GET
COFFEE
AND
WRITE
CODE

Greedy Method

Scheduling

Simple Scheduling Problem

N명의 손님이 동시에 계산대에 도착했다. 계산대는 하나이다. 각 손님마다 계산에 소요되는 시간이 서로 다르다. 각 손님들이 기다린 시간의 **종합**을 최소화하는 순서는?

(1) Ladies First ← the worst choice, in my opinion

(2) Me First ← the best for me, but…

(3) Shortest Processing Time First

(4) Longest Processing Time First ← the same as (1)

(5) Dynamic Programming

Why SPTF is the optimal ?

- SPTF가 아닌 최적해가 있다고 가정해보자.

optimal schedule

	T_i	T_{i+1}	

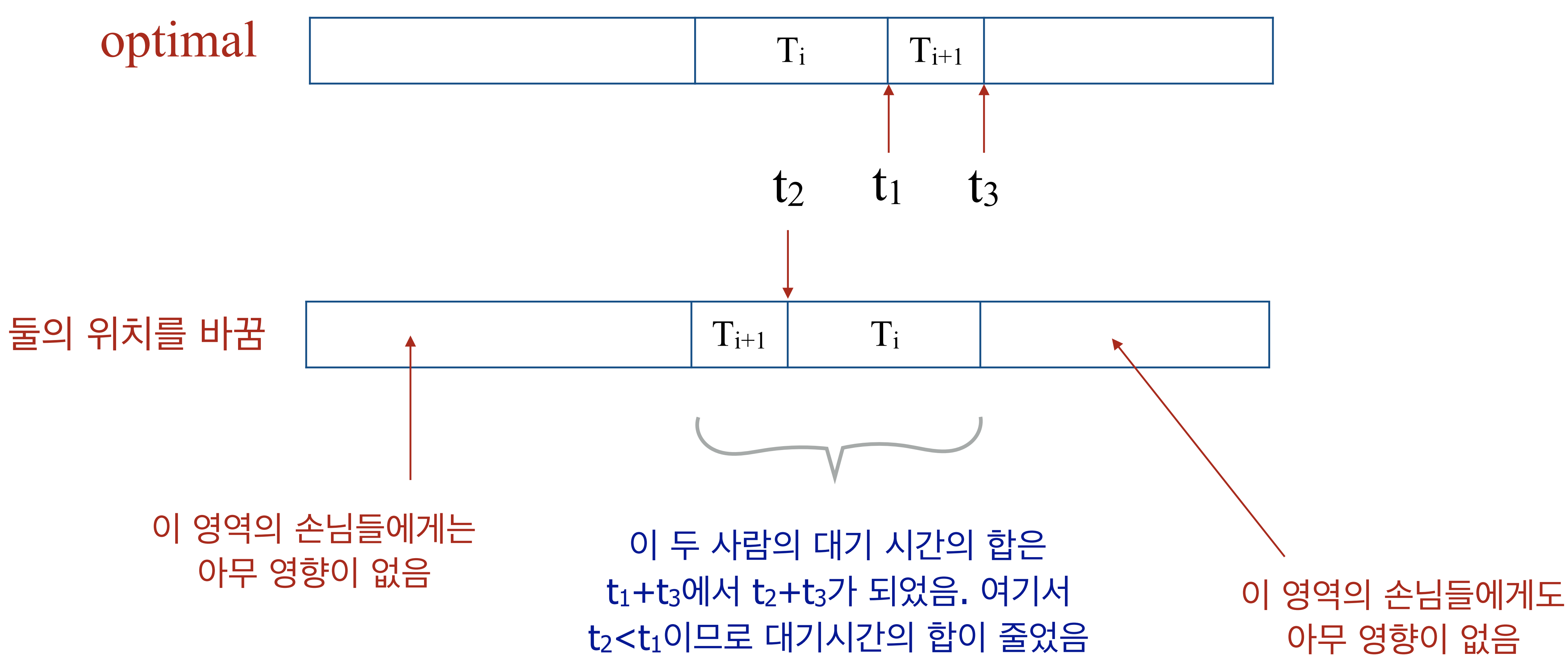
처음으로 $T_i > T_{i+1}$인 손님

T_i와 T_{i+1}을 각각 두 손님의 처리시간이라고 하자.

이 둘의 위치를 바꾸면 어떨까?

Why SPTF is the optimal ?

- SPTF가 아닌 최적해가 있다고 가정해보자.



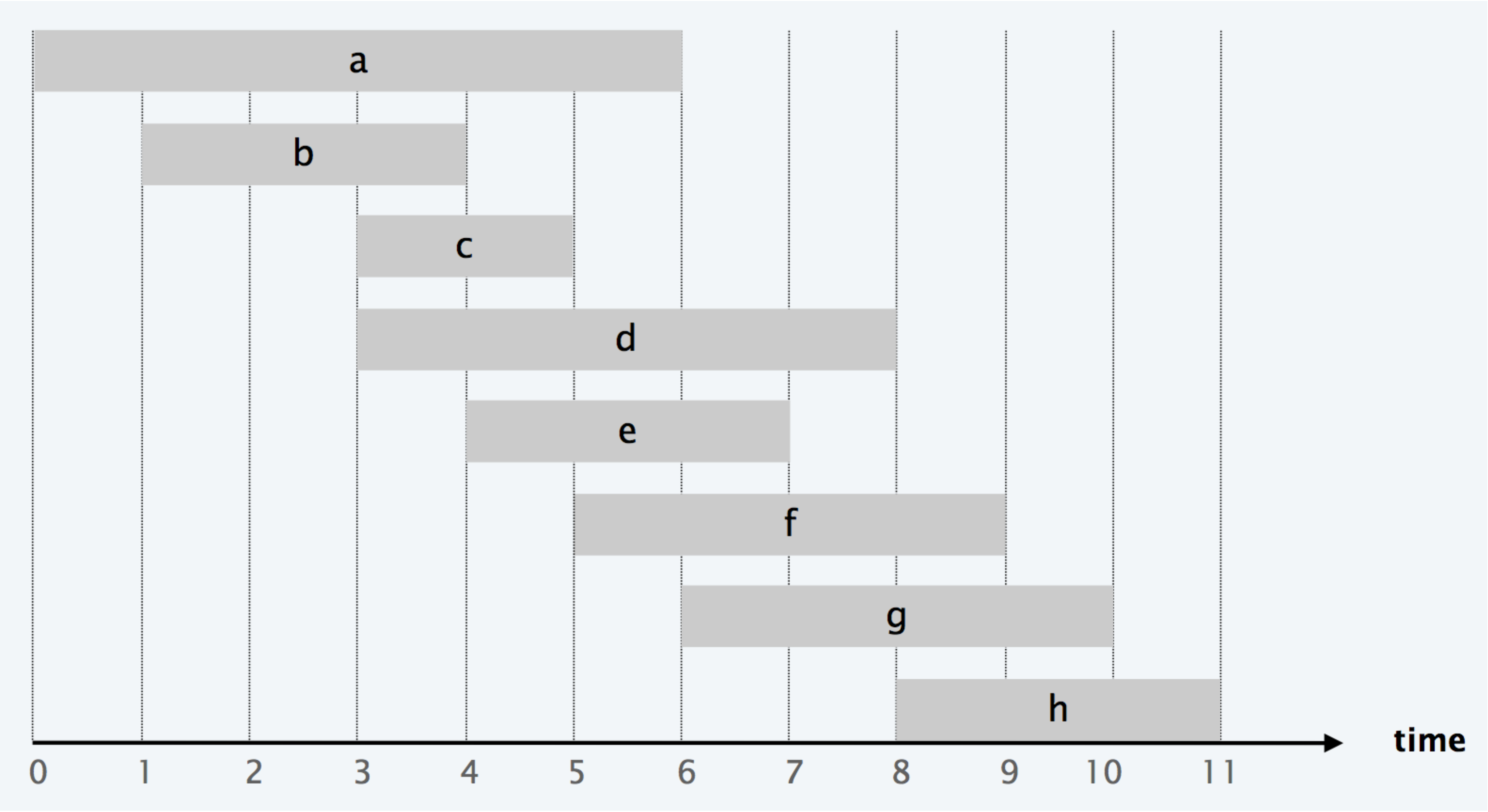
이런 쌍이 존재할 때 마다 자리를 바꾼다면
결국 SPTF와 동일해짐

Interval Scheduling

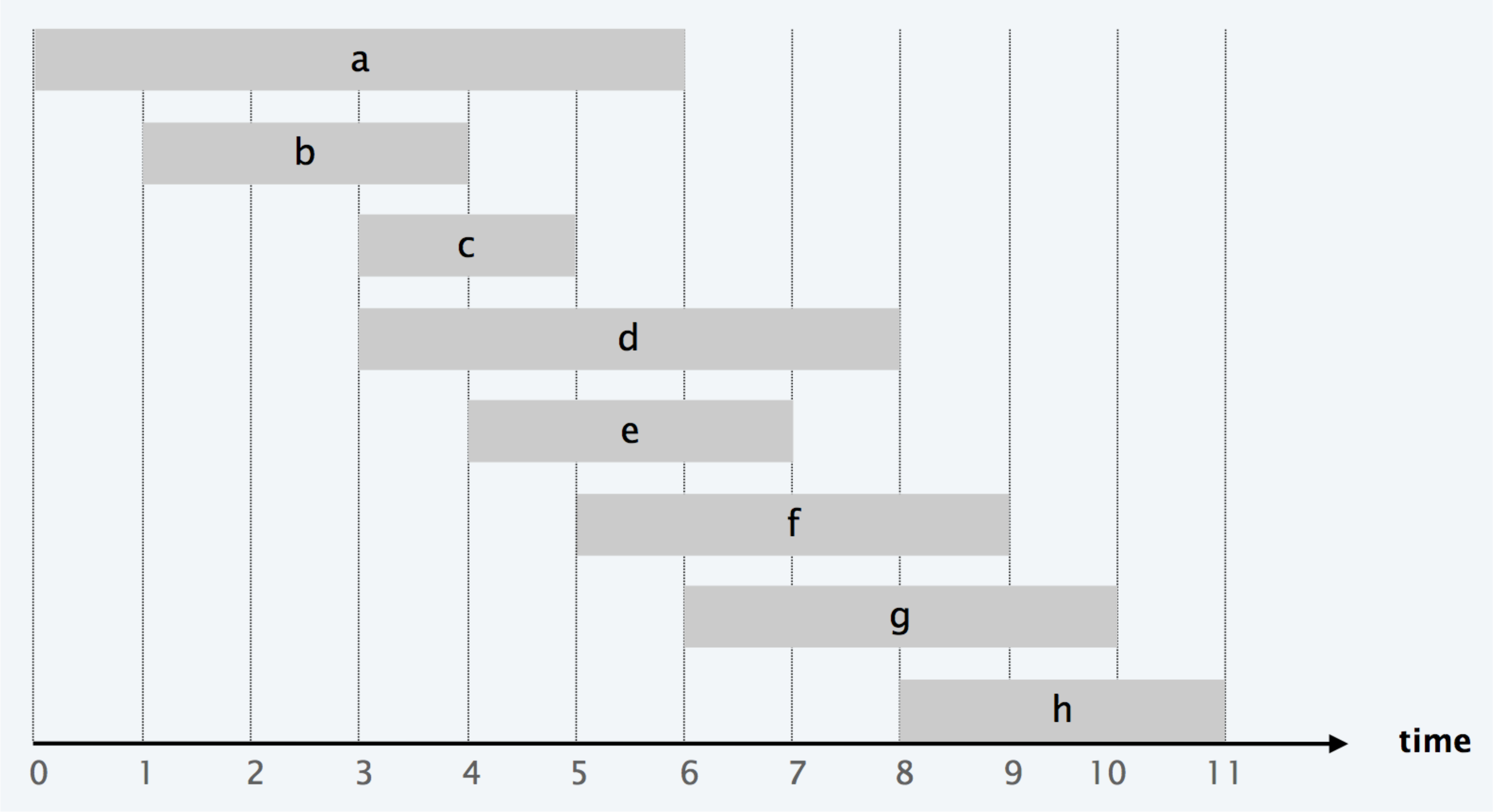
Weighted Interval Scheduling

- N개의 작업이 주어짐. 각각의 작업(job)은 시작시각과 종료 시각, 그리고 가중치를 가짐
- 즉 작업 j는 (s_j, f_j, w_j)로 표현됨, 여기서 s_j는 시작시각, f_j는 종료시각, 그리고 w_j는 가중치
- 시간적으로 겹치지 않는 두 작업은 서로 compatible하다고 말함
- 서로 compatible하면서 가중치의 합이 최대가 되는 부분집합을 찾아라.

Weighted Interval Scheduling

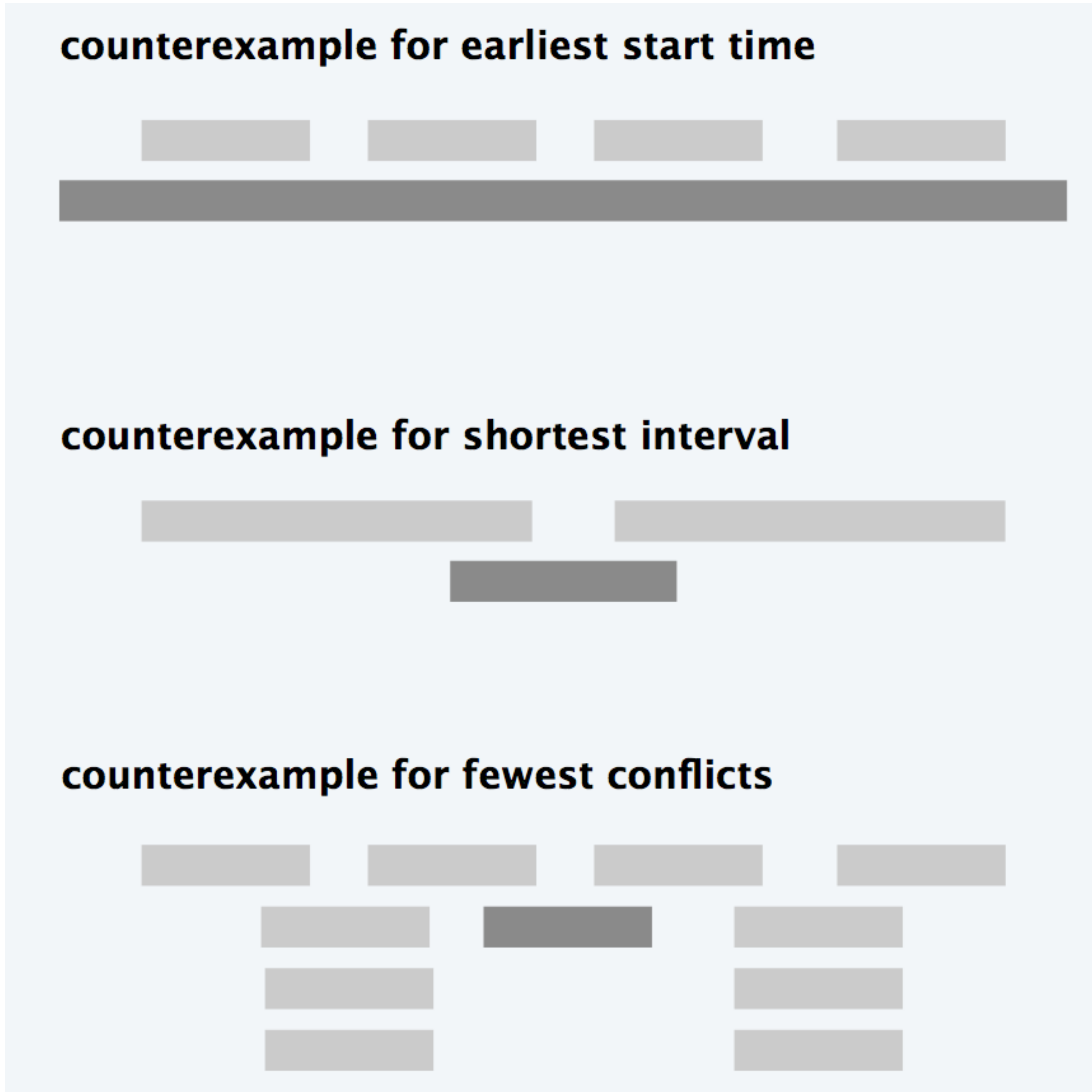


모든 작업의 가중치가 1이라면?



서로 compatible한 최대개수의 부분집합을 찾는 문제

모든 작업의 가중치가 1이라면?

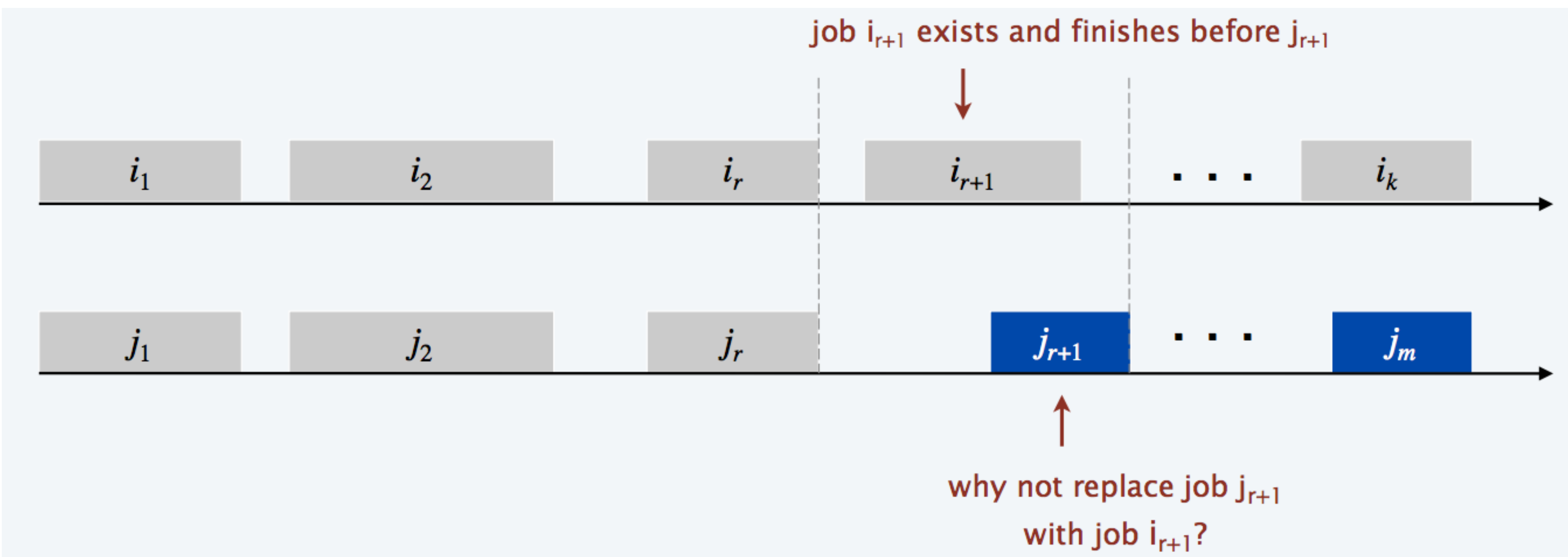


Earliest-Finish-Time First (EFTF)

- Finish Time이 빠른 것 부터 순서대로 고려한다.
- 이미 선택한 작업들과 compatible하면 선택한다.

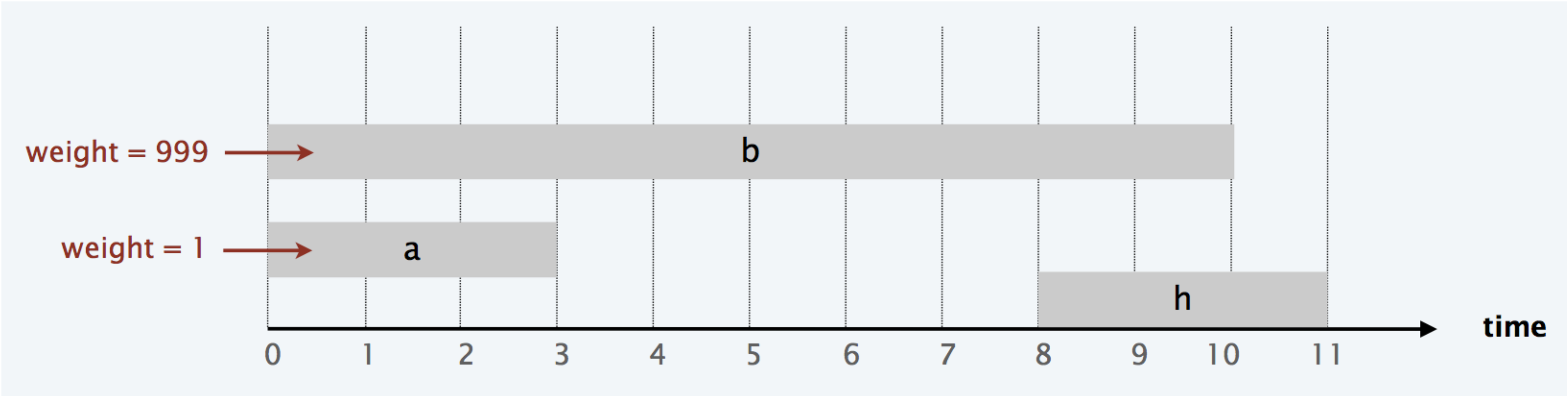
EFTF의 최적성 증명

- 최적이 아니라고 가정하자.
- $i_1, i_2, \ldots, i_k$를 EFTF 알고리즘이 선택한 작업이라고 하고, $j_1, j_2, \ldots, j_m$을 최적해라고 하자.
- $i_1=j_1, i_2=j_2, \ldots, i_r=j_r$이고 $i_{r+1} \neq j_{r+1}$인 인덱스를 r이라고 하자.



가중치가 있는 경우

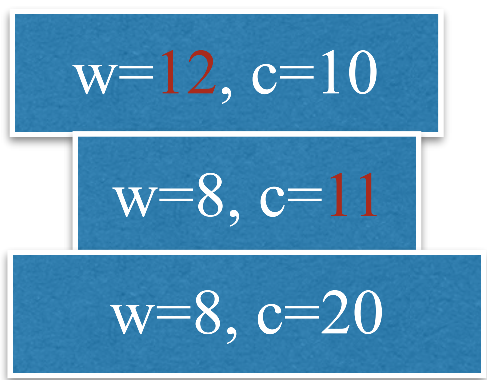
- 가중치가 있는 경우에는 성립하지 않음



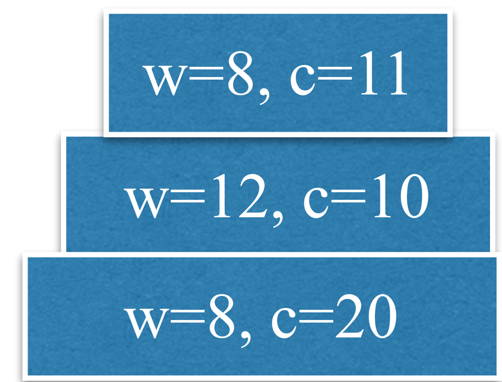
상자 쌓기

Greedy Approach

- N개의 상자가 있다. 각 상자 i는 무게 w_i와 허용하중 c_i를 가진다.
- 모든 상자를 일렬로 쌓을 수 있는가?
- Largest Capacity First ← first means bottom

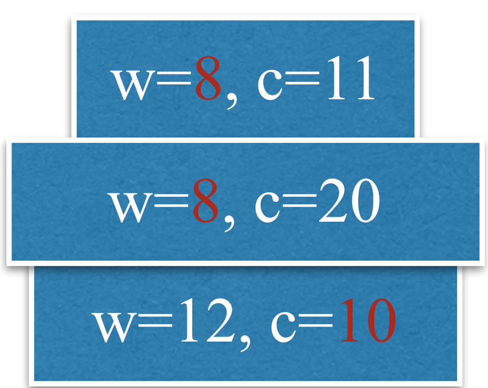


No



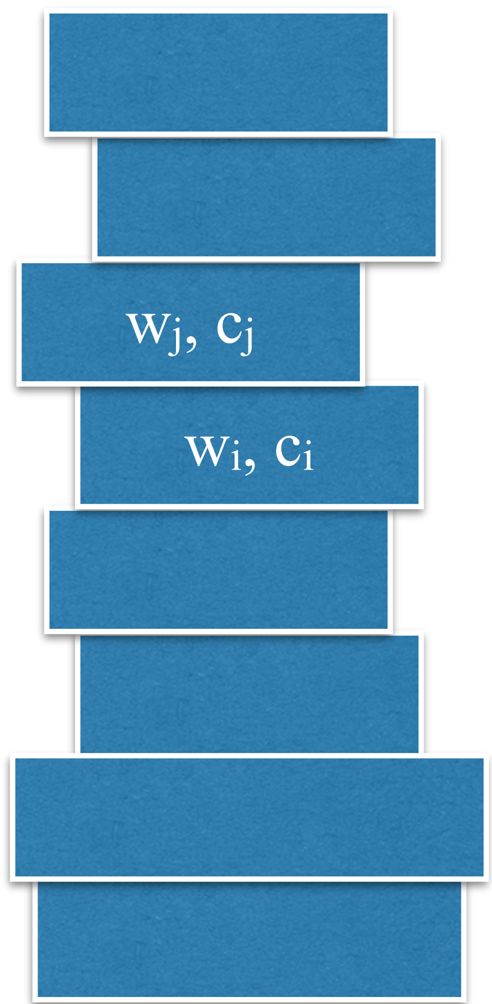
but, yes

- Largest Weight First



No

Largest [Capacity + Weight] First

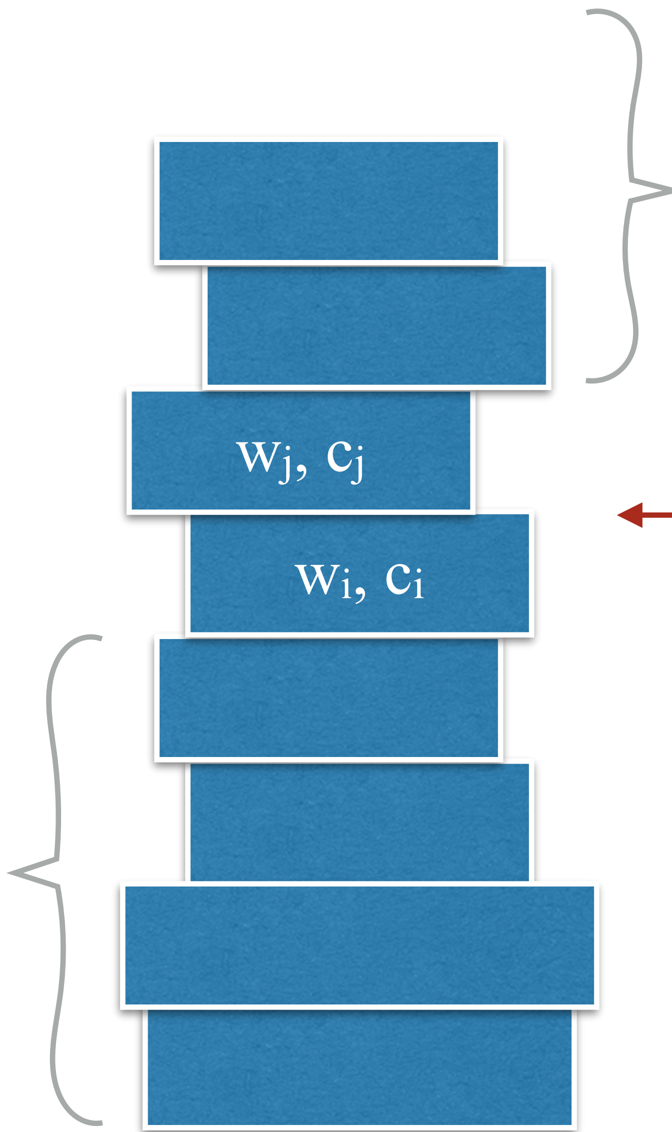


first pair (from bottom)
with $w_j+c_j>w_i+c_i$

두 박스의 자리를 바꾸면 ?

Not LCWF
but feasible

Largest [Capacity + Weight] First



이 위쪽 박스의 무게의 총합을 W_{upper}라고 하자.

$c_j \geqq W_{upper}$

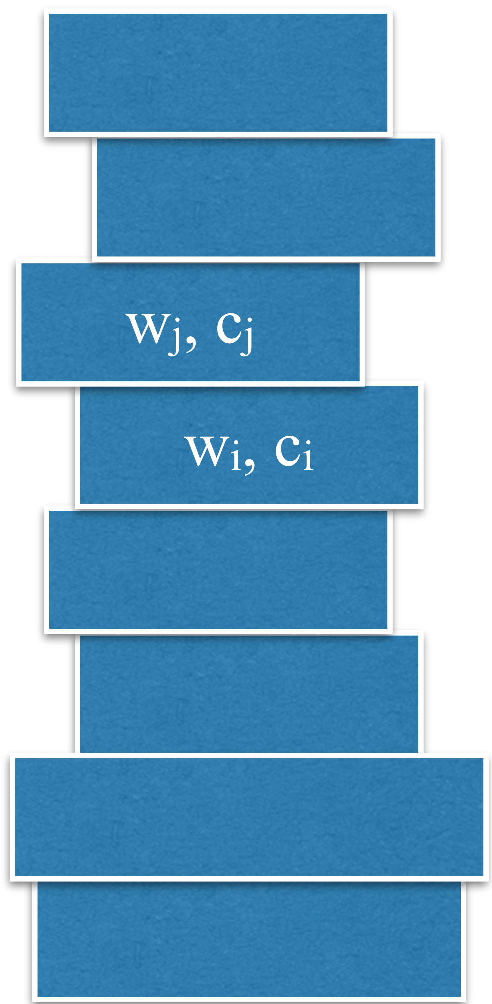
$c_i \geqq W_{upper} + w_j$

아무 상관이 없음

Not LCWF
but feasible

- 박스 (w_i,c_i)는 현재 위치보다 위로 올라가므로 아무 문제 없다.
- $w_j+c_j > w_i + c_i \geqq w_i + W_{upper} + w_j$ 이므로 $c_j > w_i + W_{upper}$ 이다. 따라서 박스 (w_j,c_j)가 한 칸 내려와도 아무 문제없다.

Largest [Capacity + Weight] First



first pair (from bottom)
with $w_j+c_j>w_i+c_i$

Not LCWF
but feasible

이런 모든 쌍을 찾아서 자리를 바꾼다면 결국 LCWF가 된다.

즉, 상자쌓기가 가능하다면 LCWF로 쌓을 수 있다.

- 어떤 기준을 정하고 ← 상자쌓기의 경우 "무게+허용하중", 구간 스케쥴링의 경우 finish time 등
- 그 기준에 최적인 결정을 내려 나가며, ← incremental하게 해를 구성해 나감
- 한 번 내린 결정은 번복하지 않는다.

Greedy Method Examples

- Prim's algorithm for MST
- Kruskal's algorithm for MST
- Dijkstra's algorithm for shortest path problem
- Huffman coding algorithm
- …
- Very common in heuristics (or approximation algorithms)