5. Greedy algoritm

5.0 Basics

5.1 Minimum spanning trees

5.2 Knapsack problem

5.3 Job sequencing with deadline

5.4 Optimal merge patterns

5.5 Huffman encoding

5.3 Job sequencing with deadline

- Problem:
  - We are given n jobs.
  - Each job i has a deadline $d_i \geq 0$ and a profit $p_i \geq 0$.
  - For any job i, the profit p_i is earned if and only if the job is completed by its deadline.

5.3 Job sequencing with deadline

- Problem:
  - In order to complete a job, one has to process the job on a machine for one unit of time.
  - Feasible solution for this problem is a subset, J, of jobs such that each job in this subset can be completed by its deadline.
  - The value of J is $\Sigma_i p_i$.
  - Optimal solution: J with maximum value

5.3 Job sequencing with deadline

- Example:
  - n = 4, $(p_1, p_2, p_3, p_4) = (100, 10, 15, 27)$ and $(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$.
  - What are the feasible solutions and their values?

5.3 Job sequencing with deadline

- Solution strategy
  - J is a set of k jobs and $\sigma = i_1, i_2, \ldots, i_k$ be a permutation of jobs such that $d_{i1} \leq d_{i2} \leq \ldots \leq d_{ik}$.
  - J is a feasible solution if and only if the jobs in J can be processed in the order σ without violating any deadline.
  - $D(J(1)) \leq D(J(2)) \leq \ldots \leq D(J(k))$.
  - $D(J(r)) \geq r$, for $1 \leq r \leq k$.

5.3 Job sequencing with deadline

- Solution strategy
  - We assume that the jobs are sorted such that $p_1 \geq p_2 \geq \ldots \geq p_n$.
  - We assume that min { D(i) } = 1.

  - **Select the job in the non-ascending order of profit.**
  - **If the pre-selected jobs can yield, then make them yield as much as possible.**

5.3 Job sequencing with deadline

- Job scheduling

```
void JobSchedule( int D[], int J[], int n )
//  initially jobs are sorted such that p1 ≥ p2 ≥ … ≥ pn
{
    D[0] ← J[0] ← 0;
    k ← 1; J[1] ← 1;
    for ( i ← 2 to n by 1 )
        r ← k;
        while ( D[J[r]] > D[i] and D[J[r]] != r )
            r ← r - 1;
        if ( D[J[r]] <= D[i] and D[i] > r )
            for ( l = k; l >= r + 1 by -1 )
                J[l+1] ← J[l];
            J[r+1] ← i; k ← k+1;
}
```

5.3 Job sequencing with deadline

- Job scheduling

```
void JobSchedule( int D[], int J[], int n )
//  initially jobs are sorted such that p1 ≥ p2 ≥ … ≥ pn
{
    D[0] ← J[0] ← 0;
    k ← 1; J[1] ← 1;
    for ( i ← 2 to n by 1 )
        r ← k;
        while ( D[J[r]] > D[i] and D[J[r]] != r )
            r ← r - 1;
        if ( D[J[r]] <= D[i] and D[i] > r )
            for ( l = k; l >= r + 1 by -1 )
                J[l+1] ← J[l];
            J[r+1] ← i; k ← k+1;
}
```

Find the one that can yield

5.3 Job sequencing with deadline

- Job scheduling

```
void JobSchedule( int D[], int J[], int n )
//  initially jobs are sorted such that p1 ≥ p2 ≥ … ≥ pn
{
    D[0] ← J[0] ← 0;
    k ← 1; J[1] ← 1;
    for ( i ← 2 to n by 1 )
        r ← k;
        while ( D[J[r]] > D[i] and D[J[r]] != r )
            r ← r - 1;
        if ( D[J[r]] <= D[i] and D[i] > r )
            for ( l = k; l >= r + 1 by -1 )
                J[l+1] ← J[l];
            J[r+1] ← i; k ← k+1;
}
```

If feasible, insert the job

5.3 Job sequencing with deadline

- Example:
  - $n = 5$, $(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$, $(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$

Condition of feasibility?

$$D(J(1)) \le D(J(2)) \le \dots \le D(J(k))$$
$$\&$$
$$D(J(r)) \ge r$$

5.3 Job sequencing with deadline

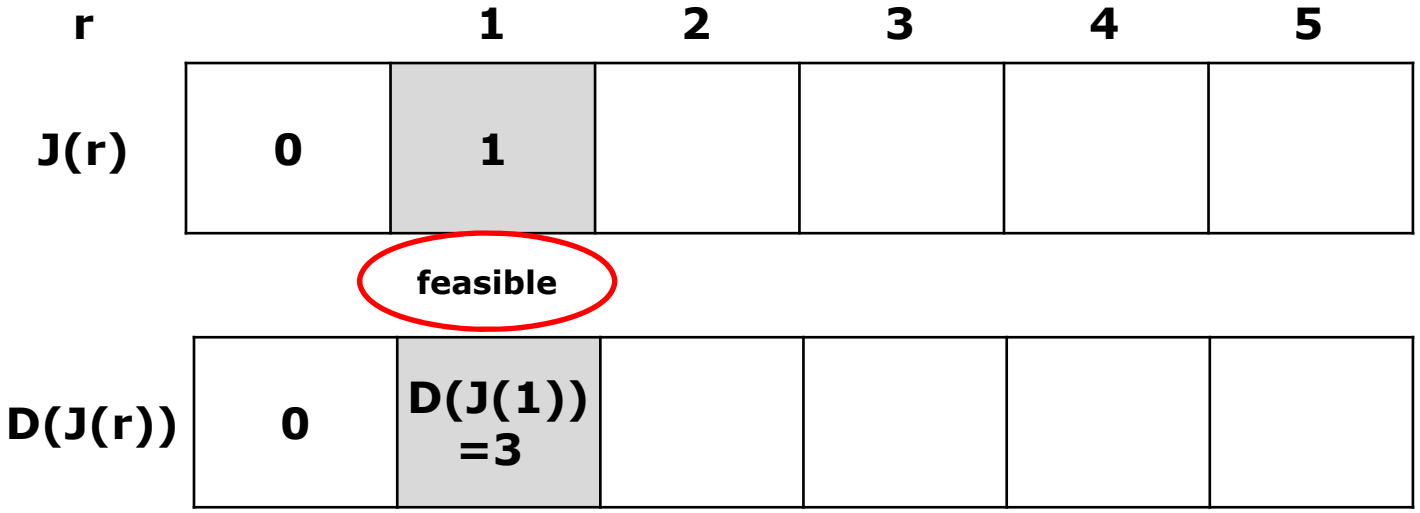
- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \dots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$



5.3 Job sequencing with deadline

- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \dots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$

r		1	2	3	4	5
J(r)	0	1	2			

feasible

D(J(r))	0	D(J(1)) =3	D(J(2)) =4			

5.3 Job sequencing with deadline

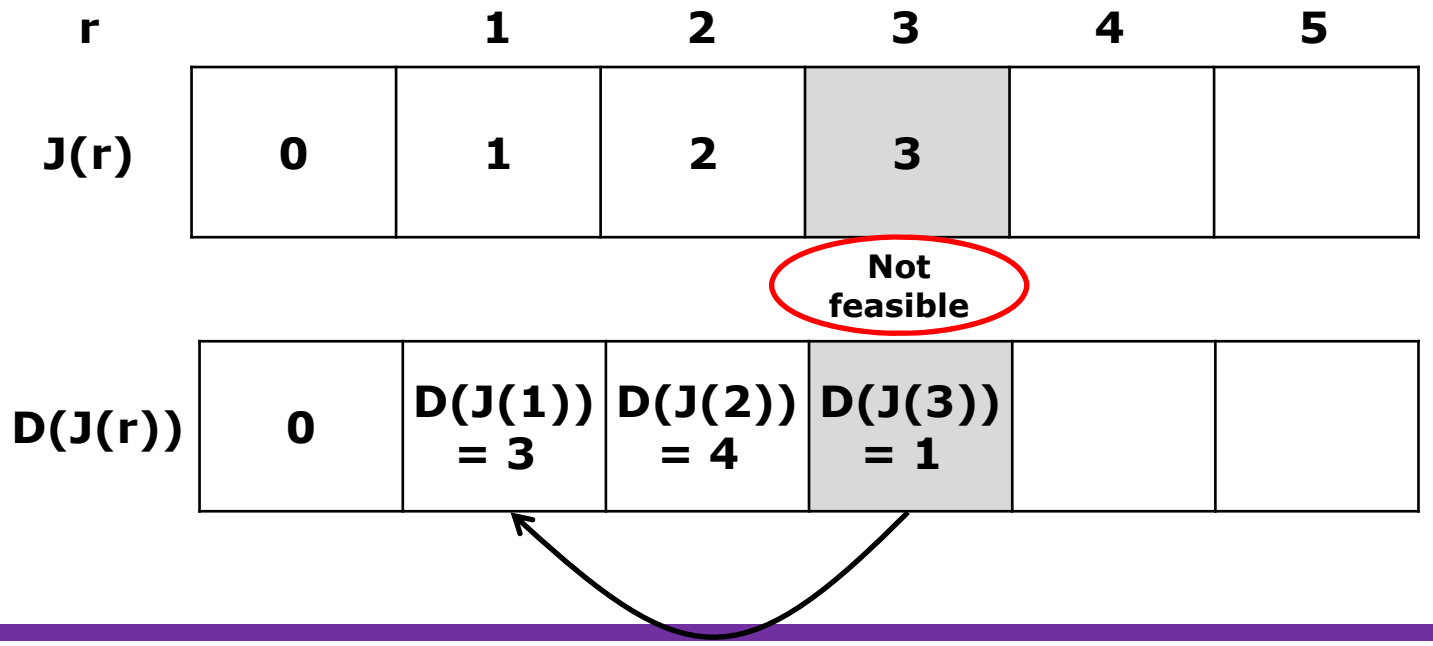
- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \dots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$



5.3 Job sequencing with deadline

- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \ldots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$

r		1	2	3	4	5
J(r)	0	3	1	2		
D(J(r))	0	D(J(1)) = 1	D(J(2)) = 3	D(J(3)) = 4		

feasible

5.3 Job sequencing with deadline

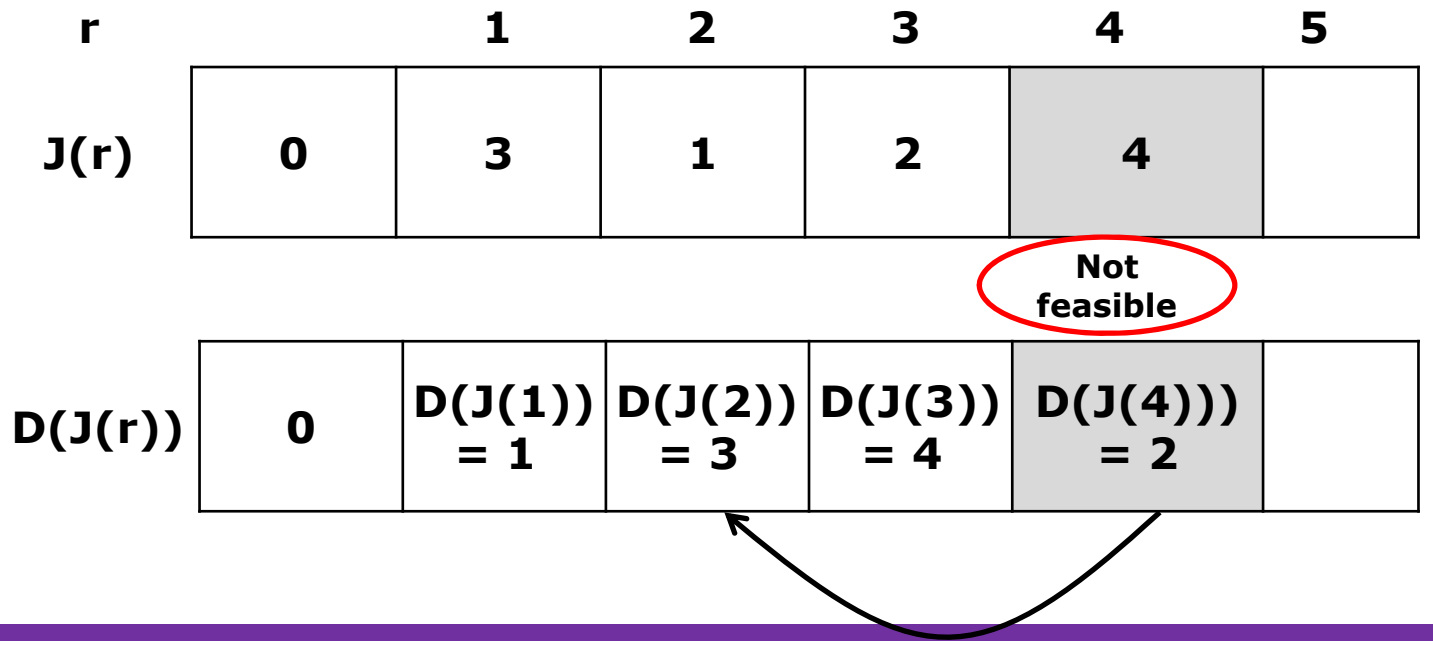
- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \ldots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$



5.3 Job sequencing with deadline

- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \ldots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$

r		1	2	3	4	5
J(r)	0	3	4	1	2	
D(J(r))	0	D(J(1)) = 1	D(J(2)) = 2	D(J(3)) = 3	D(J(4))) = 4	

feasible

5.3 Job sequencing with deadline

- Example:
  - n = 5,

Condition of feasibility?

$D(J(1)) \le D(J(2)) \le \ldots \le D(J(k))$
&
$D(J(r)) \ge r$

```
r ← k;
while ( D[J[r]] > D[i] and D[J[r]] != r )
    r ← r - 1;
if ( D[J[r]] <= D[i] and D[i] > r )
    for ( l = k; l >= r + 1 by -1 )
        J[l+1] ← J[l];
    J[r+1] ← i; k ← k+1;
```

$(p_1, p_2, p_3, p_4, p_5) = (20, 15, 10, 5, 1)$,
$(d_1, d_2, d_3, d_4, d_5) = (3, 4, 1, 2, 5)$

r		1	2	3	4	5
J(r)	0	3	4	1	2	5

feasible

D(J(r))	0	D(J(1)) = 1	D(J(2)) = 2	D(J(3)) = 3	D(J(4))) = 4	D(J(5))) = 5

5.3 Job sequencing with deadline

다음 job sequencing에 대한 질문 중 옳은 것을 모두 고르시오.

(a) J[i] = 3의 의미는 첫번째로 하는 일이 세번째 job이라는 뜻이다.

(b) D[J[1]] = 2의 의미는 첫번째로 하는 일이 무엇인지 모르겠지만, 그 일의 deadline은 2라는 뜻이다.

(c) JobSchedule 알고리즘에서 while-loop의 의미는 양보하는 과정을 나타낸다.

(d) Job sequencing에서 끝나고 나면 profit이 더 작은 job이 더 큰 job 보다 먼저 수행되지 않는다.

(e) Job sequencing이 끝나고 나면 deadline이 더 늦은 job이 더 빠른 job보다 먼저 수행되지 않는다.