Lab3 Solution

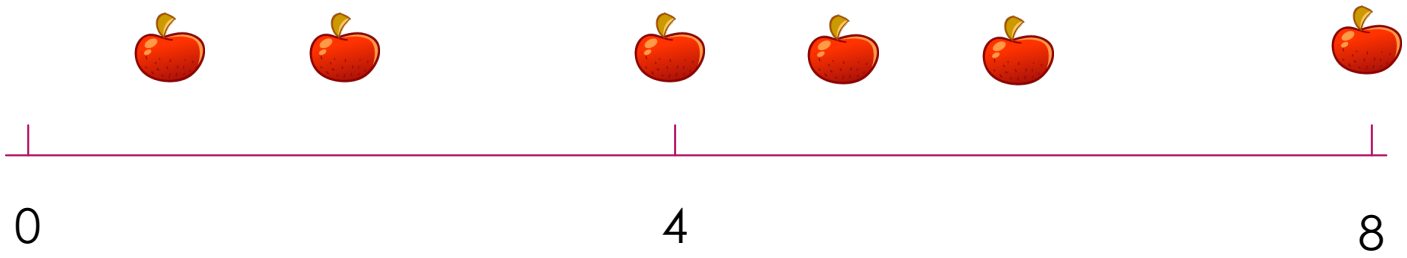
YAO ZHAO

Lab3.A Careless Boy

- Dy is a careless boy. One day, on the way of delivering apples, Dy pays all attention on playing Nihsneg on his phone. Dy is so focused on playing that he doesn't realize that all apples drop out from the baskets! Now Dy needs to collect apples along the way and take them back to the truck. All tools he can use are the baskets on the truck and himself.
- Dy need to set off from the truck to collect apples with his baskets. Every basket has its own volume v_i, representing the number of apples it can hold. At any time, Dy can carry only one basket. The truck and the apples are on the same line. The line can be considered as a number axis and the truck is its origin. Every apple has its own position p_i on the number axis. Once Dy passes by an apple, he can pick up the apple and keep collecting apples if the basket is not full. Once the basket is full, Dy must take it back to the truck.
- Dy wonders how far it would take him to fill up all the baskets (If the total number of apples is less than the sum of basket volume ($n < \sum_1^{\mathrm{m}} v_i$), then Dy just need to collect all apples). And he wants the total distance to be as small as possible.

Input:

```
6 3
2 1 3
1 2 4 5 6 8
```



2→1→3: 2+2+4+4+8+8 = 28
1→2→3: 1+1+4+4+8+8 = 26
3→2→1: 4+4+6+6+8+8 = 36
…

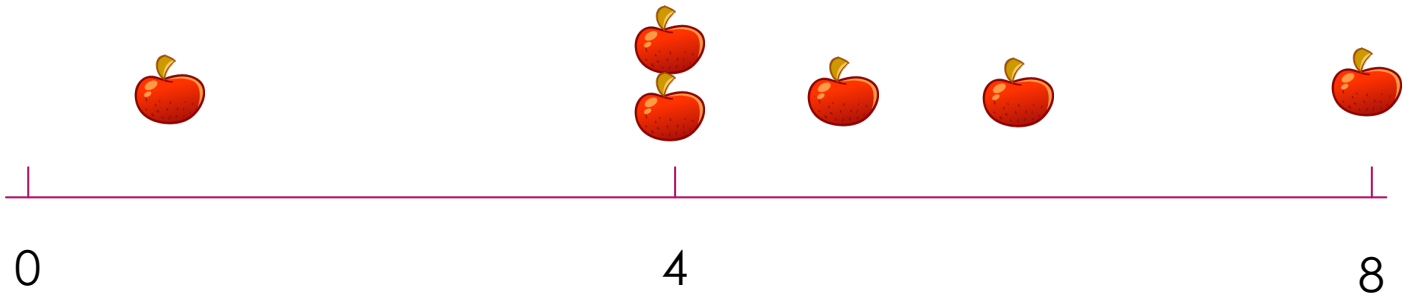
All possible total distance, 26 is the smallest.

Output:

```
26
```

Input:

```
6 3
1 2 2
1 4 4 5 6 8
```



1→2→2: 1+1+4+4+6+6 = 22
2→1→2: 4+4+4+4+6+6 = 28
2→2→1: 4+4+5+5+6+6 = 30

All possible total distance 22 is the smallest.

Output:

22

Suppose we always collect apples from near to far.

1st round: assume the first basket volume is v_{j1}
the distance of 1st round : 2*p[v_{j1}]
2nd round: assume the second basket volume is v_{j2}
the distance of 2nd round : 2*p[$v_{j1} + v_{j2}$]

…

kth round: assume the kth basket volume is v_{jk}
the distance of kth round : 2*p[$v_{j1} + v_{j2}$+… v_{jk}]

total distance = 2*(k*p[v_{j1}]+(k-1)*(p[$v_{j1} + v_{j2}$]- p[v_{j1}])+… +p[v_{jk}])

Obviously, if we want the total distance to be as small as possible, the p[v_{j1}] should be the smallest, the p[v_{j2}] should be the second smallest, etc.

$n \geq \sum_{1}^{m} v_i$ fill up all the baskets

v_{j1} should be the smallest volume of all the baskets,
v_{j2} should be the second smallest volume of all the baskets,
…
etc.
k = m

$n < \sum_{1}^{m} v_i$ collect all apples

means some baskets may empty, 1 basket may not full
v_{jk} should be the largest volume of all the baskets, to take the farthest apples
v_{jk-1} should be the second largest volume of all the baskets,
…
etc.
k ≤ m

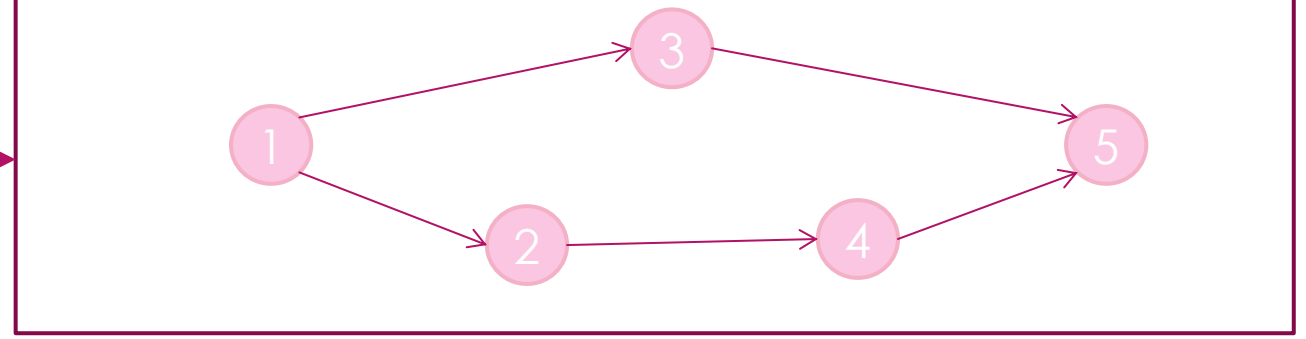
Lab3.B Flow Control

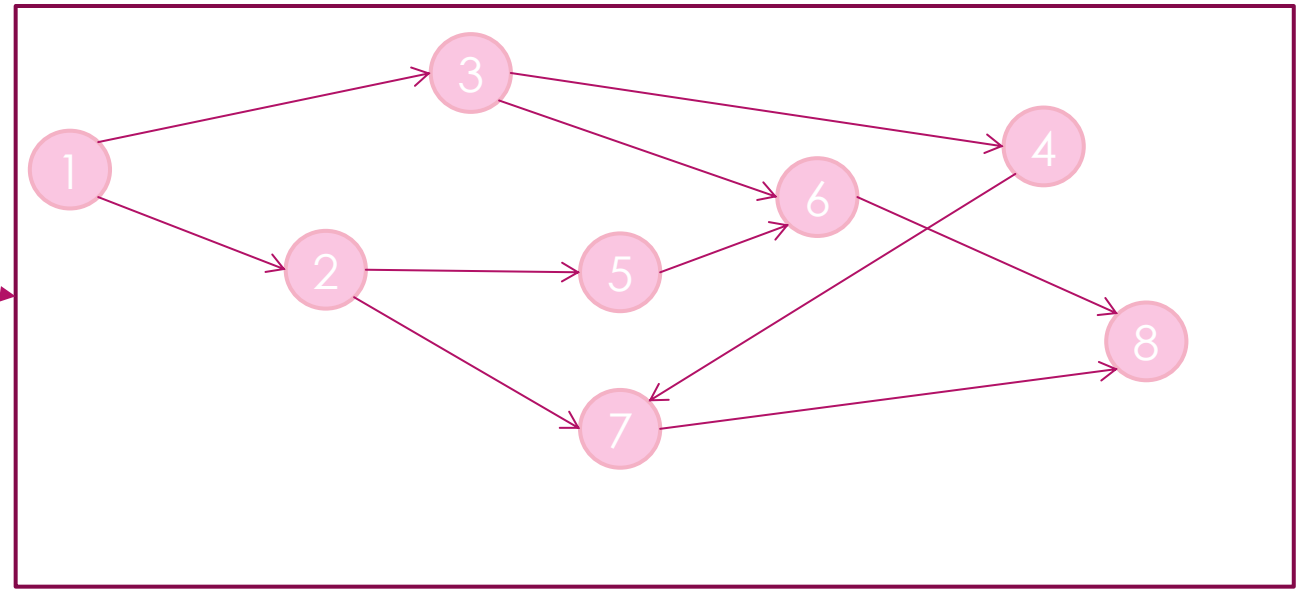
- There are n scenic spots in the scenic area, with the entrance as 1 and the exit as n. The roads between the scenic spots are directed edges, and the roads in the entire scenic area form a directed acyclic graph (DAG). In order to control the flow of each vertex in the scenic area, the manager hopes to assign a flow value w_i to each point, so that the sum of the flow on all paths from 1 to n is equal. Can you help him solve this problem?

Input:

2 test cases

```
2
5 5
1 2
1 3
2 4
3 5
4 5
8 10
1 2
1 3
2 5
2 7
3 4
3 6
5 6
4 7
6 8
7 8
```

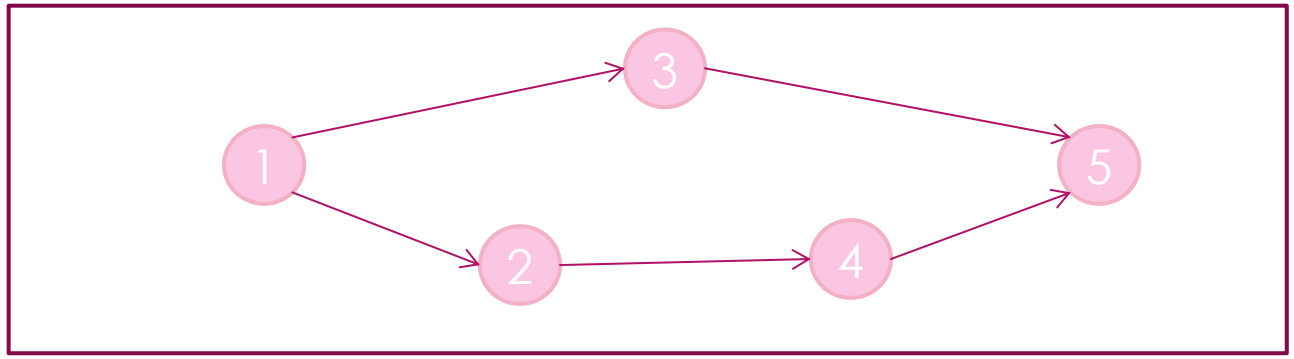




Output:

```
Yes
1 1 2 1 3
No
```

test case 1：



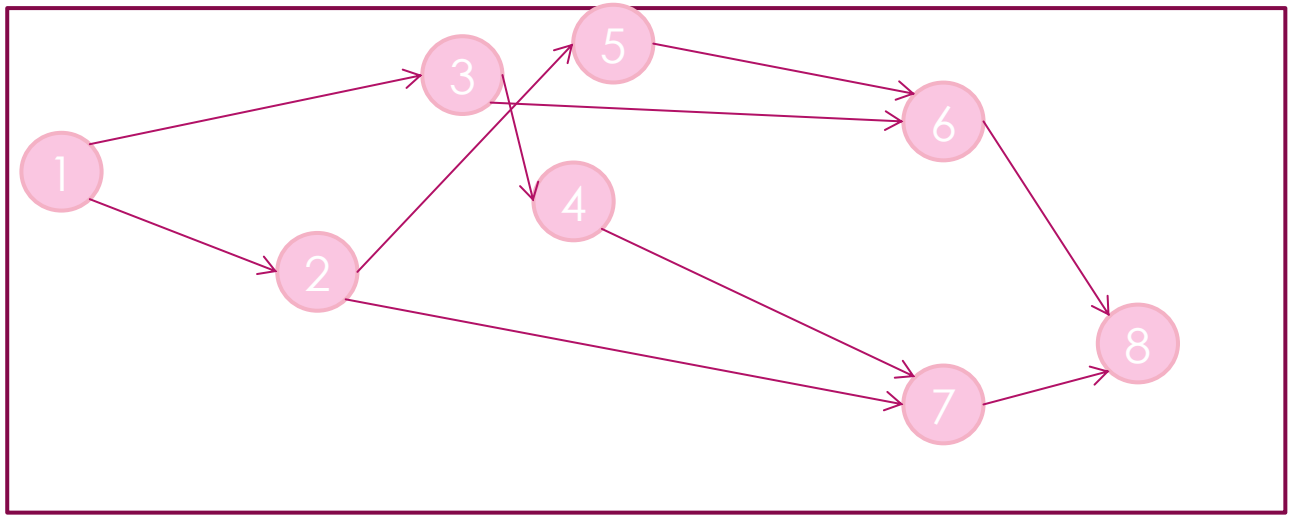
all paths from 1 to 5:
1 → 3 →5
1→ 2→4→5

$w_1 + w_3 + w_5 = w_1 + w_2 + w_4 + w_5$
let $w_3 = w_2 + w_4$

There are multiple solutions.
1 1 2 1 3
1 2 5 3 4
8 8 10 2 8
…

You can print any of them.

test case 2:



all paths from 1 to 8:

1 → 3 →4→7→8

1 → 3 →6→8

1 → 2 →5→6→8

1 → 2 →7→8

$w_1 + w_3 + w_4 + w_7 + w_8 = w_1 + w_3 + w_6 + w_8 = w_1 + w_2 + w_5 + w_6 + w_8 = w_1 + w_2 + w_7 + w_8$

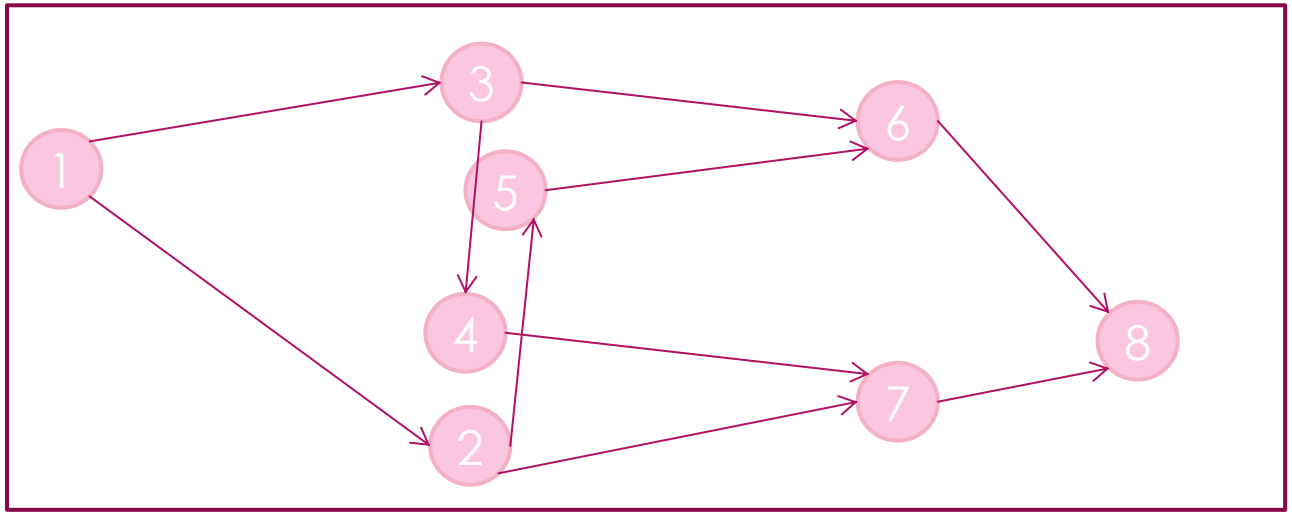
$w_3 + w_4 = w_2$

$w_3 = w_2 - w_4$

$w_3 = w_2 + w_5$

All w_i are positive integers

no w_3 can satisfy the above equations



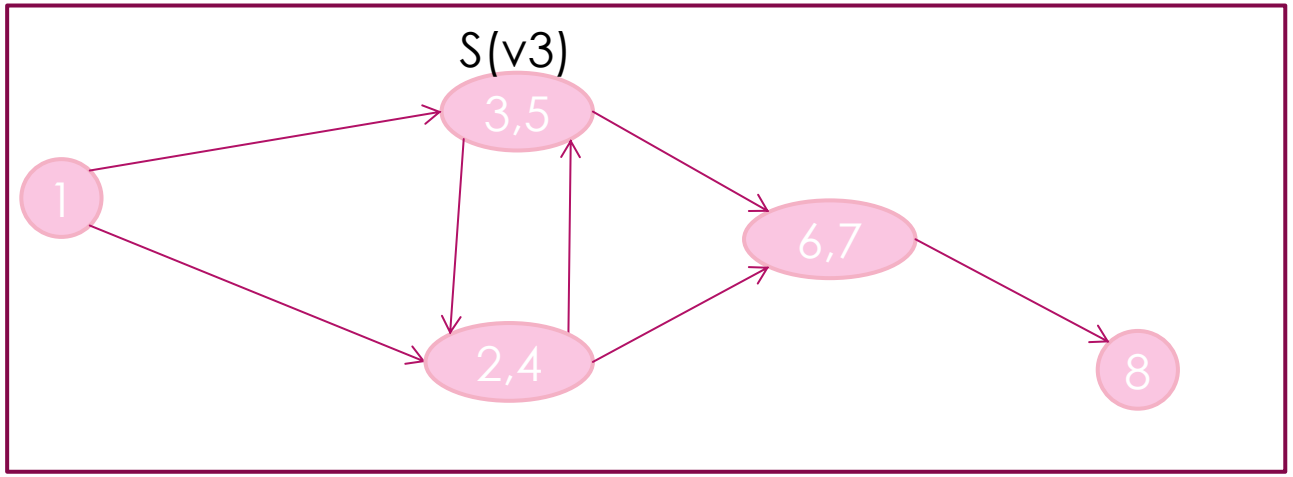
$w_1 + w_3 + w_4 + w_7$
$w_1 + w_2 + w_7$
$w_1 + w_3 + w_6$
$w_1 + w_2 + w_5 + w_6$

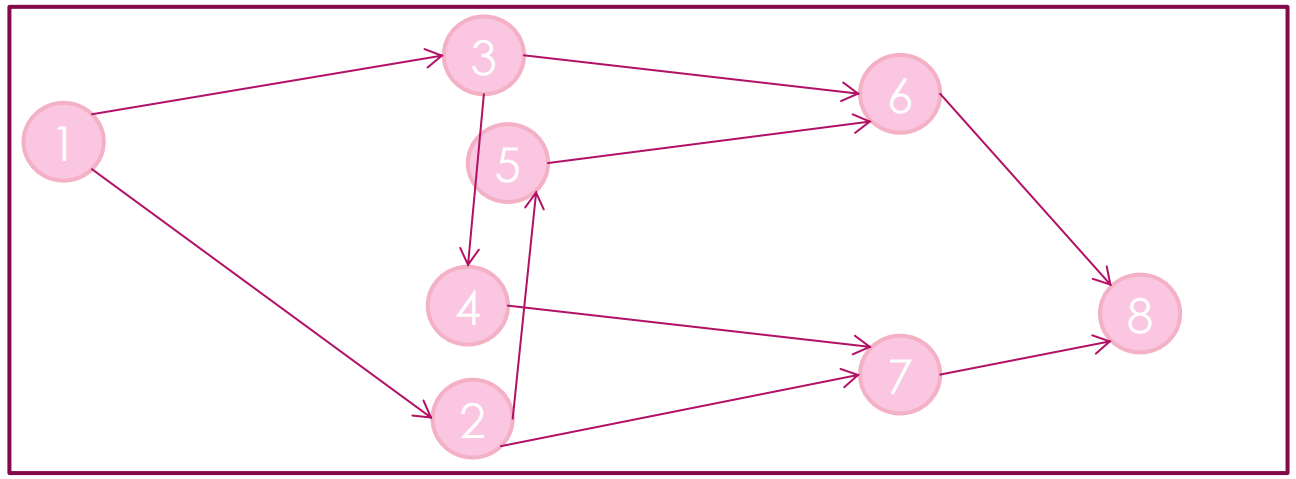
$w_1 + w_3$
$w_1 + w_2 + w_5$

$w_1 + w_3 + w_4$
$w_1 + w_2$

let S(vi) = sum of paths from 1 to vi
S(v6) = S(v7)
S(v3) = S(v5)
S(v2) = S(v4)

if vk has multiple prev. points, all S values of prev. points should be equal, and merge to one point.





original graph

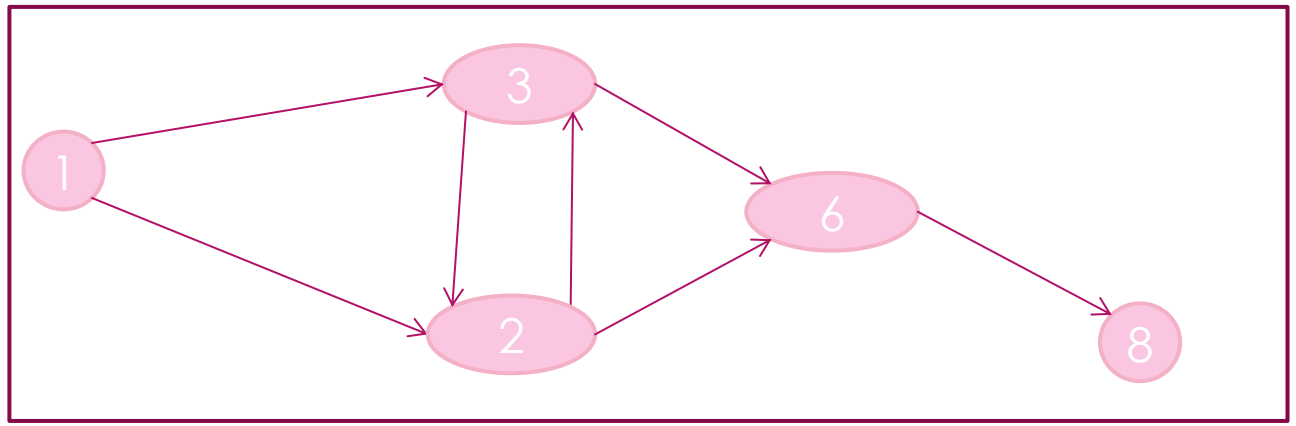
original point:

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

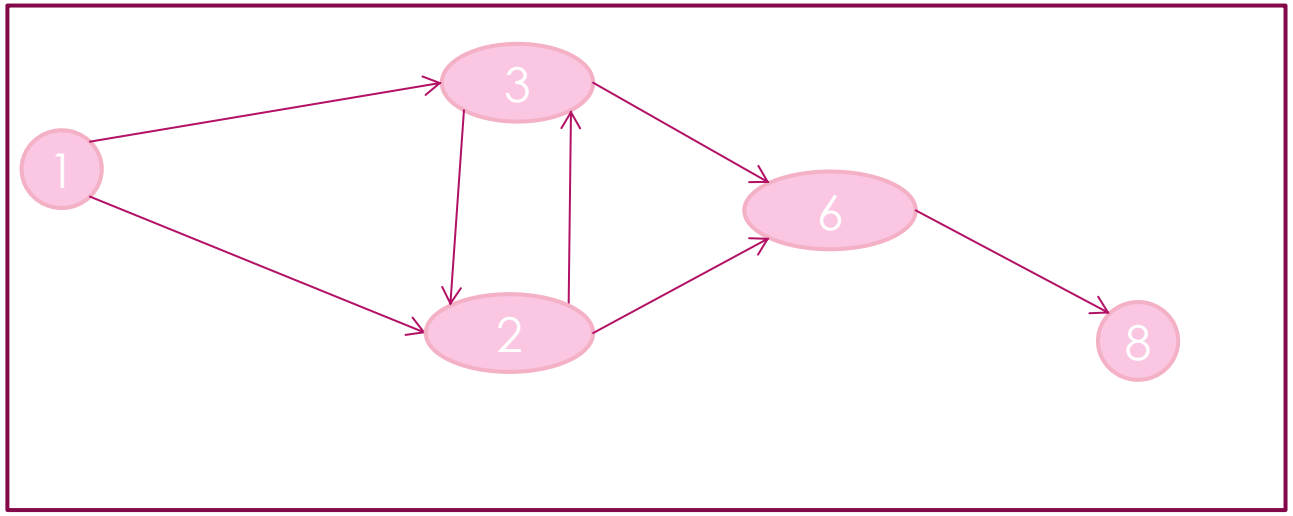
if a point's indegree>=2:
merge its prev points to one point

merged point:

1	2	3	4	5	6	7	8
1	2	3	2	3	6	6	8



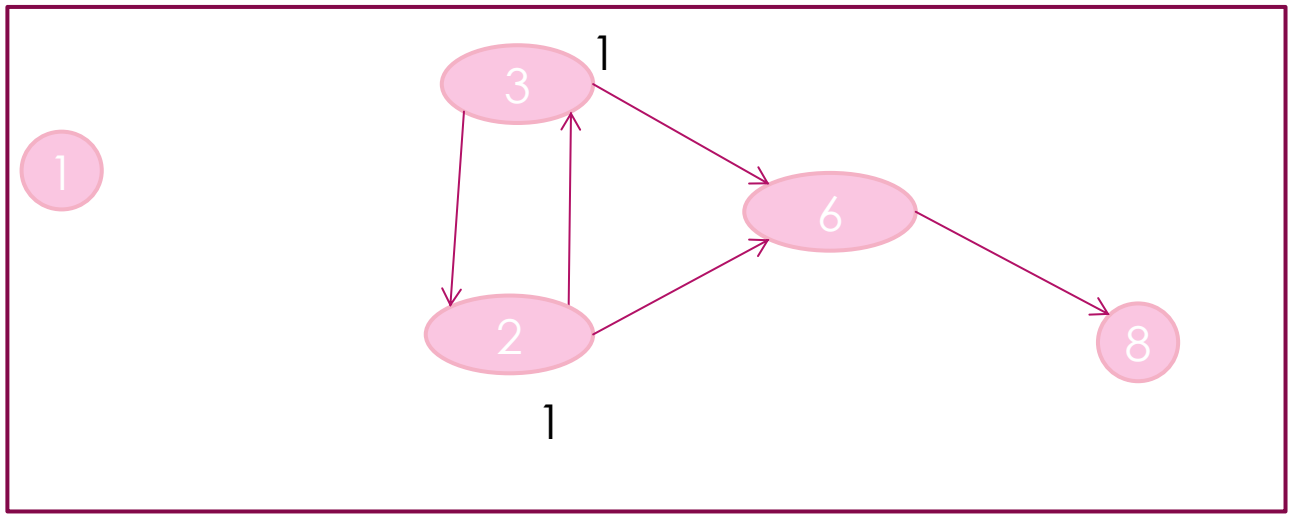
new graph



if the above graph has a circle
output “No”
else output “Yes”

Topological Sorting Algorithm

In the process of performing a topological sort, you can calculate the farthest distance from each point to the starting point



start from point 1, the distance of point 1 ← 0

initial：

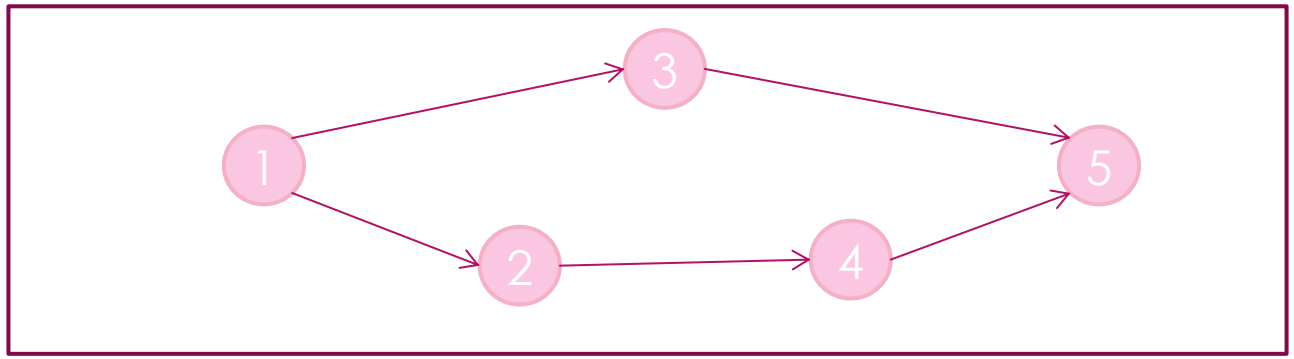
	1	2	3	6	8
distance:	0	0	0	0	0

1 has children 2 and 3, update their distance to 0+1

	1	2	3	6	8
distance:	0	1	1	0	0

No point with indegree == 0, stop

the graph has at lease 1 circle, output “No”.



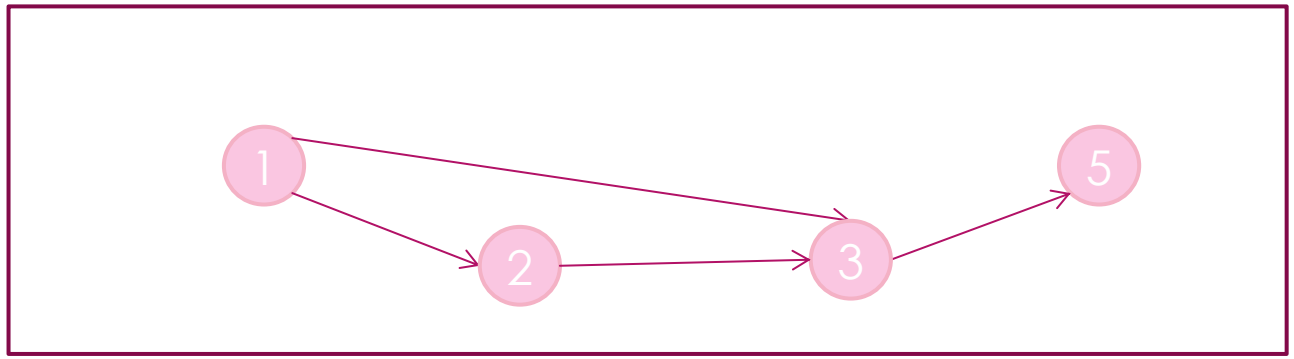
original graph

original point:

1	2	3	4	5
1	2	3	4	5

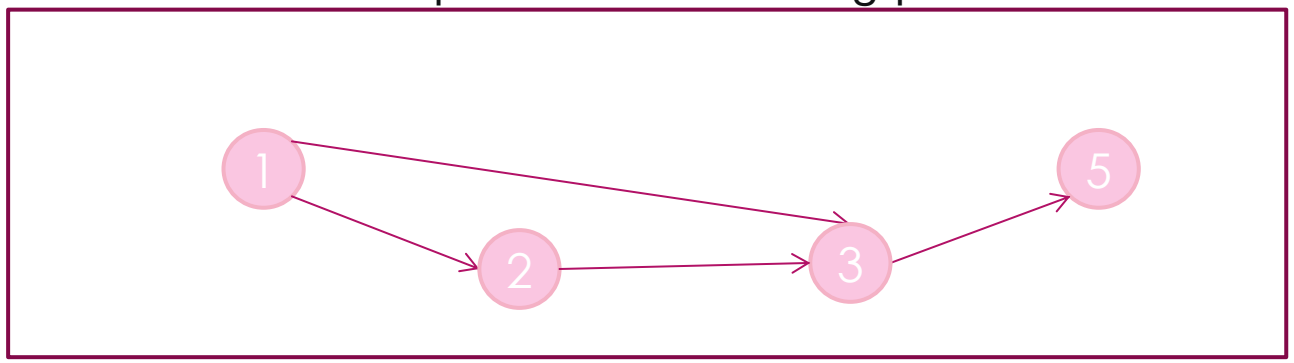
merged point:

1	2	3	4	5
1	2	3	3	5



new graph

In the process of performing a topological sort, you can calculate the farthest distance from each point to the starting point



new graph

start from point 1, the distance of point 1 ← 0

1	2	3	5
0	0	0	0

1 has children 2 and 3, update their distance to 0+1

1	2	3	5
0	1	1	0

2 has a child 3, 2>1 update 3's distance to 2

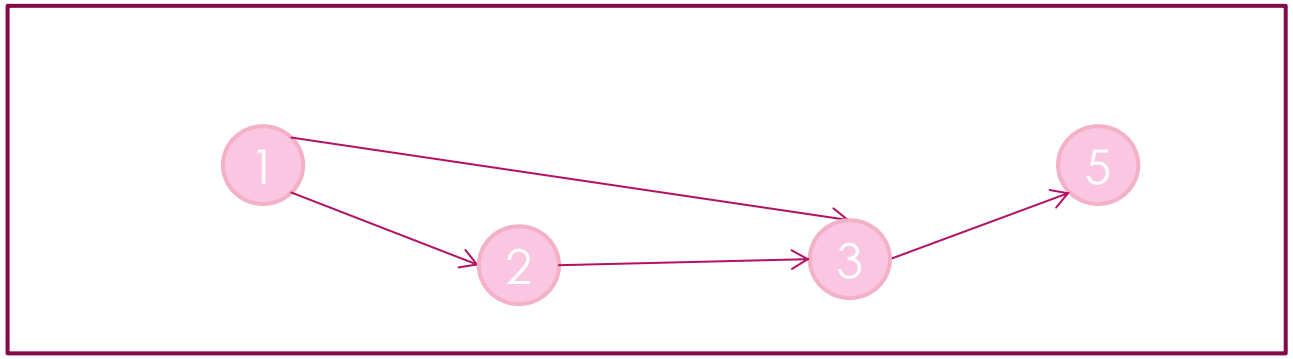
1	2	3	5
0	1	2	0

3 has a child 5, 3>0 update 5's distance to 3

distance:

1	2	3	5
0	1	2	3

new graph



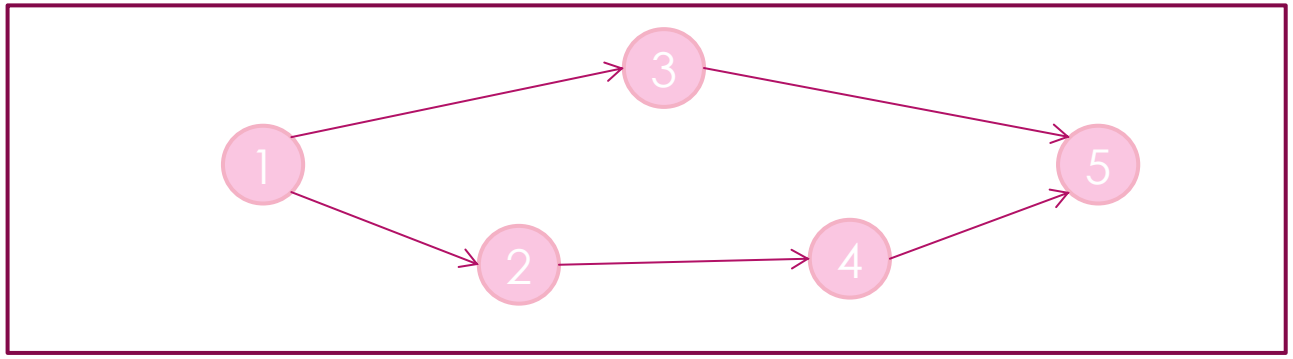
distance:

1	2	3	5
0	1	2	3

merged point:

1	2	3	4	5
1	2	3	3	5

original graph




```
w[1] = 1
w[2] = dis[2]-dis[1]=1
w[3] = dis[3]-dis[1]=2
w[4] = dis[3]-dis[2]=1
w[5] = dis[5]-dis[3]=1
```

original point　　original ->new　　original prev->new