Greedy Algorithms
Chapter 16

Mei-Chen Yeh

4/13 START

Activity-Selection Problem

- Given the time table of activities, select a ***maximum-size*** subset of ***mutually compatible*** activities.

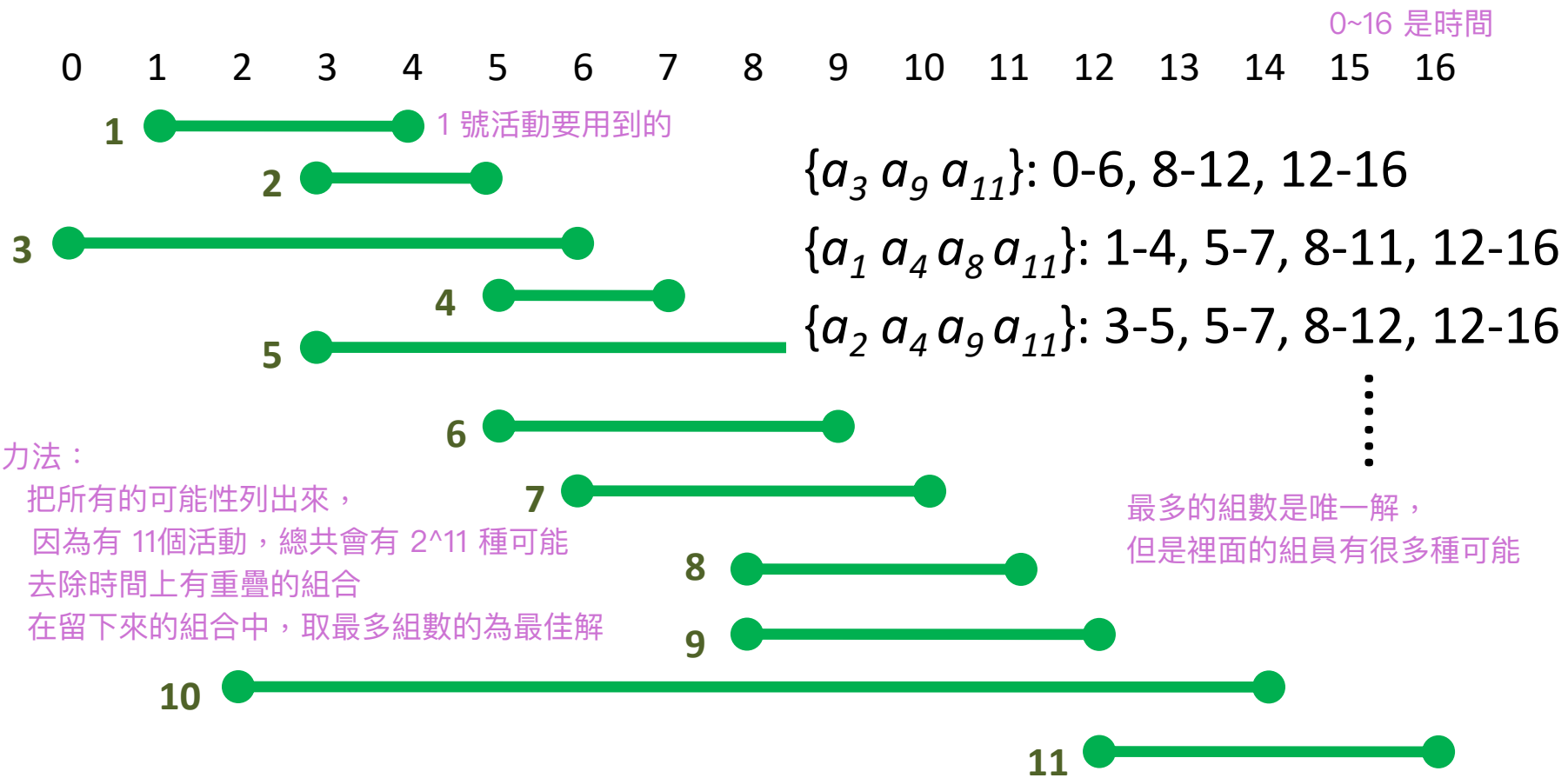
假設要辦活動，要租借場地
S：什麼時候開始
F：什麼時候結束

不可以活動還沒結束下一個活動
就進來 –> 活動不會互相重疊

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

第 11個活動是從 12點到 16點

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



$\{a_3\ a_9\ a_{11}\}$: 0-6, 8-12, 12-16

$\{a_1\ a_4\ a_8\ a_{11}\}$: 1-4, 5-7, 8-11, 12-16

$\{a_2\ a_4\ a_9\ a_{11}\}$: 3-5, 5-7, 8-12, 12-16

最多的組數是唯一解，
但是裡面的組員有很多種可能

暴力法：

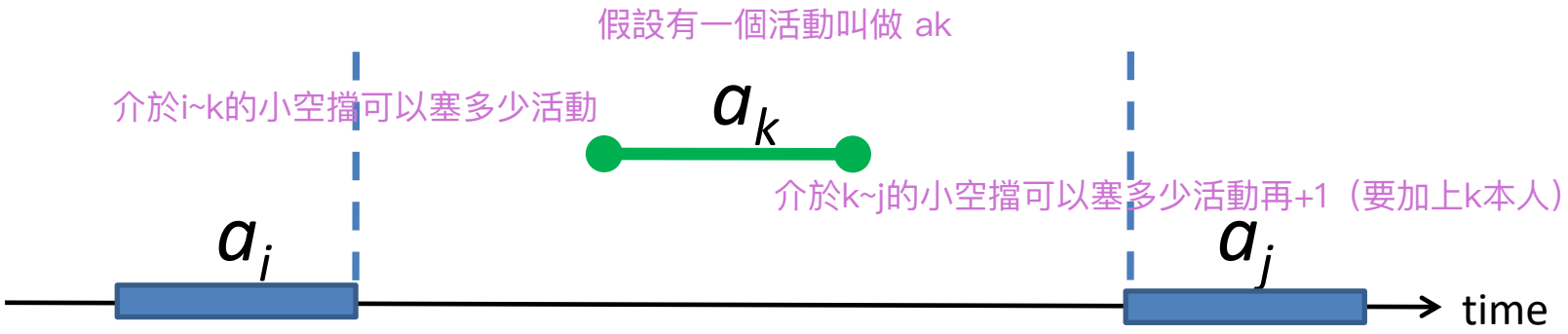
1. 把所有的可能性列出來，
   因為有 11個活動，總共會有 2^11 種可能
2. 去除時間上有重疊的組合
3. 在留下來的組合中，取最多組數的為最佳解

Dynamic Programming Approach

- The optimal substructure?

Sij ：只考慮發生在ai~aj 這個時間段中的活動

  - S_{ij}: The activity set that start after a_i finishes and that finish before a_j starts



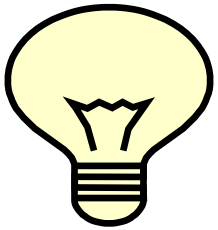
  - $c[i, j]$: ***An optimal*** solution for S_{ij} 存的答案是最多可以塞幾個活動
  - *Recurrence?* $c[i, j] = c[i, k] + c[k, j] + 1$

Dynamic Programming Approach

$$c[i,j] = \begin{cases} 0 & \text{if } S_{ij} = \emptyset \\ \max\limits_{a_k \in S_{ij}} \{c[i,k] + c[k,j] + 1\} & \text{if } S_{ij} \neq \emptyset \end{cases}$$

The algorithm?

The time complexity?



Greedy Method

- Pick the one that looks best at the moment

短視近利

貪婪法會當下就選一個對自己最有利的，
DP法是所有的小問題都做完後才選最有利的

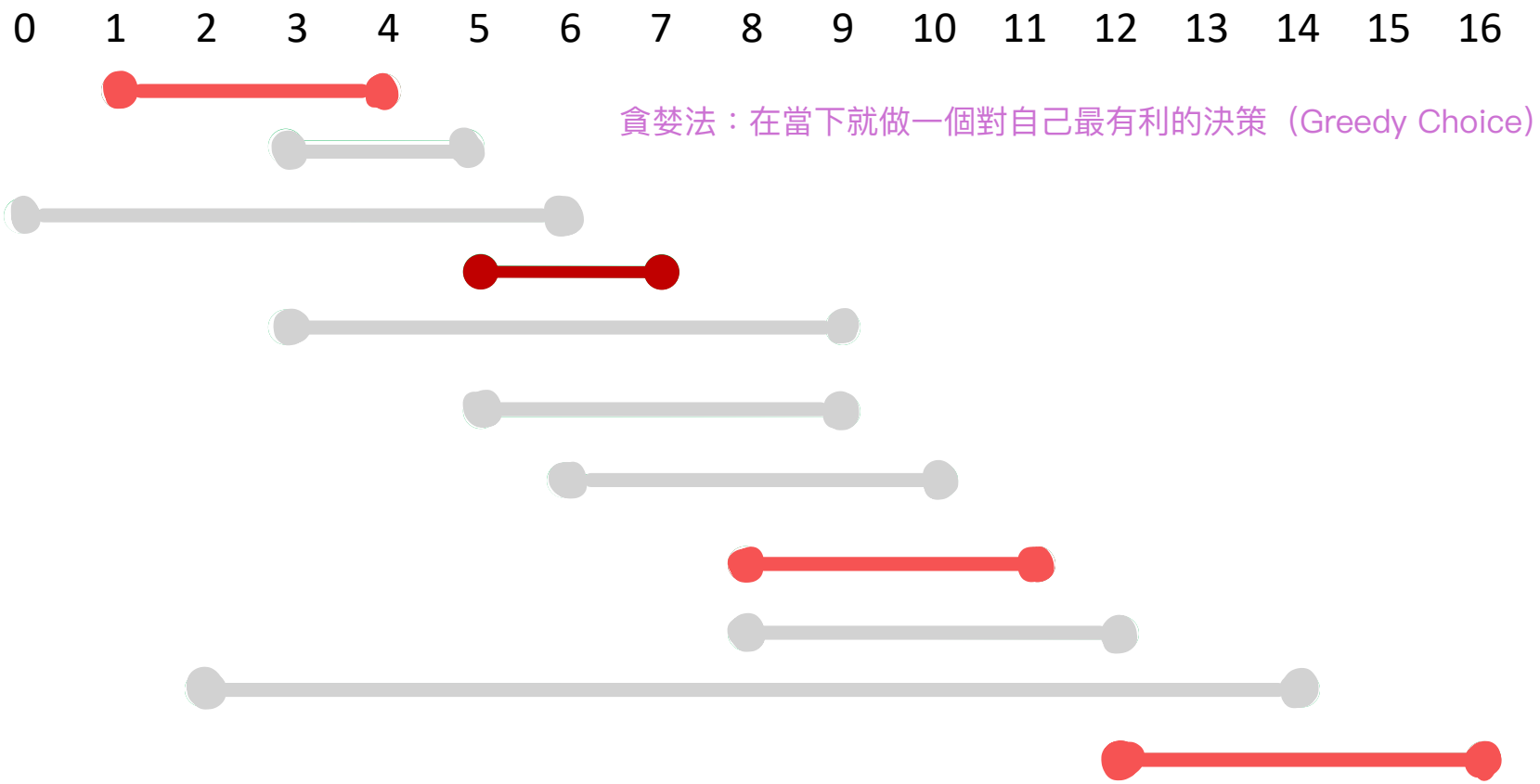
- The activity-selection problem
  - Choose the activity that has the ***earliest finish time***!

挑最早結束的–>剩下的可用時間最多

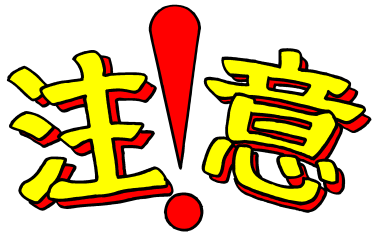
- Once we make the greedy choice, only one sub-problem remains.

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

已經根據活動結束時間（fi）從小到大排序了。選擇最早結束的活動，並保證他們不重疊



貪婪法：在當下就做一個對自己最有利的決策（Greedy Choice）



- The greedy algorithm does ***not*** always give the optimal solution! 貪婪法並不保證會給你最佳解！！
- However, in the activity-selection problem, the greedy method leads to an optimal solution. 但有些特定問題搭配正確的決策（Greedy Choice）是可以給你最佳解的
  例如：activity–selection problem

- Theorem

  Consider any nonempty sub-problem S, and let a_m be an activity in S with ***the earliest finish time***. Then a_m is included in some maximum-size subset of mutually compatible activities of S.

am 這個活動肯定存在於某個最佳解裡面

am 是在所有活動裡面結束時間最早的

a_m : An activity in S with *the earliest finish time*
a_m must be included in some optimal solution!

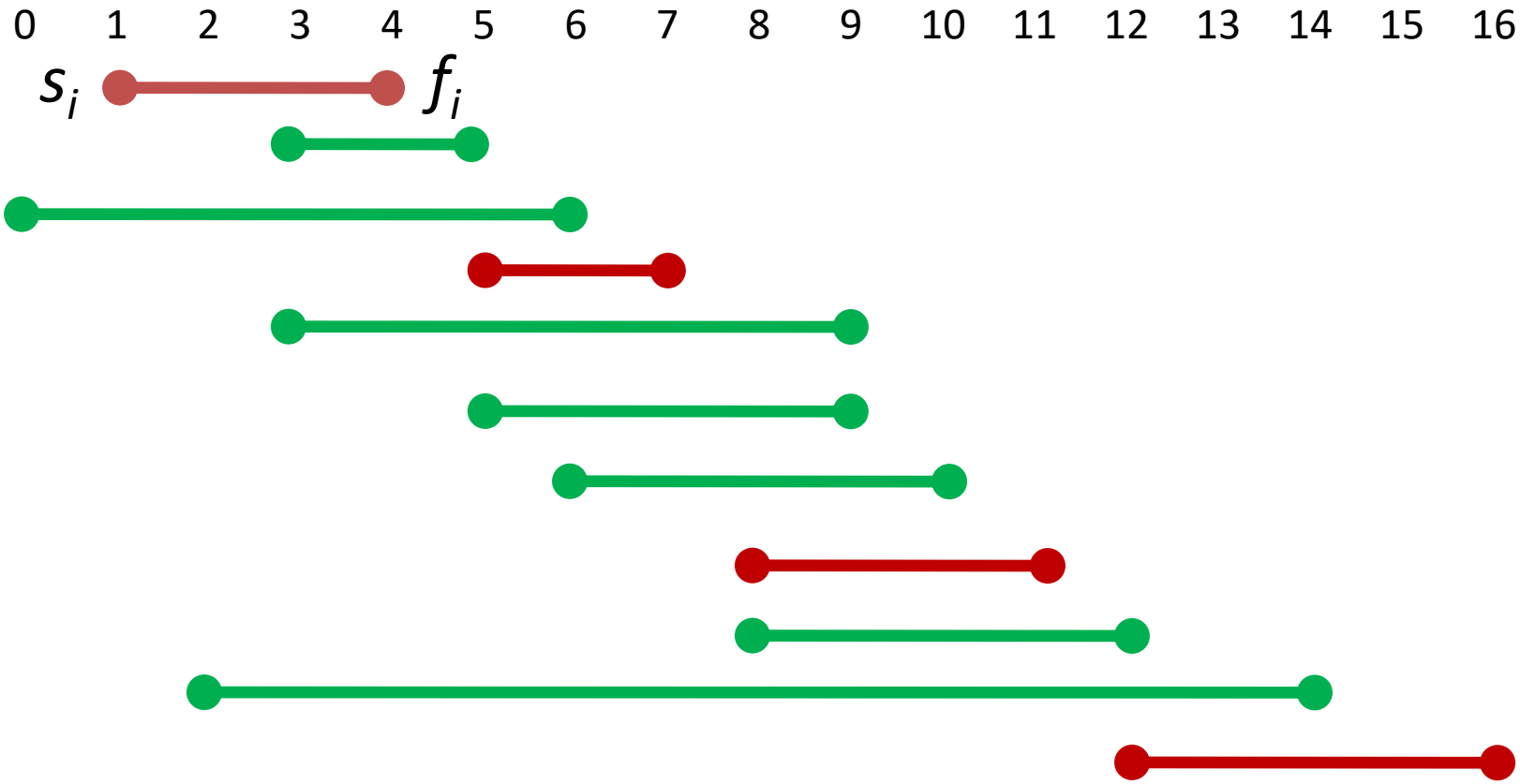
am 這個活動肯定存在於某個最佳解裡面

- Proof
  - Let A be an optimal solution given S. 假設 A組合是最佳解
  - Let a_j be the activity in A with the earliest finish time.
  - If $a_j = a_m$, we are done!

令 aj為 A組合中結束時間最早的活動

  - If $a_j \neq a_m$,
    - let $A' = A - \{a_j\} \cup \{a_m\}$
    - A': also an optimal solution?
      - Yes! Because activities in A' are disjoint and $|A| = |A'|$
    - We conclude that A' is a maximum-size subset of mutually compatible activities of S, and it include a_m.

Algorithm



Assume $f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_{n-1} \leq f_n$

Assume that the n activities are ordered by non-decreasing finish time

An iterative algorithm $\Theta(n)$

兩個 array，s 存開始時間，f 存結束時間

GREEDY-ACTIVITY-SELECTOR(s, f)

1. $n = s.length$
2. $A = \{a_1\}$
3. $k = 1$ // k keeps the most recent addition to A
4. **for** $m = 2$ **to** n
5. **if** $s[m] \geq f[k]$
6. $A = A \cup \{a_m\}$
7. $k = m$
8. **return** A

k 紀錄的是最後一個被放進 A 組合裡的活動 是第幾號活動

從 2號活動跑到最後一個活動

如果 m號活動的開始時間，比k號活動的結束時間晚的話，就選他，把它當成新的 k，繼續挑下去

最後回傳 A 就會是最佳的活動組合

Greedy vs. DP

- Both explore the optimal substructure of a problem
  - An optimal solution to the problem contains within it optimal solutions to sub-problems
- DP: a bottom-up manner
  - solve sub-problems, then make a choice
- Greedy: a top-down manner
  - make a greedy choice, then solve sub-problems that remain

Practice

- Suppose that instead of always selecting the first activity to finish, we make the following greedy choices: 三個裡面只有一個會成功：最晚開始的那個方法
  - Select the last activity to start that is compatible with all previously selected activities.
  - Select the activity of least duration from among those that are compatible with previously selected activities.
  - Select the compatible activity that overlaps the fewest other remaining activities.

wrong way

wrong way

Does each of these greedy approaches yield an optimal solution?