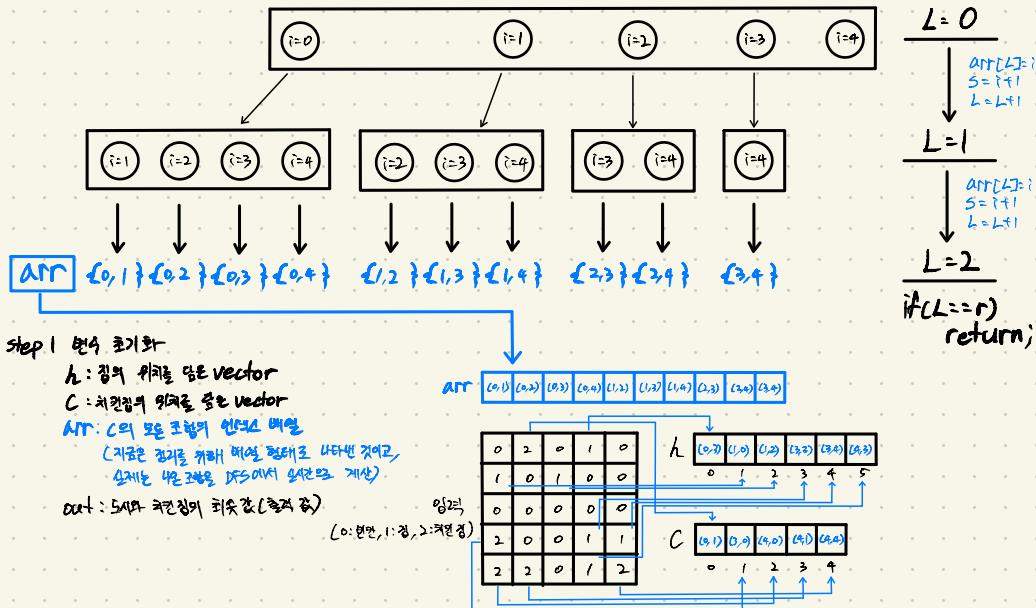


# 백준 15686, 치킨 배달

모든 조합 구하기

DFS ( $s=0, L=0$ ),  $nCr (n=5, r=2)$



step 2 ARR 전체를 돌면서 최소값을 찾는다.

pseudo code

```

int out = INT_MAX; // 흔적값 초기화
for(int i=0; i<arr.size();) // 전체 배열 순회
{
    int sum = 0; // 현재 조합의 거리의 총
    // 침위치
    for(int j=0; j<h.size();)
    {
        int min_dist = INT_MAX; // 각 침마다 최초거리 찾을 변수
        // 침위치 있는 치킨집
        for(int k=0; k<arr.size())
        {
            pair<int, int> h-point = arr[j]; // 침 위치
            pair<int, int> C-point = arr[k]; // 치킨 집 위치

            int dist = abs(h-point.first - C-point.first) +
                abs(h-point.second - C-point.second); // 거리 계산
            min_dist = min(min_dist, dist); // 최소 거리 업데이트
        }
        sum += min_dist; // 침위치 있는 치킨집 거리 합
    }
    out = min(out, sum); // 흔적 값 갱신
}
    
```

ARR:  $\{(0,1) (0,2) (0,3) (0,4) (1,2) (1,3) (1,4) (2,3) (2,4) (3,4)\}$

$h$ :  $\{(0,0) (1,0) (2,0) (3,0) (4,0) (0,1) (1,1) (2,1) (3,1) (4,1)\}$

$C$ :  $\{(0,1) (0,2) (0,3) (0,4) (1,2) (1,3) (1,4) (2,3) (2,4) (3,4)\}$

백준 1197, 최소 스패닝 트리

### step 1 면적을 계산

NET: 전면 소매처 도적 대처  
WPS: WPS 차장 경보 설정기 Vector

Sum: 47ms 47ms

2008. 3월 드래프트 회의 개정안

1	2	1	3	2	3
1		3		2	

(문서 파일에 2행짜리 행  
넣어보면됨)

MST	1	2	3	Graph
	(1)	(2)	(3)	

Step 2 모든 변수를 가중치 오름 차순으로 정렬

vec	1	2	2	3	1	3
	1		2		3	

MST	1	2	3	Graph
	(1)	(2)	(3)	

Step 3 모든 VecDoJ에서 침경(1과 2)을 Union-Find 알고리즘으로 연동한다  
이후, 정답은  $\rightarrow$  2 이상인 경우를 반환한다

vec	1	2	2	3	1	3
	1	2	2	3	1	3

The diagram shows a stack structure with three elements: 1, 2, and 3. The element 2 is highlighted with a blue box. A pointer labeled "sum21" points to the value 2.

Union-Find 05.22(3)

34 [Find \(2nd\)](#) < 0.9 3E-32

THE LIMA'S A MELLO

```

    if (MST[a] == a)
    {
        return MST[a];
    }
    return MST[a] = Find(MST[a]);
}

```

```
void Union(int a, int b)
```

```

    a=find(a);
    b=find(b);
    if (a!=b) // a와 b가 서로 연결되어 있지 않으면
    {
        MST[a]=b; // a와 b를 연결
    }
}

```

step 4 그 다음, VCE[기본]에서도 두 가지로 Union-Find 알고리즘을 선택한다.

vec	1	2	2	3	1	3
	1		2		3	

The diagram illustrates the state of the MST array and a graph structure. The MST array is shown as a row of three boxes, each containing the value 3. To the right, a graph is depicted with three nodes labeled 1, 2, and 3. Node 1 is at the top, node 2 is at the bottom left, and node 3 is at the bottom right. A blue arrow points from node 2 to node 3, with the label "child 1" written below it.

step 4 마사화 버터[2] 채비 경향(과 3의 차트가) 결코므로 계산하여 한다.

$\text{vec}$	1	2	2	3	1	3
	1		2		3	

MST	3	3	3
-----	---	---	---

```

graph LR
    N1((1)) ---|1| N2((2))
    N1 ---|2| N3((3))
    N2 ---|2| N3
  
```

백준 1516, 토마토

step1 번역 초기화  
day: 오늘도가 목은데 걸리는 148, 053 3334  
box: 흐기 토끼로 살해, 틈새 암자  
q: 정수를 토끼로 죄를 짓는 문제

$b_{00}$	0	-1	0	0	0	0
$b_{10}$	-1	0	0	0	0	0
$b_{01}$	0	0	0	0	0	0
$b_{11}$	0	0	0	0	0	1

day	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

9

--	--	--	--	--	--	--

step2 box 뒤에를 확인하여 1인 경우 차포를 1개 push

$b_{00}$	0	-1	0	0	0	0
	-1	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	1

$(5, 3)$

day	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

9 (5,35) . . . . .

Step3 9에서 차트를 꺼내어 출판권을 갖추  
box 끝에 예전것을 빼고 깊은 10 번정  
수로, 아래에 출판권 + 1을 표시한다.

$\text{box}$	$0$	$-1$	$0$	$0$	$0$	$0$
	$-1$	$0$	$0$	$0$	$0$	$0$
	$0$	$0$	$0$	$0$	$0$	$(1, \frac{1}{2})$
	$0$	$0$	$0$	$0$	$-1$	$(4, -3)$
	$0$	$0$	$0$	$1$	$0$	$(3, 3)$

$\begin{matrix} \text{day} \\ \text{---} \end{matrix}$	0	0	0	0	0	0
$\text{---}$	0	0	0	0	0	0
$\text{---}$	0	0	0	0	0	1
$\text{---}$	0	0	0	0	1	0

$q$    $\leftarrow q_{\text{front}} := (5, 2)$

Step 4 Step 3 을 완료 (9% size 가 0일 때 까지)

box	0	-1	0	0	0	0
	-1	0	0	0	0	0
	0	0	0	0	1	1
(1)	0	0	0	1	1	1
			(2,3)	(4,3)	(5,2)	(5,3)

day	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	2	1
	0	0	0	2	1	0

q. front: (4, 3)

**Step5** 허용한 폴더  
#\* 모든 파일을 빼고 box 폴더에 다른 모든 경로는 굽이치로 이름을 바꾸고  
box 폴더에 있으면 day 폴더를 생성한 폴더를 출력

box	0	→				
-1						

day	0	0	6	5	4	3
	0	6	5	4	3	2
	6	5	4	3	2	1
	5	4	3	2	1	0

## 백준 2178, 미로 탐색

### Step 1 맵과 초기화

- 이미지: 현재 칸의 상태를 같은 배열(0=벽, 1=통로, 2=출구, 3=목표)으로 표시하는 배열
- dist: 시작 위치에서 최단거리를 저장하는 배열(0으로 초기화)
- q: 찾을 수 있는 경로 좌표를 담은 Queue

arr	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	dist	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1																																														
1	0	1	0	1	0																																														
1	0	1	0	1	1																																														
1	1	1	0	1	1																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														

q	<table border="1"> <tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> </table>						

### Step 2 시작 좌표 L(1,1)을 큐에 push 한다.

제출 결과: 미로 탐색 성공

arr	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	dist	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1																																														
1	0	1	0	1	0																																														
1	0	1	0	1	1																																														
1	1	1	0	1	1																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
		push	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1																								
1	0	1	1	1	1																																														
1	0	1	0	1	0																																														
1	0	1	0	1	1																																														
1	1	1	0	1	1																																														

q	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1
1	0	1	1	1	1		

### Step 3 arr[1][1]가 목적지인가? 경우에는 경로 표출 후 pop 한다.

제출 결과: 미로 탐색 성공

arr	<table border="1"> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	0	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	dist	<table border="1"> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1																																														
1	0	1	0	1	0																																														
1	0	1	0	1	1																																														
1	1	1	0	1	1																																														
1	0	0	0	0	0																																														
2	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
		push	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1																								
1	0	1	1	1	1																																														
1	0	1	0	1	0																																														
1	0	1	0	1	1																																														
1	1	1	0	1	1																																														

q	<table border="1"> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	0	1	1	1	1
1	0	1	1	1	1		

### Step 4 arr[4][4]가 목적지인가? 경우에는 경로 표출 후 pop 한다.

제출 결과: 미로 탐색 성공

arr	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	dist	<table border="1"> <tr><td>1</td><td>0</td><td>9</td><td>10</td><td>11</td><td>12</td></tr> <tr><td>2</td><td>0</td><td>8</td><td>0</td><td>12</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>7</td><td>0</td><td>13</td><td>14</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>0</td><td>14</td><td>15</td></tr> </table>	1	0	9	10	11	12	2	0	8	0	12	0	3	0	7	0	13	14	4	5	6	0	14	15
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
0	0	0	0	0	0																																														
1	0	9	10	11	12																																														
2	0	8	0	12	0																																														
3	0	7	0	13	14																																														
4	5	6	0	14	15																																														
			<table border="1"> <tr><td>1</td><td>0</td><td>9</td><td>10</td><td>11</td><td>12</td></tr> <tr><td>2</td><td>0</td><td>8</td><td>0</td><td>12</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>7</td><td>0</td><td>13</td><td>14</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>0</td><td>14</td><td>15</td></tr> </table>	1	0	9	10	11	12	2	0	8	0	12	0	3	0	7	0	13	14	4	5	6	0	14	15																								
1	0	9	10	11	12																																														
2	0	8	0	12	0																																														
3	0	7	0	13	14																																														
4	5	6	0	14	15																																														

q	<table border="1"> <tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> </table>						

백준 1753, 최단거리

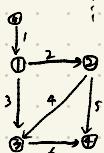
기워드  
다익스트라 알고리즘  
최단거리

정의

5(경점 A) 6(경점 B)

1(시작점)

5(경점 A) 1(경점 B) 1(경점 C)



Step1. 최단거리를 나타내는 dist 배열을 무한대로 초기화

INF	INF	INF	INF	INF
1	2	3	4	5

Step2. dist[시작점, 1]을 0으로 초기화

0	INF	INF	INF	INF
1	2	3	4	5

Step3. dist[]가 1→2 거리를 따른 값을 빼서 dist[2]와 비교하여 작은 값으로 업데이트

이때, 2를 통한 거리를 빼면 1→2→3→4→5로 5단계 방식이變成 1→2→5로 push 한다.

0	2	INF	INF	INF
1	2	3	4	5

Step4. 최단거리가 0이 될 때까지 Step3를 반복

0	2	3	7	INF

백준 11675, 탐욕DP

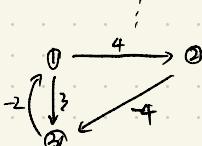
기워드

백만·포도 알고리즘

정의

3(경점 A) 4(경점 B)

1(경점 A) 2(경점 B) 4(경점 C)



Step1. dist 배열 (1부터 각 경점 까지 걸리는 시간)을 무한대로 초기화

INF	INF	INF
1	2	3

Step2. dist[시작점, 1]을 0으로 초기화

0	INF	INF
1	2	3

Step3. 간선 수-1 × 경점 수-1 번큼 가중치와 가중치를 더한 값을 서로 찾고 계속 업데이트  
한 dist가 INF는 경우 원래는 계산하지 않기

0	INF	INF
1	2	3

 → 

0	9	INF
1	2	3

 → 

0	4	3
1	2	3

Step4. 마지막으로 경점 수만큼 연례에 푸는 데 기본값보다 적어지면 사이를 이어갈

0	4	3

 ⇒ 

0	0	3

같이 출력적으로 푸기 사용을 권장

