you. **Each task can be finished in a unit time, and each unit time can only do one task**. For the i-th task, you can finish it in a unit time before *Ti*, and you will be paid *Vi*.

  Hong wants to know the maximum income you can get.

```
Sort the tasks by Vi so that V1>V2>V3…>Vn
Initialize all time slot to 0
For i = 1 to n{
    For j = Ti  to 1{
        if (time slot j is 0){
            put task i to time slot j
            break
        }
    }
    For j = n to Ti+1{
      if (time slot j is 0){
            put task i to time slot j
            break
        }
    }
}
```

- What happens when add a constraint condition about starting time?
- Preliminary judgment: When to use greedy? How to use greedy?

- Sort the tasks by V_i so that $V_1 > V_2 > V_3 \ldots > V_n$
- Consider each task in order
- Assume we picked first i tasks to set **S** and ensures that each task is matched to a time slot.
- If add the (i+1)th task to **S** , every task in the **S** can still allocate a time slot, then the (i+1)th task can be chosen, otherwise the (i+1)th task should be abandoned.

- Difficulty: which time slot does the most valuable task be put in?
- $[S_i, T_i]$, $1 \le Si \le Ti \le 10^8$
- The range is huge

Algorithm 1：Backtrack

- Sort the tasks by Vi so that V1>V2>V3…>Vn
- From i to n, for each task, enumerate every free slot from Si to Ti . Let Max{Ti}-min{Si}+1=L, there are at most L choices for each task, at least one choice, after enumerating all possible choices, record the maximum income.
- Time complexity $O(L^{N-1})$。
- L upper bound： 10^8
- **Intolerable inefficiency**

- Observe: $N \le 5000$， $1 \le S_i \le T_i \le 10^8$
- A large number of points within the range are useless
- How to identify and remove these useless points

Define Active Points

- Initially, all points are marked by black. Then, for each task i(Si, Ti, Vi), find the smallest k that satisfies k≥Si and the point k is black, then mark the moment k by white. Finally, you end up with exactly N points that are marked white, which are useful points. Call these points " Active Points".

```
Sort the tasks by Si so that S1<S2<S3…<Sn
Active Points Set  S ← ø
X←0
For i=1 to N {
    x=max(x+1, Si)
    add x to S
}
```

- We discover the following：

For all the tasks in a task set S, if all tasks in S can be finished in Active Points set T, it must exist such a greedy algorithm: scan every Active Point in T in the order from small to large, for each Active Point , if there are some tasks can be done in the moment, choose the task with the smallest Ti value.

Proof

- Proof: Suppose that the active points and the tasks have corresponding matches:

(a1, b1) (a2, b2) (a3, b3)…(an, bn)   while $a1<a2<a3<..<an$ （1）

- Assume the first i-1 pairs are the same with greedy algorithm.
- From i-th active point ai , matches task bj according the greedy algorithm.

We can inference : $S(bj)<=ai<=T(bj)$, and $T(bj) < T(bi)$, From(1) we know $i<j$, $ai < aj$

- For active point aj : $S(bi)<ai < aj< T(bj) < T(bi)$ so aj can be used to finish bi
- Therefore, this greedy method must be able to construct matching relations between the tasks and the activity points.

Lab 8 Q2 Solution

Sort the tasks by Vi so that V1>V2>V3…>Vn

Initialize all time slot to 0

Generate Active Points // see Page.9

```
For i = 1 to n{
    x←Si
    find(i, x)
}
```

Check if fine an active point to match task i

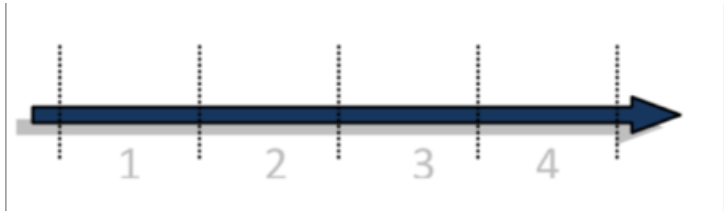
```
Find(i, x){  //x← Si
    if (x > Ti){
        return false
    }
    if (time slot x have no task){
        put the task i into time slot x
        return true
    }
    j ← the task no in the time slot x
    if (Ti > Tj){   //see page 10
        return find(i, x+1)
    }
    else{
        if find(j,x+1){
            put the task i into time slot x
            return true
        }
        else return false
    }
}
```

sample

input	output
4 1 1 3 1 2 2 1 3 1 1 2 4	8

Sort by V:
1 2 4
1 1 3
1 2 2
1 3 1

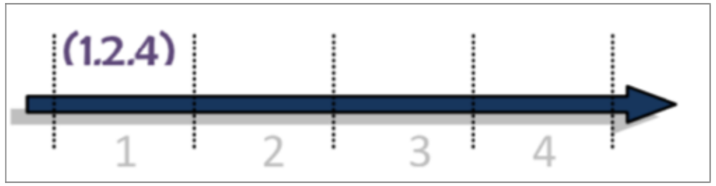
Activity points
1
2
3
4



Task1

Task 1 will be put on Activity Point 1

Task(1,2,4)←→ 1

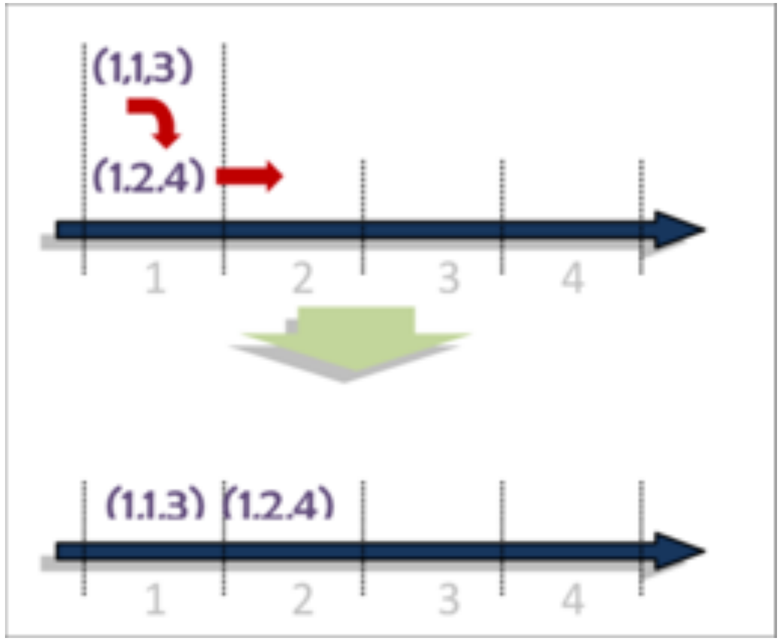


Task 2

Task 2 find activity points from 1

But 1 has task(1,2 4), T2 > T1, so Task(1,2,4) try to find a new available point. Fortunately, Task(1,2,4) find activity point 2

Now:
Task(1,2,4)←→ 2
Task(1,1,3)←→ 1



Task 3

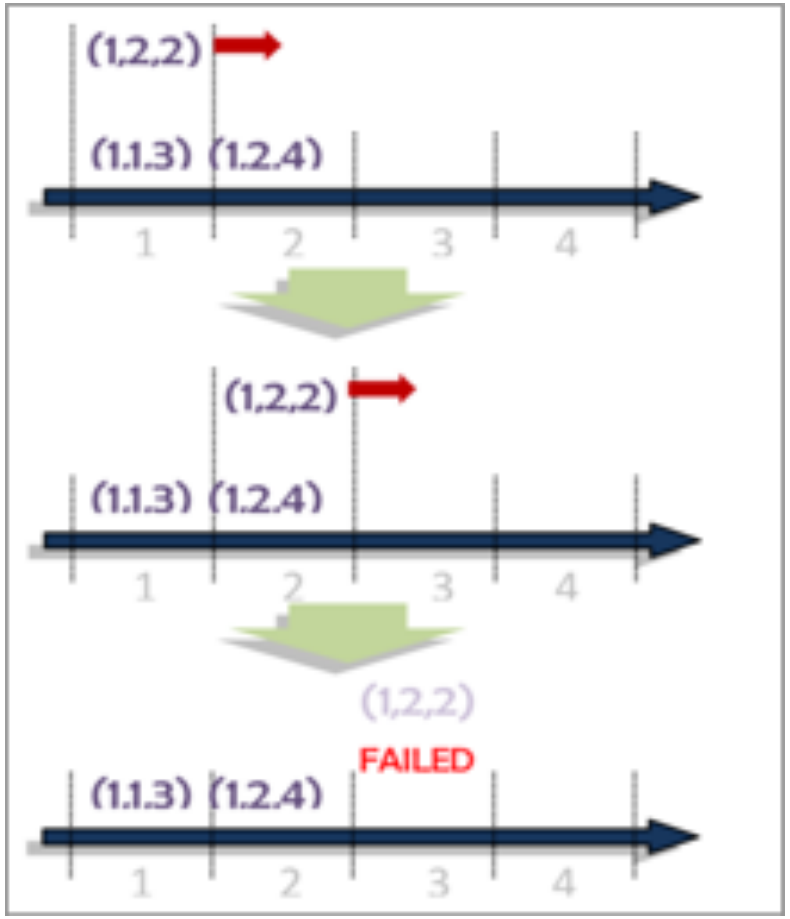
Task3 find activity points from 1

But 1 has task(1,1,3), T2 < T3, so Task3(1,2,2) try to find a new available point.

It come to activity point 2, 2 has task (1,2,4), T1 <= T3, so Task3(1,2,2) go on finding.

It come to activity point 3, but 3 > T3, so task 3 failed.

No change:
Task(1,2,4)←→ 2
Task(1,1,3)←→ 1



Task 4

Task4(1,3,1)，find activity points from 1.
But 1 has task2(1,1,3), T2 < T4, so Task4(1,3,1) try to find a new available point.

2 has task1(1,2,4)， T1 < T4, Task4(1,3,1) go on finding.

3 is free, so put the task4 to current active point.

Finally:
Task(1,2,4)←→ 2
Task(1,1,3)←→ 1
Task(1,3,1)←→ 3