Assignment Problem

bipartite graph

✓ Linear Assignment Problem

- LAP(Linear Assignment Problem): 동일한 수의 작업자를 작업에 <u>고유</u>하게 매칭하고 매칭의 전체 <u>비용을 최소화</u>
  - <u>Hungarian Algorithm != Greedy Algorithm</u>
- 시간, 비용이라는 현실적인 문제 때문에 이분 그래프를 이용해서 최적화 찾음 = 복잡한 문제를 휴리스틱하게, 근사치로 구함
  - 최적화, 최적해, 최대(최소) 할당, 최대(최소) 비용
- 인공지능 개발 관련성: 자동 작업 할당 AI 시스템 외 로봇, tracking 분야에서 사용됨.
  - Openpose에서는 가중치(PAF)를 통해 예측한 관절을 person마다 matching

✓ 이분 그래프

- 가중치 없는 이분 그래프: 제약 조건내에서 최대 할당이 목적, 1:1 대응 > 1:1 함수
- 가중치 있는 이분 그래프(가중 이분 그래프): 최소 비용으로 완전 할당이 목적, 1:1 대응
  - 최소 비용에는 여러 가지 가능한 할당(조합)이 있을 수 있지만 우리는 하나만 찾는 데 관심이 있음.
  - 1:1 대응: 한 명의 작업자가 모든 작업자에게 작업을 분배하는 것보다 혼자서 모든 작업을 더 효율적으로 수행할 수는 없다고 가정

Hungarian Algorithm

가중치 있는 이분 그래프 - perfect matching

✓ Hungarian Algorithm or Kuhn-Munkres 알고리즘

- 할당 문제의 최적해 찾기가 목적. 헝가리 수학자인 Harold Kuhn이 1955년에 제안하고, 1957년에 James Munkres가 보완
- 행렬로 표현 가능: 행은 작업자(일), 열은 일(작업자), 행렬의 값은 작업비용을 의미함
- 모든 비용은 0이랑 같거나 크다. (실제로 음수에 적용 가능)
- 1: 1 대응
- n x m 비용행렬
  - **if n = m,** 균형 할당 문제, 균형 할당 문제에 대해서는 최적해(optimal solution)를 항상 찾을 수 있다고 알려짐
  - **if n != m,** 불균형 할당 문제, 균형 할당문제로 만드는 과정 추가 후, 헝가리 알고리즘 적용
    - openpose에서는, 한 사람의 모든 관절이 보이지 않는다면, 불균형 할당 문제로 볼 수 있을 것으로 예상함 (Opinion)

✓ “Pre-Payment” 개념을 통해 이해가 쉬움 - 선지불, 준비금

- 입력으로 주어지는 행렬의 행과 열에 같은 수를 더하거나 빼도 결과에는 영향을 미치지 않음
  - 증명됨: Local Improvement(헝가리안 알고리즘)은 유한번 반복, pre-payment 관계없이 같은 결과의 비용행렬
- **0의 값을 갖는 원소(비용)만**을 가지고 1:1 할당이 가능하면, optimal solution이 됨.

✓ 쾨니그의 정리(Kőnig's theorem): 최소 vertex cover가 만족할 때까지 반복

- 모든 bipartite graph에서 (maximum-size matching의 크기 == **minimum-size** vertex cover의 크기)
- vertex cover: 정점을 선택, 모든 간선을 커버할 수 있어야함. (모든 정점을 선택 - 자명한 vertex cover)
  - 이분법칙에서 최소 vertex cover: n (=m)

$m \times n$ $(m = n)$ 비용 행렬
Step 1. /* 수행 복잡도 : $O(n^2)$
    for $i = 1$ to m
        최소 비용 ${}_{\min}c_i$ 를 찾음. /* 각 행에서 최소 비용 ${}_{\min}c_i$ 을 찾음.
        for $j = 1$ to n
            $c_{ij} = c_{ij} - {}_{\min}c_i$. /* 각 열의 비용에서 최소 비용을 감산함.
        end
    end
    for $j = 1$ to n
        최소 비용 ${}_{\min}c_j$ 를 찾음. /* 각 열에서 최소 비용 ${}_{\min}c_i$ 을 찾음.
        for $i = 1$ to m
            $c_{ij} = c_{ij} - {}_{\min}c_j$. /* 각 행의 비용에서 최소 비용을 감산함.
        end
    end
Step 2. /* 수행 복잡도 : $O(n^2)$
    모든 $c_{ij} = 0$ 를 포함하는 최소한의 선을 그음.
    /* 선의 수 : $|l|$
    if $|l| = m$ then go to Step 4.
    else if $|l| < m$ then go to Step 3.
Step 3. /* 수행 복잡도 : $O(n^3)$
    선에 포함되지 않은 $c_{ij} > 0$ 들인 감소된 비용 행렬 (Reduced Cost Matrix)에서 최소 비용 ${}_{\min}c_{ij}$ 를 찾음.
    $c_{ij} > 0$: $c_{ij} = c_{ij} - {}_{\min}c_{ij}$.
    행과 열의 선이 교차된 c_{ij} : $c_{ij} = c_{ij} + c_{ij}$.
    go to Step 2.
Step 4. 각 열에서 0이 중첩되지 않게 1개씩 선택, 선택된 셀의 비용을 모두 더하여 최적해 z를 얻음.

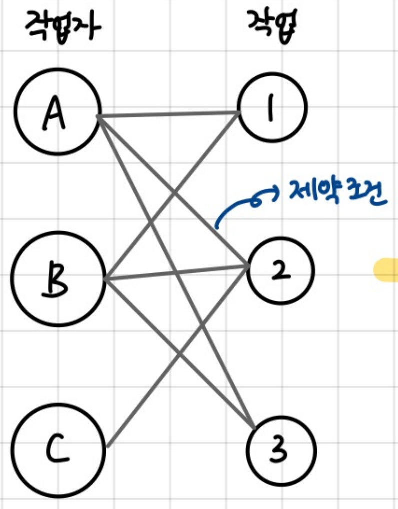
그림 1. Hungarian 알고리즘
Fig. 1. Hungarian Algorithm

https://koreascience.kr/article/JAKO201229765634714.pdf

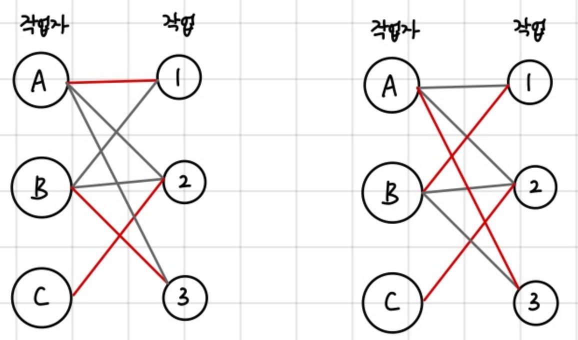
헝가리안 알고리즘 - 필기

⓪ 가중치 없는 그래프 : 제약 조건 내에서 최대 할당이 목적

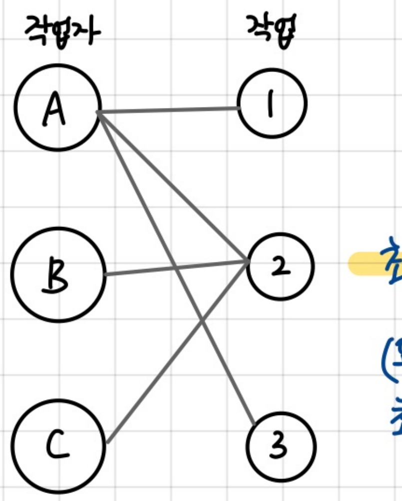
1) case 1



유일한해X

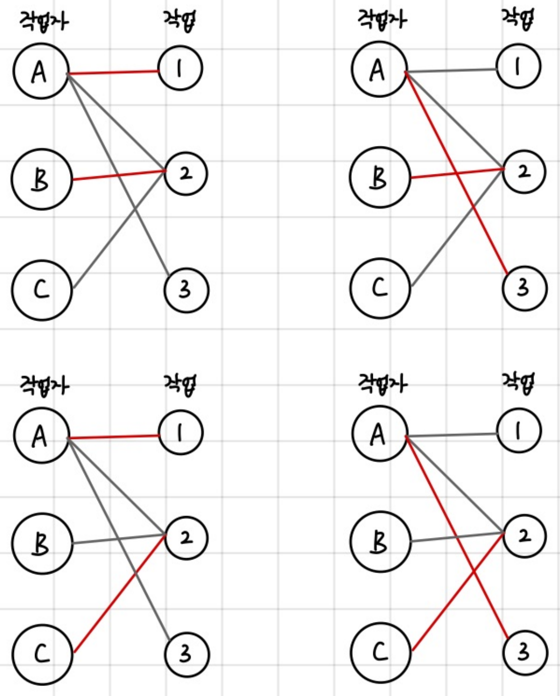


2) case 2



최대할당

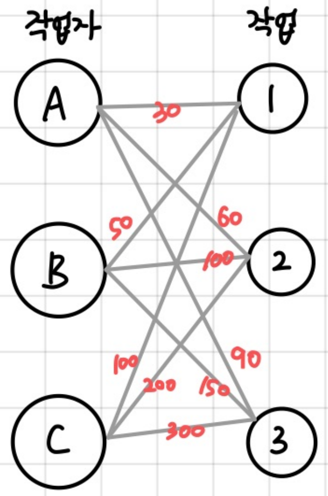
(완전할당:3)
최대할당수:2



⓪ 가중치 있는 그래프 : 최소 비용, 완전할당이 목적

3개의 작업을 3개의 작업자에게 중복되지 않게 할당하여
총 작업비용을 최소화시키는 제약조건을 만족하는 최적해를 찾아야 함

1)



행렬로 표현 가능
(인접행렬)

	1	2	3
A	30	60	90
B	50	100	150
C	100	200	300

* 행: 작업자
* 열: 작업
* 행렬 값: 비용

⓪ 헝가리안 알고리즘

1) ↱ 쾨니그의 정리 : maximum matching의 개수 (크기) = minimum vertex cover의 개수 (크기)

모든 비용은 0이상 → (행렬값 =0)의 조합이 1:1 대응이 성립하면, 당연하게 최소 비용으로 matching

	1	2	3
A	0	*	*
B	*	0	*
C	*	*	0

→ (A→1, B→2, C→3)

(* : 0보다 큰 수)

	1	2	3
A	*	*	0
B	0	0	*
C	0	0	0

(A→3, B→1, C→2) or (A→3, B→2, C→1)

최소비용에는
여러가지 가능한 할당 (조합) 有
But, 우리는 하나만 찾는데 관심 있음

	1	2	3
A	0	0	0
B	0	*	*
C	0	*	*

→ 당연한 최소비용 할당 X

(3 명이 동시에 비용 0이 되도록 하는 서로 다른 작업을 고르는 방법이 없음)

⇓

vertex cover = 2

목적 - (최소 vertex cover = 3)

⇓

perfect matching 불가

2) 'Pre-Payment' 개념 - 선지급, 준비금

작업자 \ 작업	1	2	3
A	30	60	90
B	50	100	150
C	100	200	300

1) 행 기준, 각 행의 최소비용만큼 뺀다.

		1	2	3
30	A	0	30	60
50	B	0	50	100
100	C	0	100	200

• A는 30, B는 50, C는 100만큼 선지급했다.
⇒ A는 어떤 업무를 하든 적어도 30은 필요로함 → 선지급
⇒ 선지급했으니, A행은 30만큼 감산한다.

(변환된 비용행렬과 original 비용행렬 동치)

⇒ '당연한' 최소비용 할당을 위해서
세명이 동시에 비용이 0이 되도록
하는 서로 다른 작업을 골라야 함.
(1:1 대응)

2) 열 기준, 각 열의 최소비용만큼 뺀다.

		1	2	3
		0	30	60
30	A	0	0	0
50	B	0	20	40
100	C	0	70	140

• 2는 30, 3은 60만큼 준비금으로 빼둔다.
⇒ 작업 2는 누가 일하든 적어도 30을 줘야함 → 준비금
⇒ 준비금을 빼두었으니, 2열은 30만큼 감산한다.

*쌍대문제 (duality)

비용함수의 합을 최소화
(⇒ 0의 개수를 늘린다.
⊜ pre-payment를 최대화)

↱ 증명됨.

global optimal solution을 찾을 수 있다 ↜ Local Improvement

3) (- 표시 제외하고, 관심대상)

①

		1	2	3
		0	30	60
30	A	~~0~~	~~0~~	~~0~~
50	B	0	20	40
100	C	0	70	140

⇒ 최소비용만큼 뺀다

		1	2	3
		0	30 +20	60
30	A	0	-20	0
50	B	0	0	40
100	C	0	50	140

• 0을 제외하고, 최소비용
(최소 포함 행렬 기준)

↱ B가 2를 선택 : 가장 작은 비용

0 0 * (!)
0 0 0
* * *

⇓ 음수 없애야..

②

		1	2	3
		0	50	60
30 -20	A	20	0	20
50	B	0	0	40
100	C	0	50	140

⇐

		1	2	3
		0	50	60 +20
10	A	20	0	0
50	B	0	0	20
100	C	0	50	120

①의 Σ cost = 270
-60
①의 pre payment = 270

②의 Σ cost = 210
+20
②의 pre payment = 290

∴ (A→3)+(B→2)+(C→1)
= (80+10) + (50+50) + (100+0)
= 90+100+100 = 290

		0	50	60 +20
		1	2	3
10	A	20	0	0
50	B	0	0	20
100	C	0	50	120

⓪ 가중치 없는 이분그래프 매칭의 대표 알고리즘

⇒ Hopcroft-Karp Algorithm

⇒ 일반적인 경우 최대할당이 목적이지만

Hungrian Algorithm 에서 완전할당이 목적이 모든 vertex는 연결 (1:1대응) 되어야 함
perfect matching