

파이썬으로 배우는 알고리즘 기초

Chap 4. 탐욕 알고리즘



4.3

마감시간 있는
스케줄 짜기





4.3 마감시간 있는 스케줄 짜기



- 마감시간 있는 스케줄 짜기 문제
 - 마감시간과 함께 여러 개의 작업이 주어지고,
 - 각 작업을 마감시간 전에 마치면 받을 수 있는 보상이 정해져 있다.
 - 주어진 마감시간내에 얻을 수 있는 보상을 최대화하는 스케줄을 작성하라.
- 각 작업을 끝내는데 1의 단위 시간이 걸린다고 가정한다.
- 보상은 마감시간 이전이나 마감시간에 끝내는 경우에만 주어진다.
- 마감시간 이후의 스케줄에 대해서는 고려할 필요가 없다.



4.3 마감시간 있는 스케줄 짜기

Job	Deadline	Profit
1	2	30
2	1	35
3	2	25
4	1	40

1	2	3	4
---	---	---	---

Job 1 **Job 2** (impossible)

Job 1 Job 3

Job 1 **Job 4** (impossible)

Job 2 Job 1

Job 2 Job 3

Job 2 **Job 4** (impossible)

Schedule	Total Profit
1, 3	55
2, 1	65
2, 3	60
3, 1	55
4, 1	70 (optimal)
4, 3	65

$$\text{profit}([1, 3]) = 30 + 25 = 55$$

$$\text{profit}([4, 1]) = 40 + 30 = 70$$



4.3 마감시간 있는 스케줄 짜기

- 마감시간 있는 스케줄 짜기: 탐욕법(The Greedy Approach)
 - 보상에 따라서 비오름차순으로 작업을 정렬한다.
 - 각 작업을 순서대로 하나씩 가능한 스케줄에 포함시킨다.

작업을 보상이 큰 것부터 차례로 정렬한다.

$S = \emptyset$;

while (답을 구하지 못했음):

다음 작업을 선택.

if (이 작업을 추가하면 S 가 적절하다)

이 작업을 S 에 추가.

if (더 이상 남은 작업이 없다)

답을 구했음.



4.3 마감시간 있는 스케줄 짜기



■ 적절함(feasibility)의 정의

- 어떤 순서(sequence)는 그 순서 내의 모든 작업이 데드라인 이전에 시작될 때 **적절한 순서(feasible sequence)**라 한다.
 - $[4, 1]$ 은 적절한 순서
 - $[1, 4]$ 는 부적절한 순서
- 어떤 집합(set)은 그 집합의 원소들로 하나 이상의 적절한 순서를 만들 수 있을 때 **적절한 집합(feasible set)**이라 한다.
 - $\{1, 4\}$ 는 적절한 집합: $[4, 1]$ 이라는 적절한 순서를 만들 수 있음
 - $\{2, 4\}$ 는 부적절한 집합: 적절한 순서를 만들 수 없음



4.3 마감시간 있는 스케줄 짜기

Job	Deadline	Profit
1	3	40
2	1	35
3	1	30
4	3	25
5	1	20
6	3	15
7	2	10

The jobs are already sorted by the profit

- $S = \phi$
- $S = \{1\}$, $[1]$ is feasible
- $S = \{1, 2\}$, $[2, 1]$ is feasible
- $S = \{1, 2, 3\}$, rejected
there is no feasible sequence for this set : $S = \{1, 2\}$
- $S = \{1, 2, 4\}$, **$[2, 1, 4]$** is feasible
- $S = \{1, 2, 4, 5\}$, rejected
 $S = \{1, 2, 4\}$
- $S = \{1, 2, 4, 6\}$, rejected
 $S = \{1, 2, 4\}$
- $S = \{1, 2, 4, 7\}$, rejected
 $S = \{1, 2, 4\}$ (feasible set)



4.3 마감시간 있는 스케줄 짜기

■ 보조정리 4.3

- S 를 작업의 집합이라고 할 때,
- " S 가 적절하다"는
- " S 에 속한 작업을 마감시간이 감소하지 않게 나열하여 얻은 순서가 적절하다"의 필요충분조건이다.

not feasible

→ $S = \{1, 2, 4, 7\}$
 (3) (1) (3) (2)

1	2	3	4
[2,	7,	1,	4]
(1)	(2)	(3)	(3)

not feasible



4.3 마감시간 있는 스케줄 짜기

Job	Deadline	Profit
1	3	40
2	1	35
3	1	30
4	3	25
5	1	20
6	3	15
7	2	10

The jobs are already sorted by the profit
The profits need not to be listed (Lemma 4.3)

- $J = [1]$
- $K = [2,1]$, K is feasible
 $J = [2,1]$
- $K = [2,3,1]$ is rejected, because
 K is not feasible
- $K = [2,1,4]$, K is feasible
 $J = [2,1,4]$
- $K = [2,5,1,4]$ is rejected
- $K = [2,1,4,6]$ is rejected
- $K = [2,7,1,4]$ is rejected
- $J = [2,1,4]$ is the final result
- Total Profit = $35+40+25 = 100$



4.3 마감시간 있는 스케줄 짜기



Algorithm 4.4: Scheduling with Deadlines

```
def schedule (deadline):  
    n = len(deadline) - 1  
    J = [1]  
    for i in range(2, n + 1):  
        K = insert(J, i, deadline)  
        if (feasible(K, deadline)):  
            J = K[:]  
    return J
```



4.3 마감시간 있는 스케줄 짜기



Algorithm 4.4: Scheduling with Deadlines

```
def feasible (K, deadline):  
    for i in range(1, len(K) + 1):  
        if (i > deadline[K[i - 1]]):  
            return False  
    return True
```

```
deadline = [0, 3, 1, 1, 3, 1, 3, 2]  
K = [2, 1]  
print(K, feasible(K, deadline))
```



4.3 마감시간 있는 스케줄 짜기



Algorithm 4.4: Scheduling with Deadlines

```
def insert(J, i, deadline):
    K = J[:]
    for j in range(len(J), 0, -1):
        if (deadline[i] >= deadline[K[j-1]]):
            j += 1
            break
    K.insert(j - 1, i)
    return K
```

```
J = [1]
K = insert(J, 2, deadline)
print(K)
```



4.3 마감시간 있는 스케줄 짜기



```
deadline = [0, 3, 1, 1, 3, 1, 3, 2]
profit = [0, 40, 35, 30, 25, 20, 15, 10]
J = schedule (deadline)
print("Schedule:", J)

maxprofit = 0
for job in J:
    maxprofit += profit[job]
print("Max Profit =", maxprofit)
```



주니온TV@Youtube

자세히 보면 유익한 코딩 채널

<https://bit.ly/2JXXGqz>

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
(다운로드 주소는 영상 하단 설명란 참고)

<https://bit.ly/3fN0q8t>