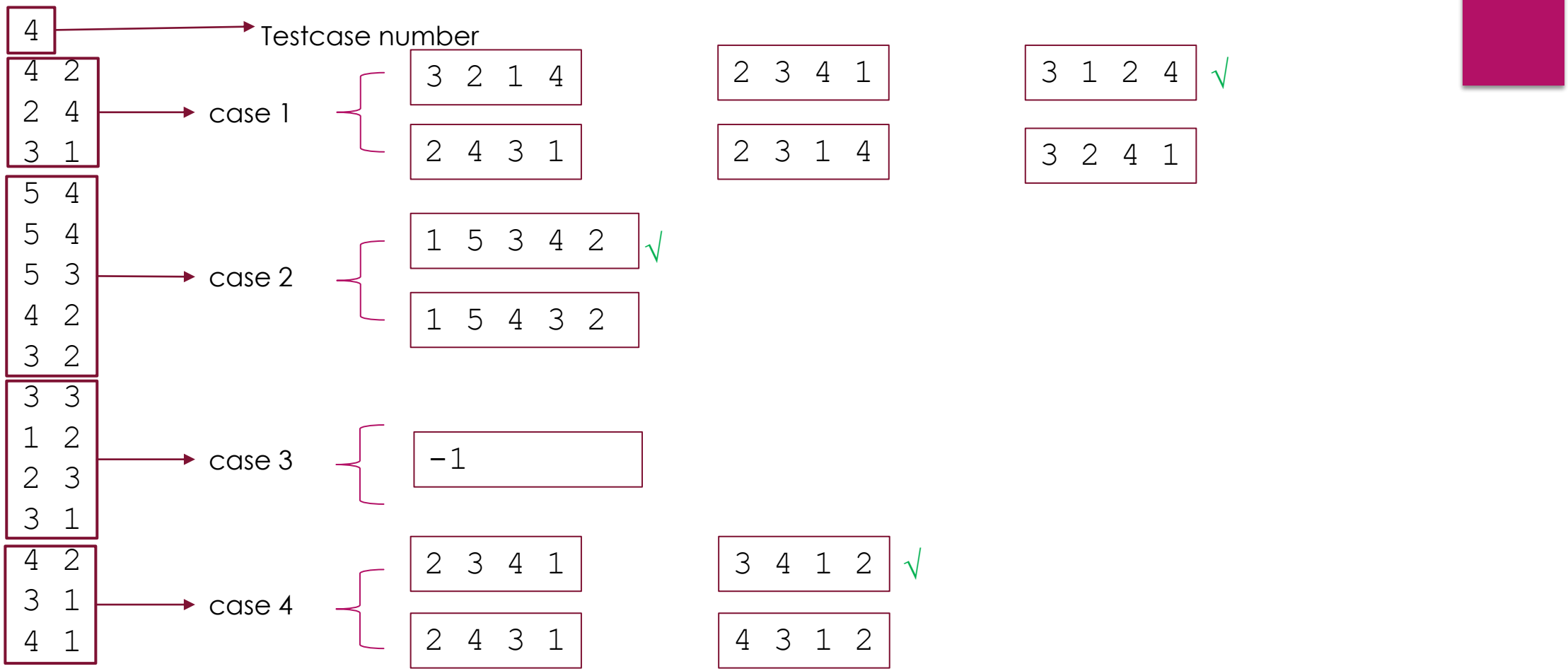
Lab 3

YAO ZHAO

Lab3.A: Sequence

- Given a sequence with n integers $1, 2, \ldots, n$. And m constraints (x_i, y_i) denotes that x_i should be in front of y_i in the sequence.
- The task is to construct the sequence. In the sequence, you should place 1 at the front of the sequence as much as possible, and after placing 1, place 2 at the front of the sequence as much as possible, and so on. Also, the sequence should be restricted by the constraints.
- You should output what the sequence is, or just output -1 denoting that it is impossible to find a such sequence.

Sample Input：

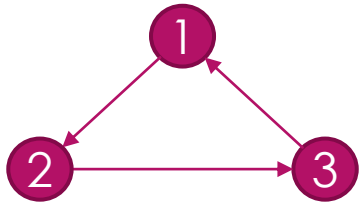


Lab 3.A Analysis

- If constraints form a cycle, the conditions cannot be met

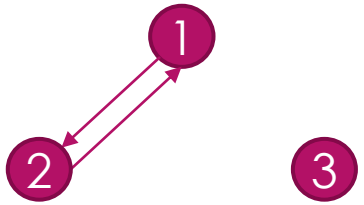
3	3
1	2
2	3
3	1

→ case 3



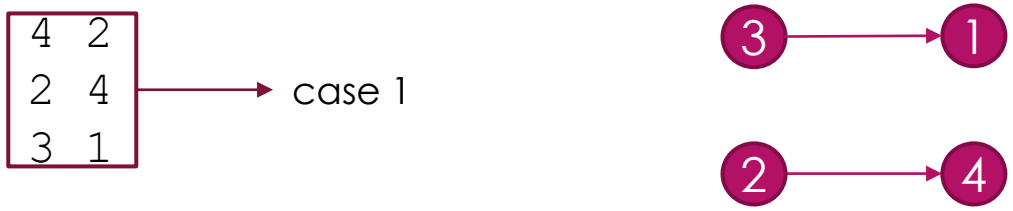
or

3	2
1	2
2	1



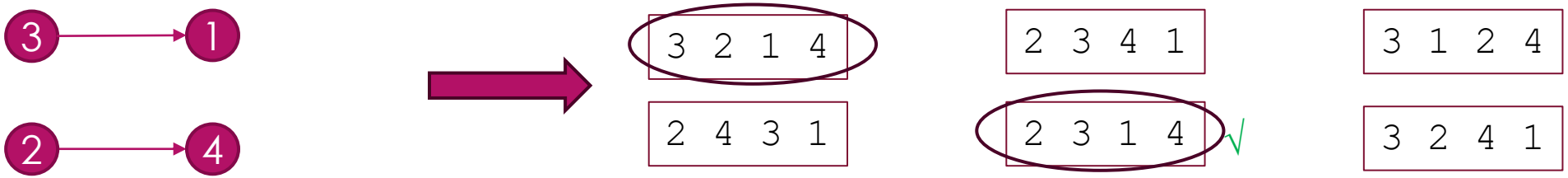
Lab 3.A Analysis

- Basic topological sorting may not get the desired order



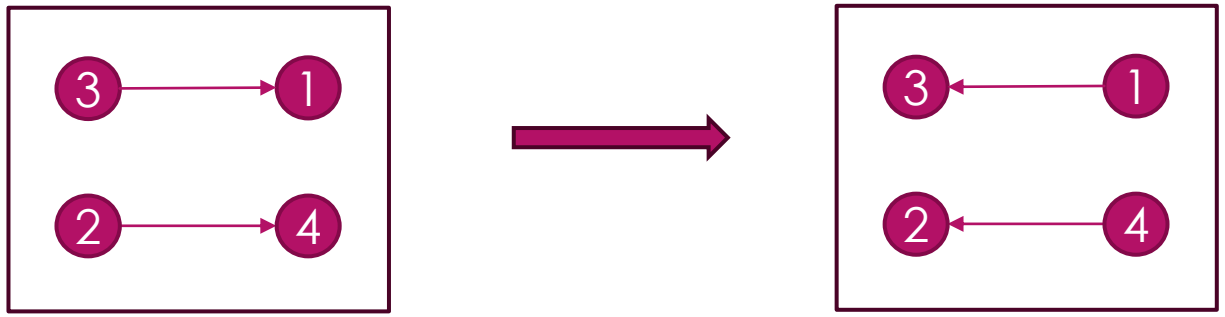
In the scenario, it is indeterminate whether to start the topological sorting process **from 3 or from 2**.

If we always choose the **least lexicographical among all the nodes which indegree is 0, we can get the least lexicographical order**. We can observe that in the process, we will place **the larger number at the end of the sequence as much as possible.**



Lab 3.A Solution

- Build a reverse graph and perform lexicographically largest topological sort, then reverse the order.



Original graph

Reverse graph

lexicographically largest topological sort

4	2	1	3

Reverse the sequence

3	1	2	4

Lab 3.A Proof

- Define "minimum sequence" as the sequence where smaller numbers are placed as forward as possible.
- Objective: Prove that in a Directed Acyclic Graph (DAG), the topological order with the smallest "minimum sequence" corresponds to the reverse largest lexicographical order.

- Base case: When there is only one node, the statement is trivially true.

Inductive Hypothesis

- Assume the statement holds for DAGs with $| V | < n$ nodes, where V is the set of nodes in the graph.

Proof for $|V| = n$

- Let x be the node with the smallest index in $G=(V,E)$.
- Divide the nodes into two sets:
  - $S = \{v \mid there\ exists\ a\ path\ v \to x\}$
  - $T = V - S$

- Here $x \in S$ and set $m = |S|$.
- The node x must be at the $m - th$ position in the topological order of S.
- Split the graph into three parts: $S - \{x\}, \{x\}, T$.

- In the subgraph $S - \{x\}$ and T , $|S - \{x\}| < n$, $|T| < n$, the hypothesis holds.
- x is the node with the smallest index, and moving to $(m + i)\ -\ th\ (i > 0)$position does not yield a larger reverse lexicographical order. So for the graph $|V| = n$, the hypothesis holds.
- By induction, the conclusion holds for the entire graph G.

Lab3.B: Path

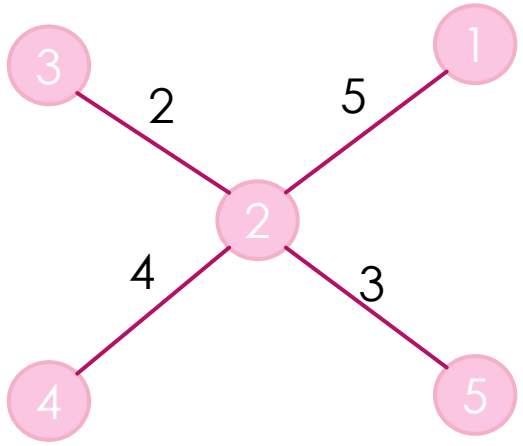
- Given an n –nodes tree where edges have weights. Nodes are numbered starting from 1 to n.
- The task is to find a simple path between two nodes whose total edge weight is less than or equal to a given value s. You can also select the same two nodes, , and this case means the total edge weight is 0. The path should be such that the maximum distance from any point not on the path to the path itself is minimized.
- To simplify the task, you only need to output the maximal distance which is the minimal answer among all the answers.

Sample Input:

5	0	
1	2	5
2	3	2
2	4	4
2	5	3

s value

Node number



In this case, *s* is 0, if we want to make the weight of the path <=0, only select 2 same nodes.

Select the node 1 → Max(dis(1,2),dis(1,3),dis(1,4),dis(1,5)) = 9

Select the node 2 → Max(dis(2,1),dis(2,3),dis(2,4),dis(2,5)) = **5**

Select the node 3 → Max(dis(3,1),dis(3,2),dis(3,4),dis(3,5)) = 7

Select the node 4 → Max(dis(4,1),dis(4,2),dis(4,3),dis(4,5)) = 9

Select the node 5 → Max(dis(5,1),dis(5,2),dis(5,3),dis(5,4)) = 8

The minimum value

Output:

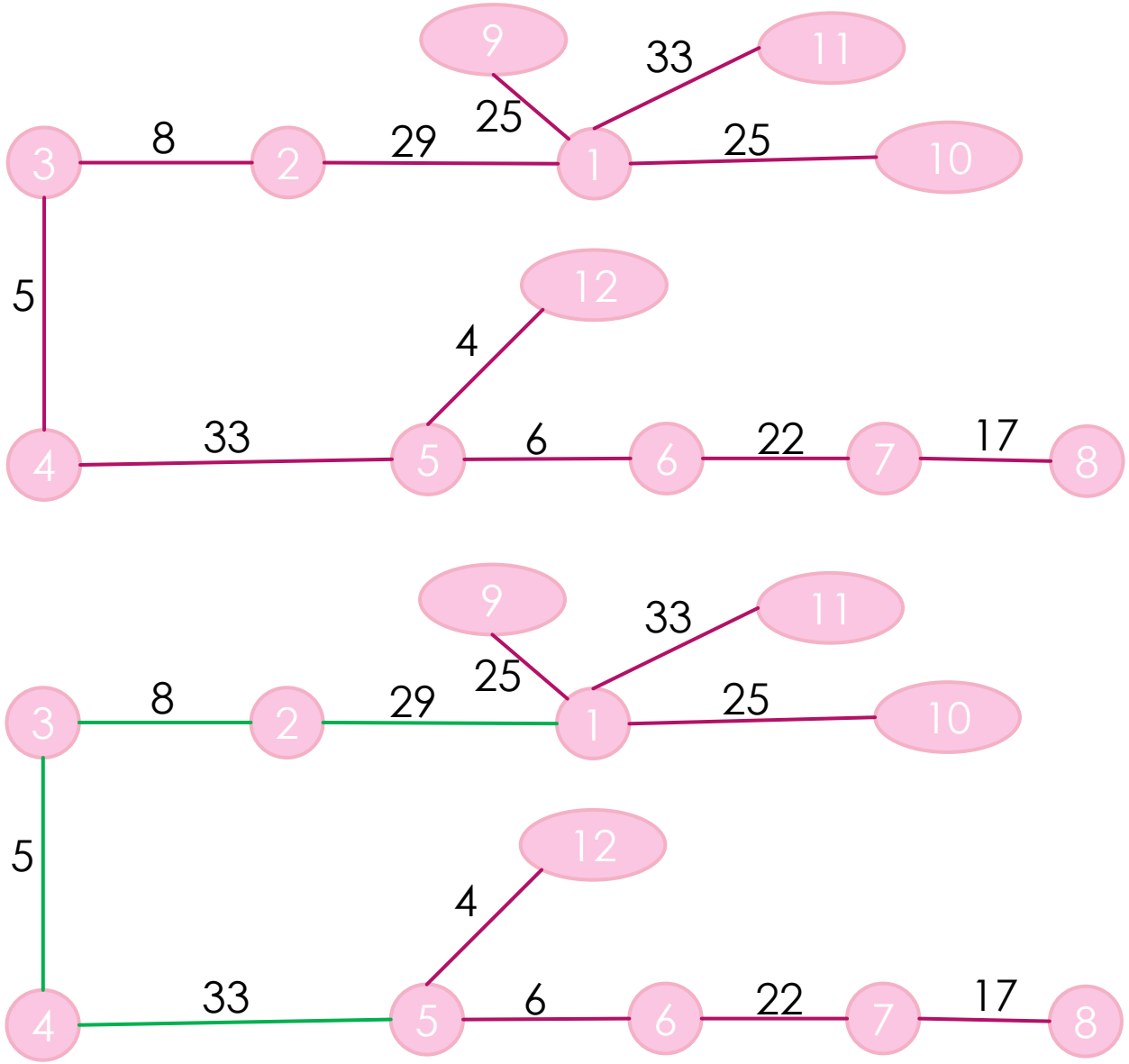
5

Sample Input:

```
12  77
1  2  29
1  10  25
4  5  33
5  6  6
5  12  4
1  11  33
2  3  8
3  4  5
1  9  25
6  7  22
7  8  17
```

s value

Node number



Hint: In this case, the path is : **5-4-3-2-1**

The diameter of a tree

In the book "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (commonly known as CLRS), the diameter of a tree is defined as the length of the longest path between any two nodes in the tree.

CLRS 22.2-8:

22.2-8 ★

The *diameter* of a tree $T = (V, E)$ is defined as $\max_{u,v \in V} \delta(u, v)$, that is, the largest of all shortest-path distances in the tree. Give an efficient algorithm to compute the diameter of a tree, and analyze the running time of your algorithm.

Lab 3.B Analysis

Without considering the threshold s, the tree's diameter represents the desired path.

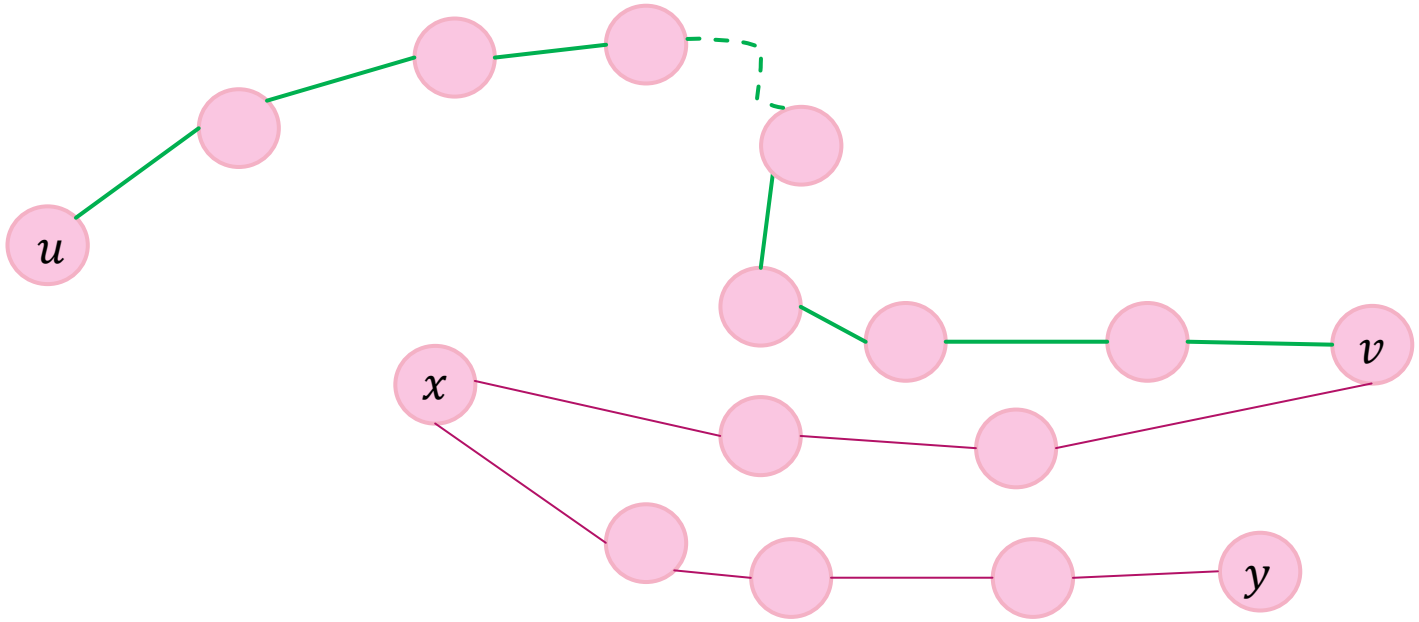
Property of the diameter of a tree

For any point x on a tree, the endpoints of the diameter are the farthest points from x

Proof:
Assume there exists a point y from x farther than the diameter endpoints u and v.
The path (y, x) plus the path from x to the other diameter endpoint will be longer than the diameter, leading to a contradiction of the definition of the tree's diameter.

Lab 3.B Analysis

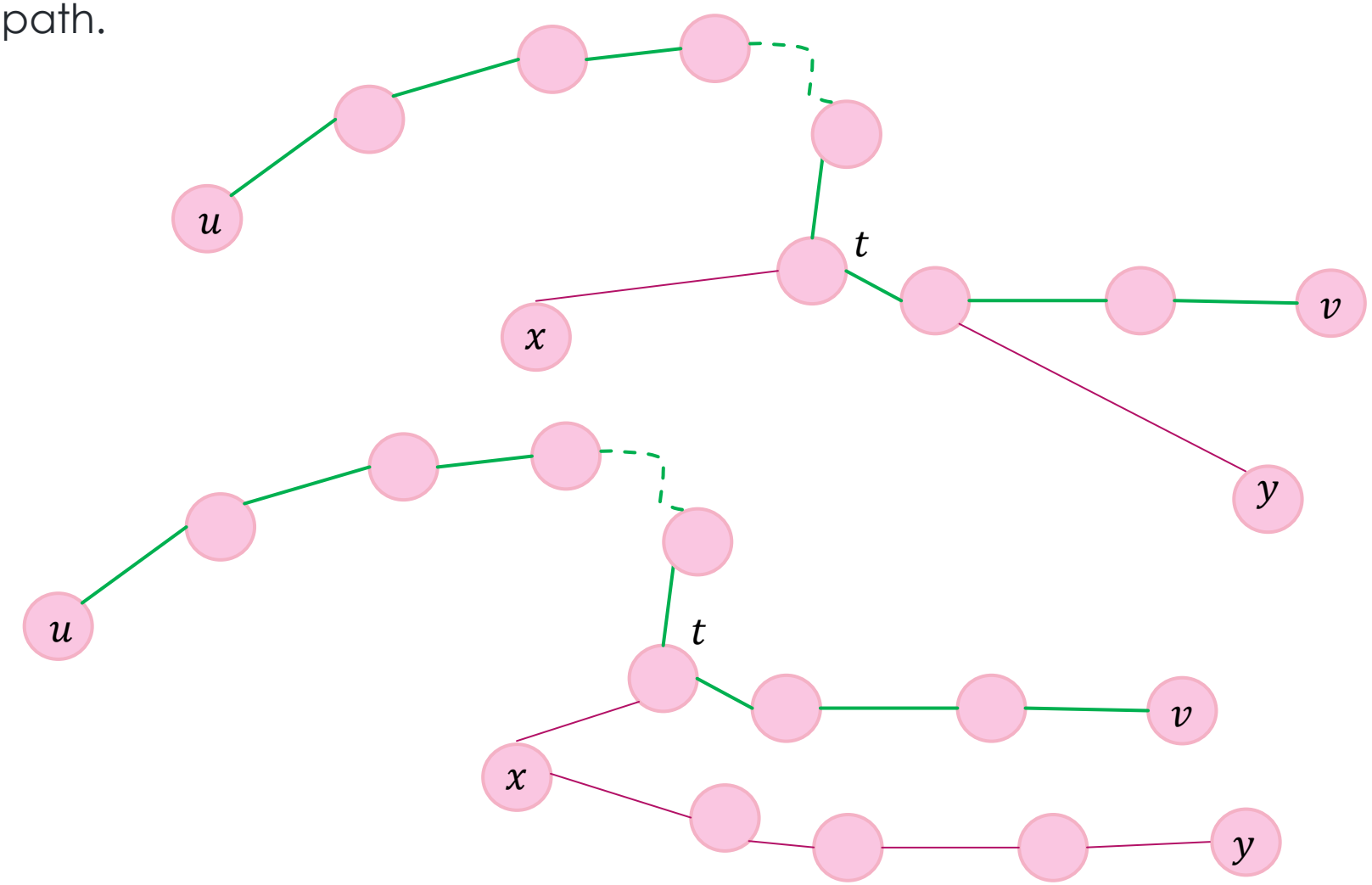
Case 1 : $dis(x, y) > dis(x, v)$ and Path(x, v) don't pass through any other nodes on the diameter path.



$dis(u, y) > dis(u, v)$
contradiction

Lab 3.B Analysis

Case 2 : $dis(x,y) > dis(x,v)$, Path(x,v) pass through some other nodes t on the diameter path.



$dis(x,y) > dis(x,v)$

$dis(t,y) > dis(t,v)$

$dis(u,y) > dis(u,v)$

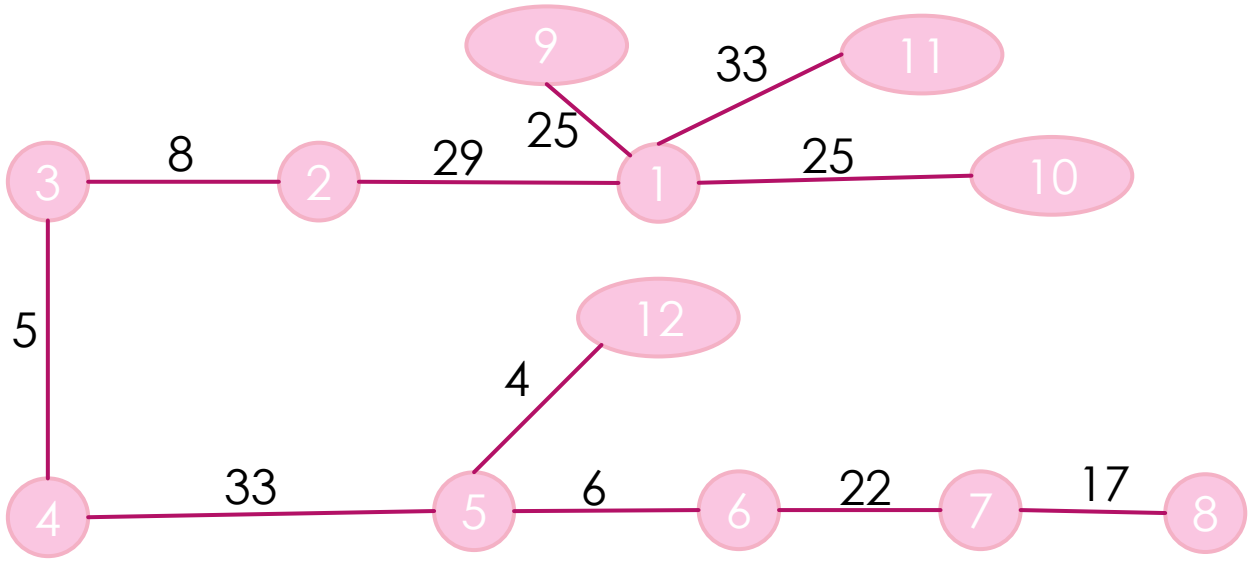
contradiction

Lab 3.B Solution

Step 1: Use DFS or BFS to find the points on the diameter path.

(1) Start from any points x and employ BFS or DFS find the point u with the longest distance from x

(2) Start from u, using BFS or DFS to find the point v with the longest distance from u. Record each point on the path from u to v and its respective distance from u.



(1) Assuming a search from node 1, we can determine that node 8 is the farthest node from 1
(2) Subsequently, conducting a search from node 8 reveals that node 11 is the farthest node from 8. Record distances in the following list.

node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153

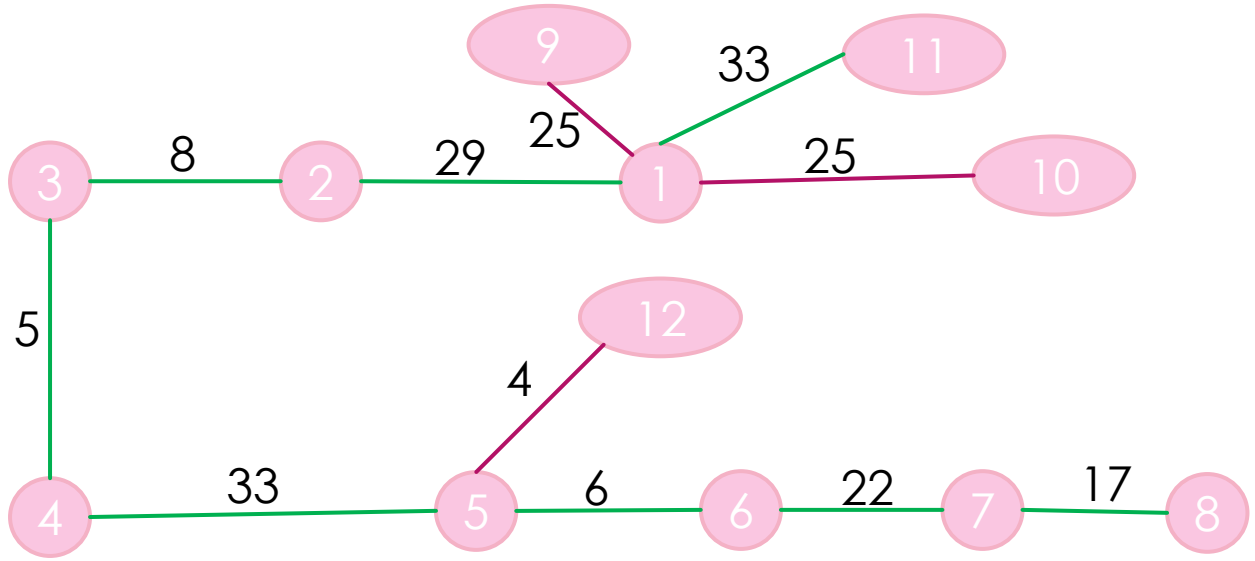
Lab 3.B Solution

Step 2: calculate the maximum distance from any point not on the diameter path to the diameter path.

For each node on the diameter path:

(1) Start a DFS or BFS from that node to find the farthest node and record the distance as **branch distance**.

(2) While exploring, ensure that you do not pass through any other nodes on the diameter path.



the maximum distance from any point not on the diameter path to the diameter path.

Branch distance list:

node	8	7	6	5	4	3	2	1	11
Branch dis	0	0	0	4	0	0	0	25	0

Lab 3.B Solution

Step 3: Considering the threshold s, if the total edge weight of the tree's diameter <= s, just output the result calculate through the step 2, else, we should find a subinterval on the diameter of the tree.

We can employ the **Two Pointers Algorithm** to find the optimal subinterval on the diameter of a tree that satisfies certain constraints.

When applying the **Two Pointers Algorithm** to enumerate intervals, follow these steps:

1.Initialize two pointers, a start pointer and an end pointer, both pointing to the beginning of the interval.

2.Adjust the pointers based on the problem's constraints:

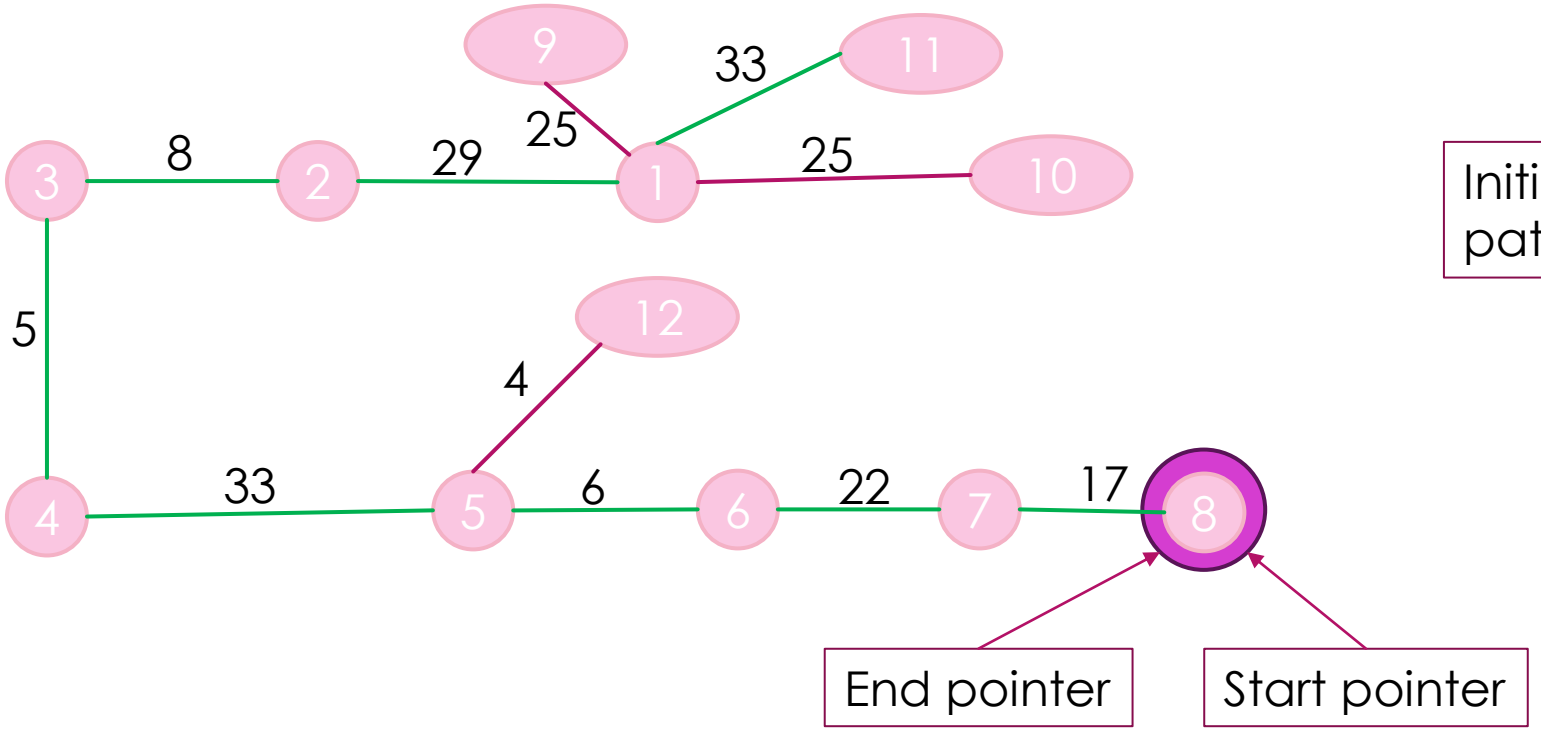
- If the current interval meets the condition (total weight <= s), expand the interval (move the end pointer).
- If the current interval does not meet the condition (total weight > s), shrink the interval (move the start pointer).

3.While moving the pointers, update the result by obtaining the minimum.

4.Keep moving the pointers until all possible intervals have been explored.

Lab 3.B Step 3 Demo

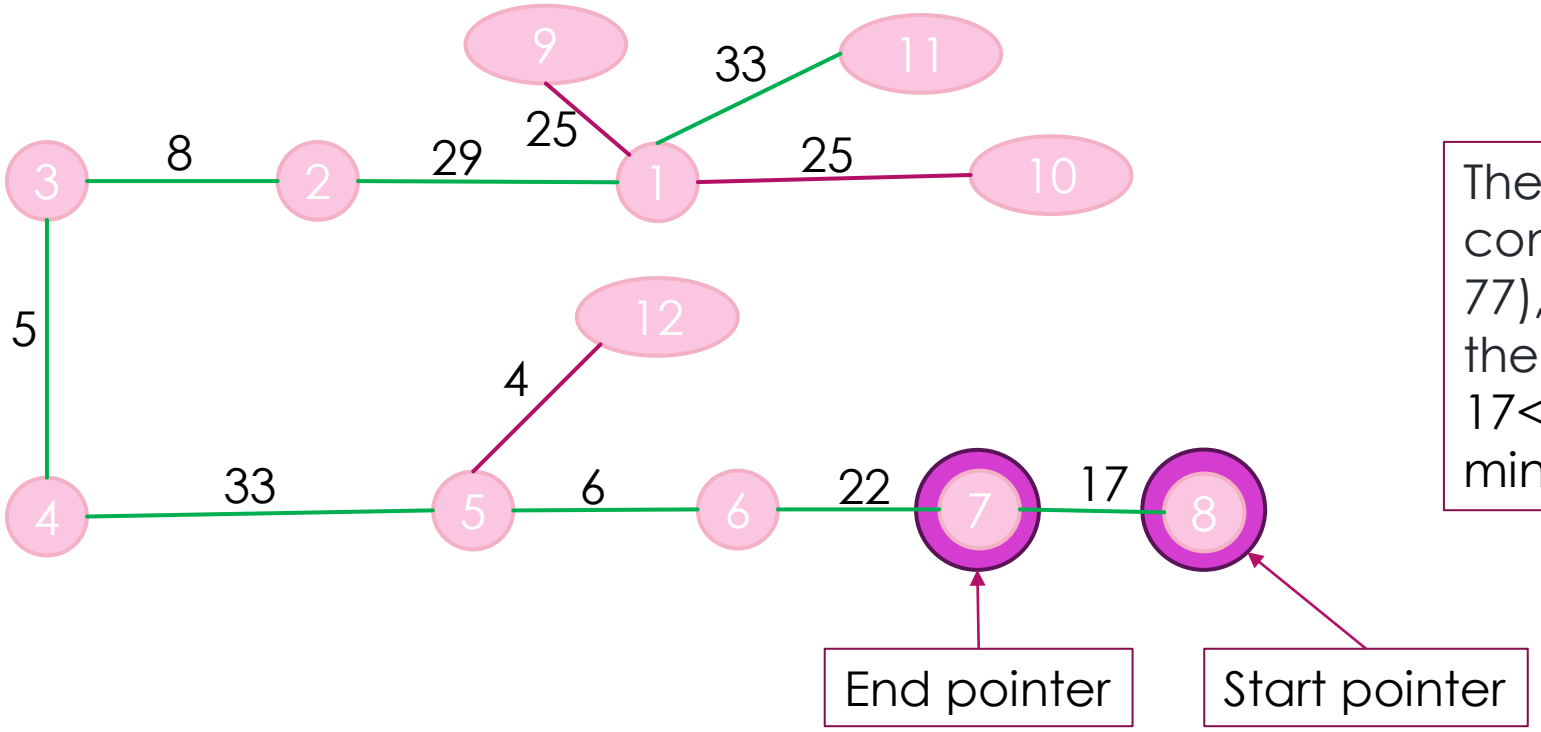
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



Initial state, only choose 8 as path, the result is max(0,153)=153

Lab 3.B Step 3 Demo

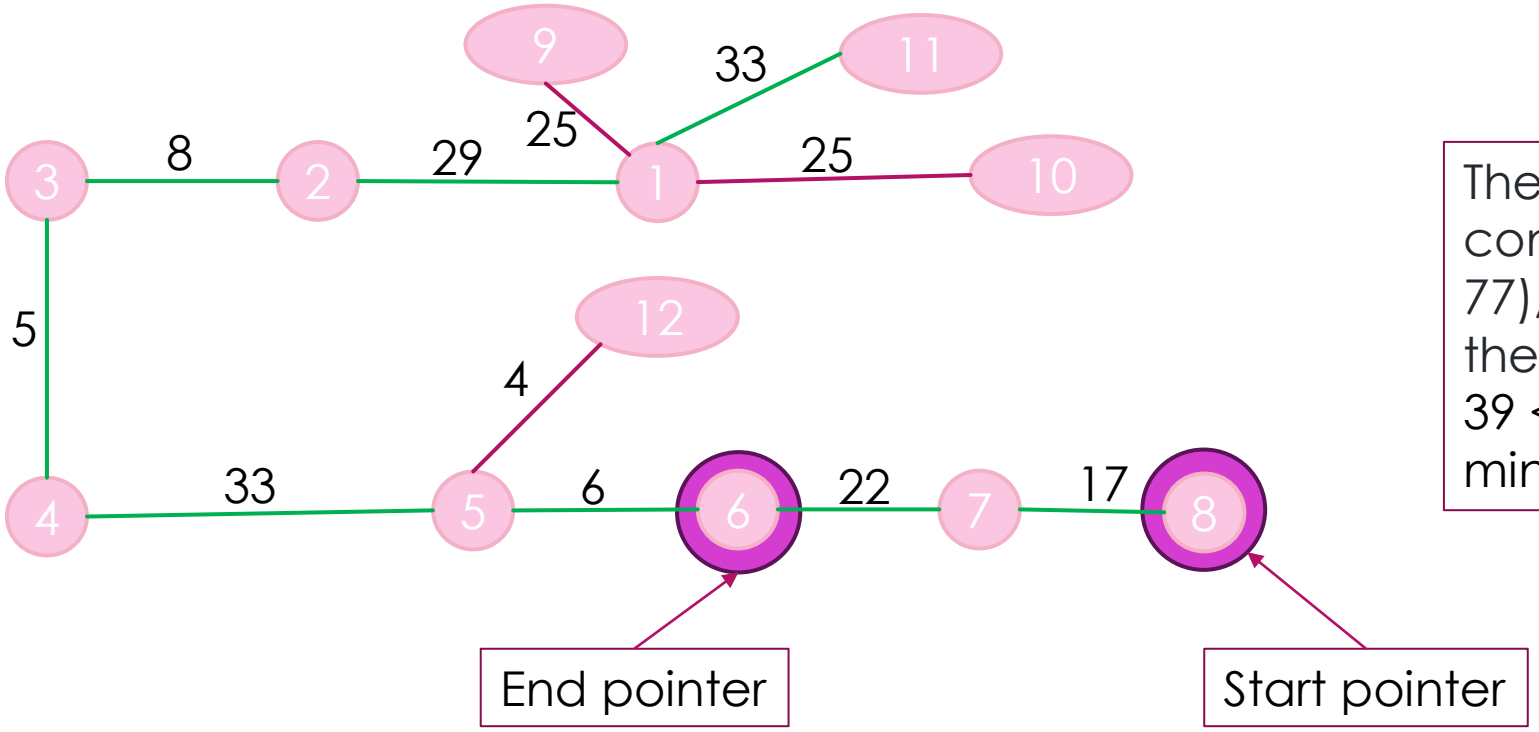
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 0 <= s = 77), expand the interval (move the end pointer), total weight = 17<=77, the result is min(153,max(0,153-17))=136

Lab 3.B Step 3 Demo

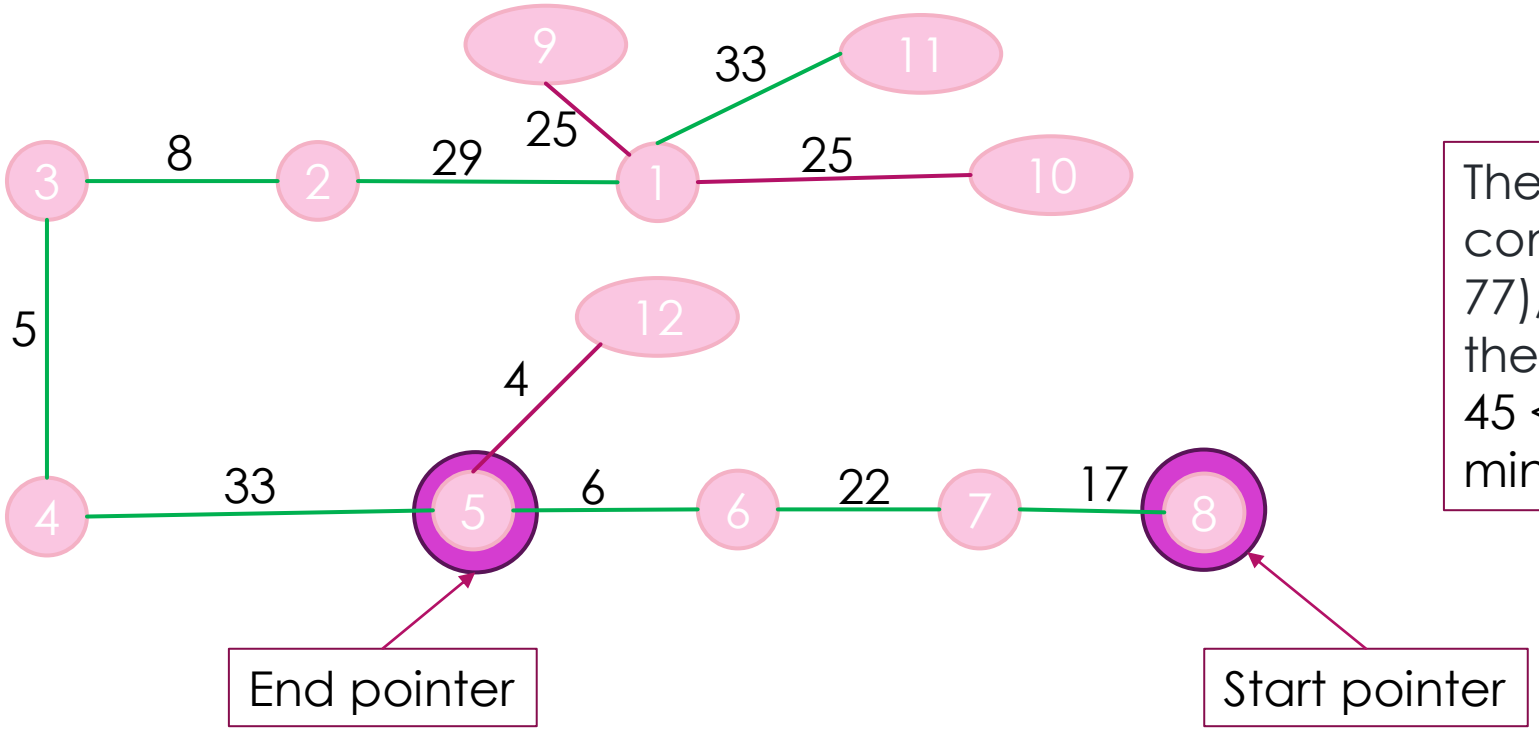
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 17 <= s = 77), expand the interval (move the end pointer), total weight = 39 <=77, the result is min(136,max(0,153-39))=114

Lab 3.B Step 3 Demo

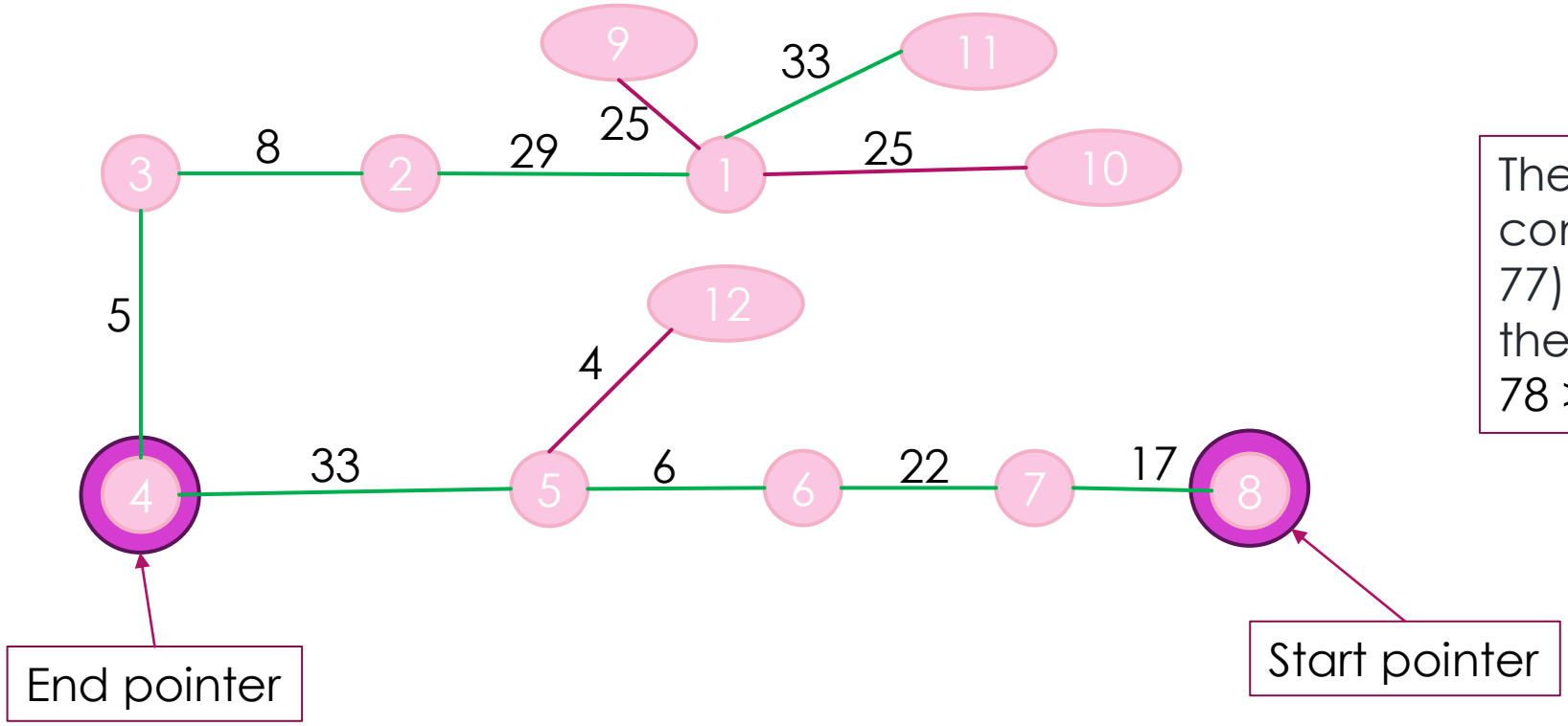
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 39 <= s = 77), expand the interval (move the end pointer), total weight = 45 <=77, the result is min(114,max(0,153-45))=108

Lab 3.B Step 3 Demo

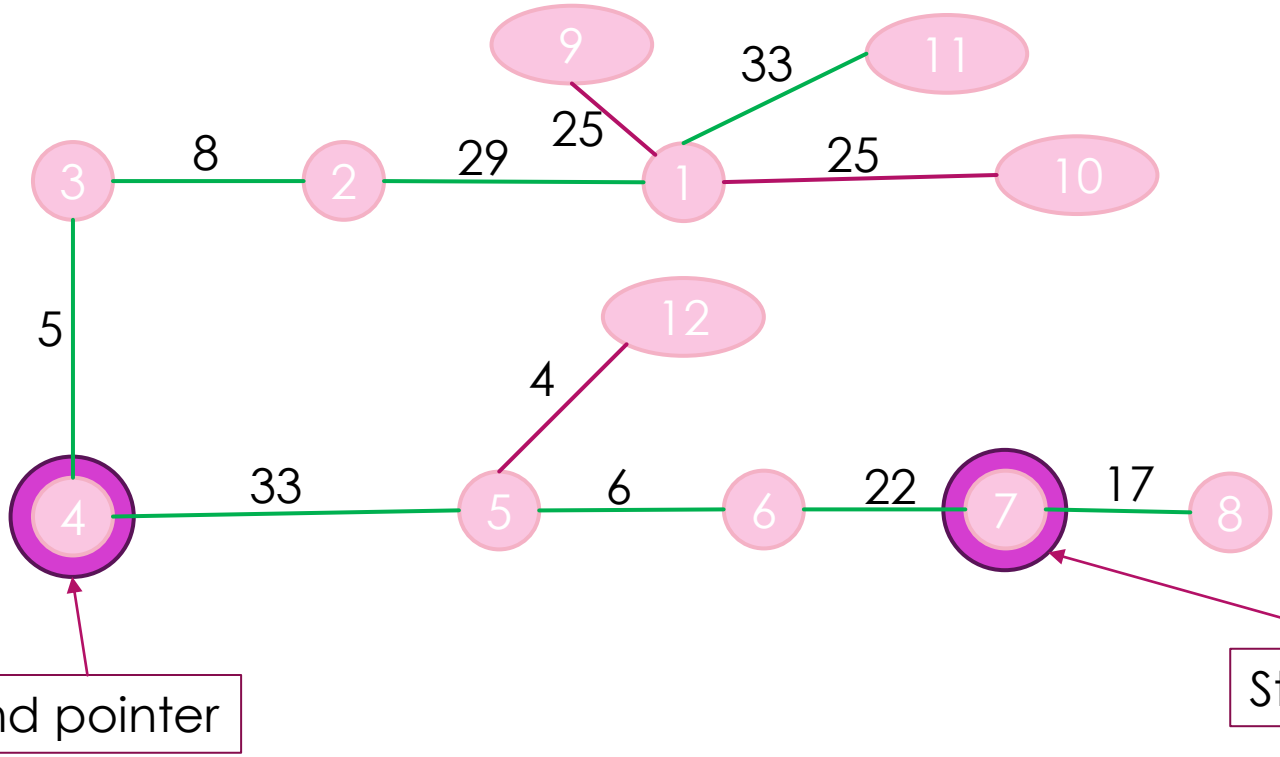
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 45 <= s = 77), expand the interval (move the end pointer), total weight = 78 >77.

Lab 3.B Step 3 Demo

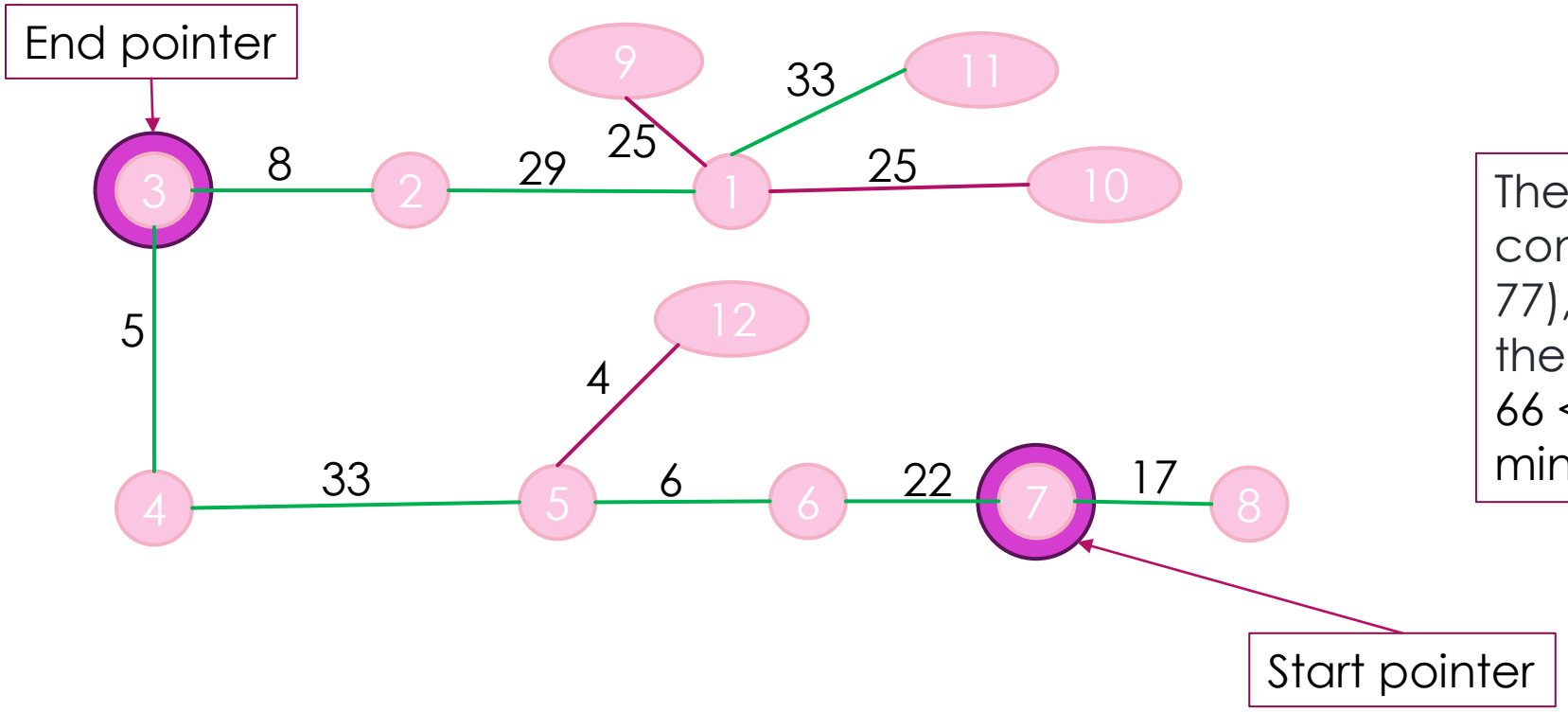
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval does not meet the condition (total weight 78 >77), shrink the interval (move the start pointer), total weight = 61 <= 77, the result is min(108,max(17,153-78))=75

Lab 3.B Step 3 Demo

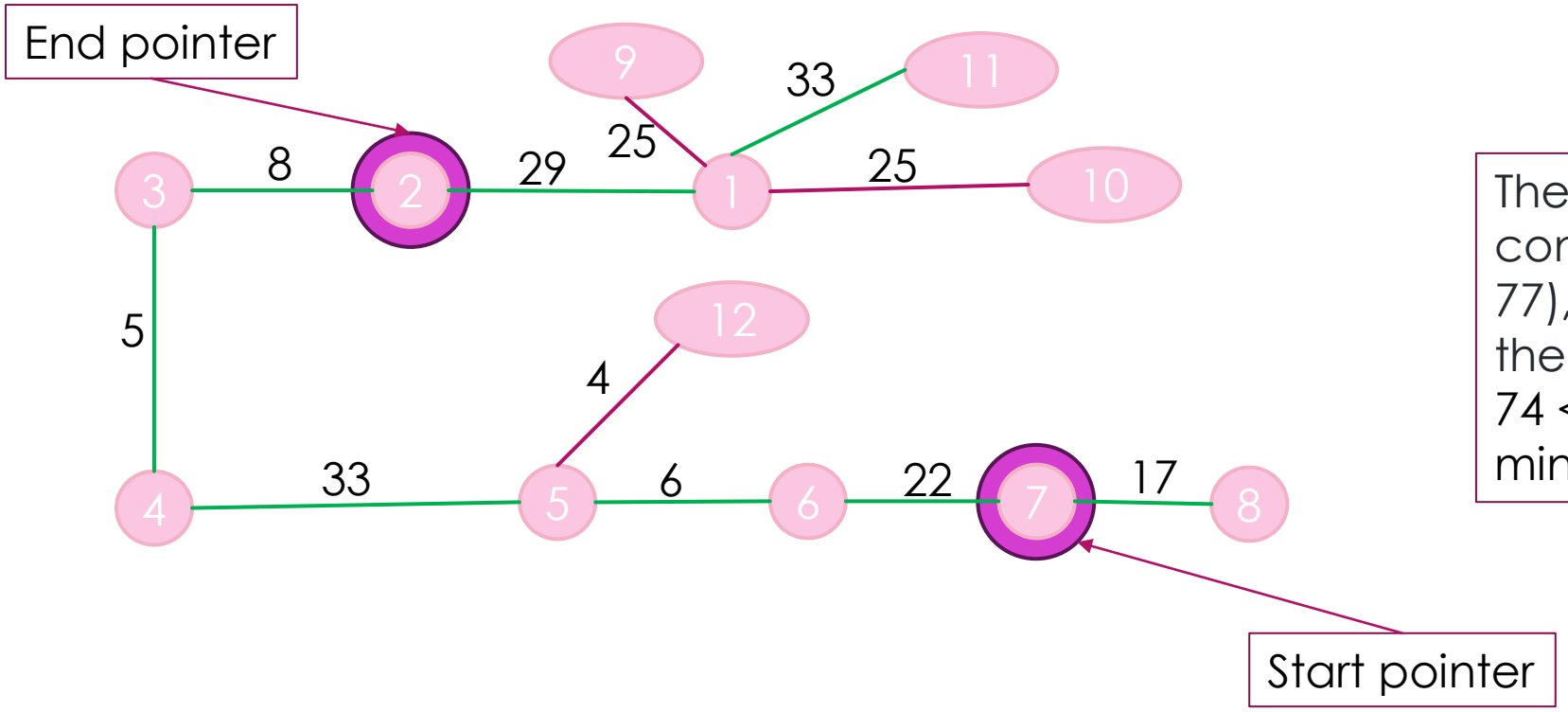
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 61 <= s = 77), expand the interval (move the end pointer), total weight = 66 <=77, the result is min(75,max(17,153-83))=70

Lab 3.B Step 3 Demo

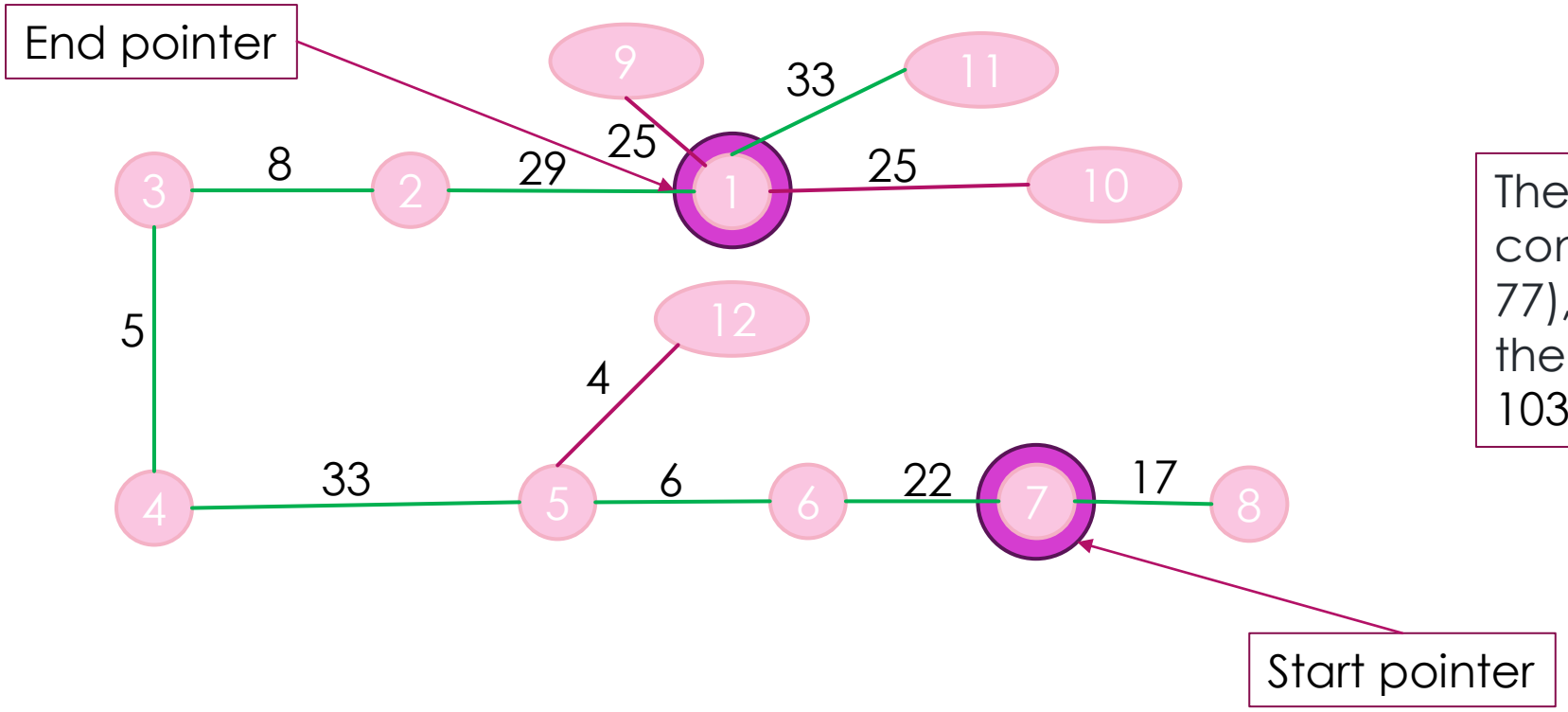
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 66 <= s = 77), expand the interval (move the end pointer), total weight = 74 <=77, the result is min(70,max(17,153-91))=62

Lab 3.B Step 3 Demo

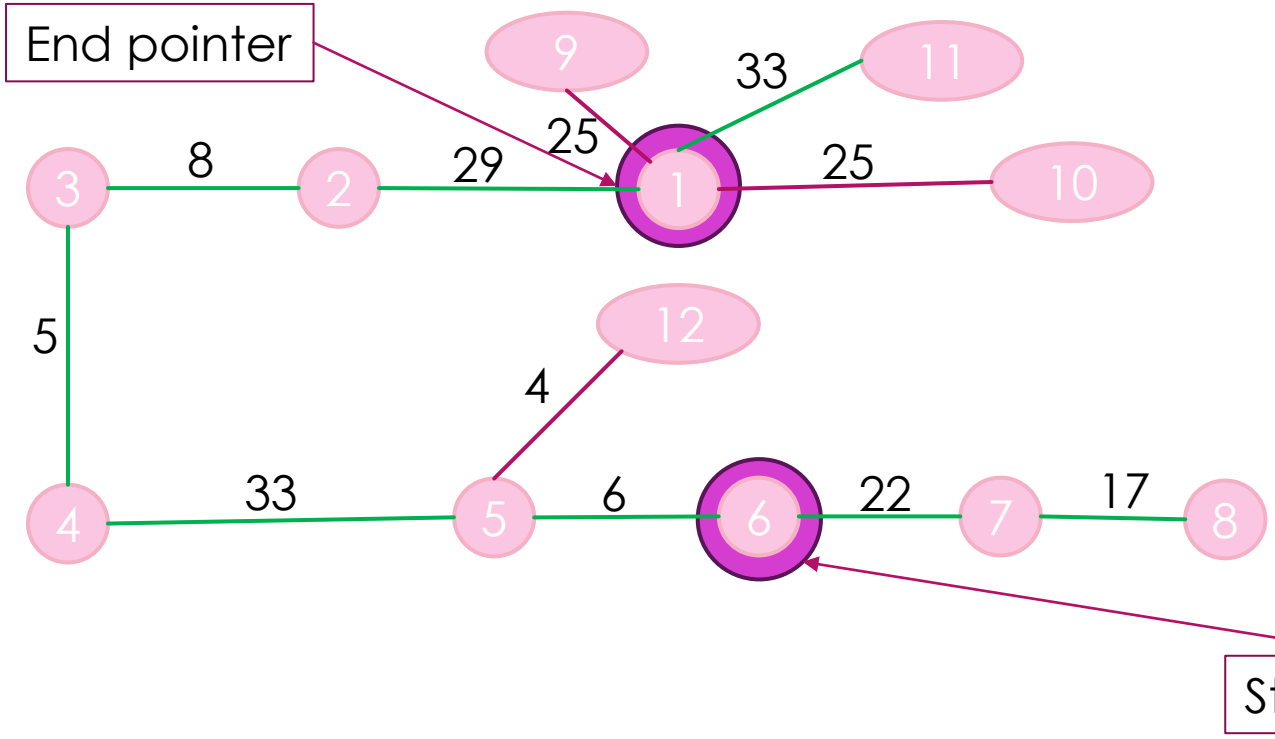
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 74 <= s = 77), expand the interval (move the end pointer), total weight = 103 > 77.

Lab 3.B Step 3 Demo

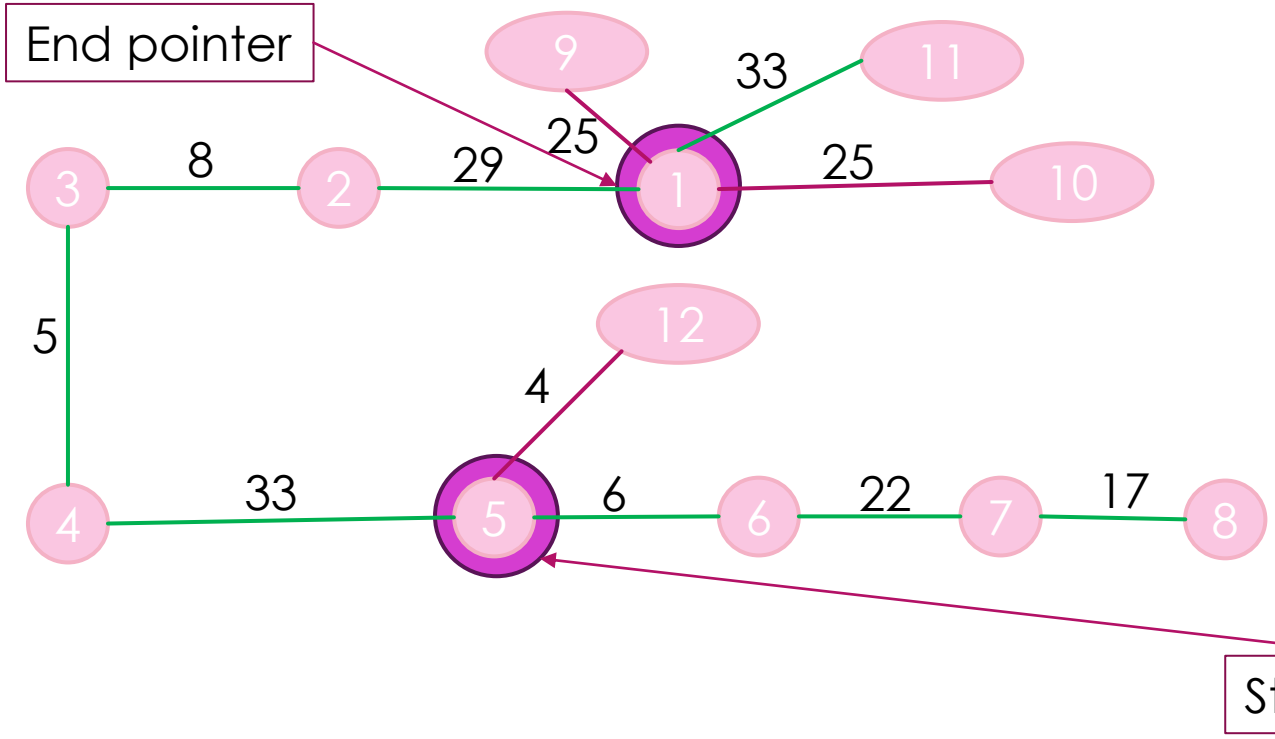
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval does not meet the condition (total weight 103 >77), shrink the interval (move the start pointer), total weight = 81 > 77.

Lab 3.B Step 3 Demo

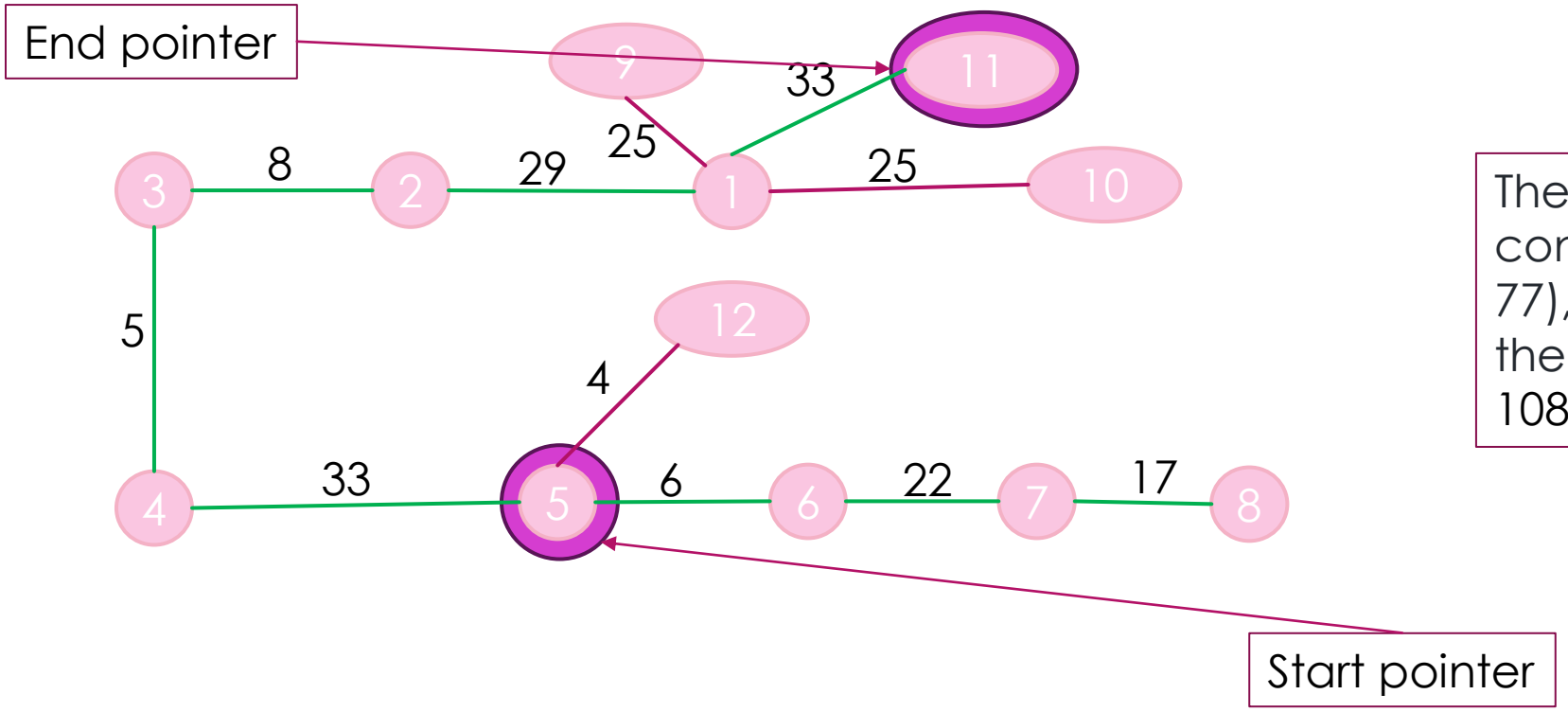
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval does not meet the condition (total weight 81 >77), shrink the interval (move the start pointer), total weight = 75 <= 77. the result is min(62,max(45,153-120))=45

Lab 3.B Step 3 Demo

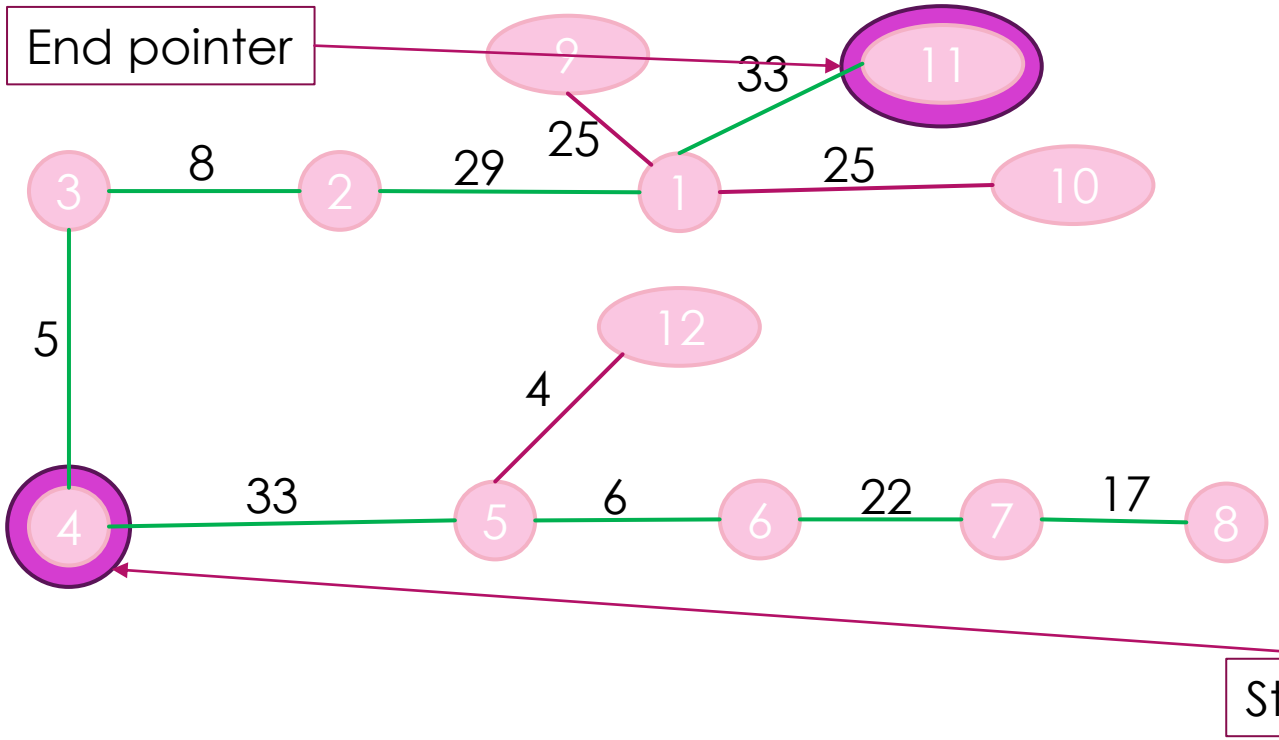
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 75 <= s = 77), expand the interval (move the end pointer), total weight = 108 > 77.

Lab 3.B Step 3 Demo

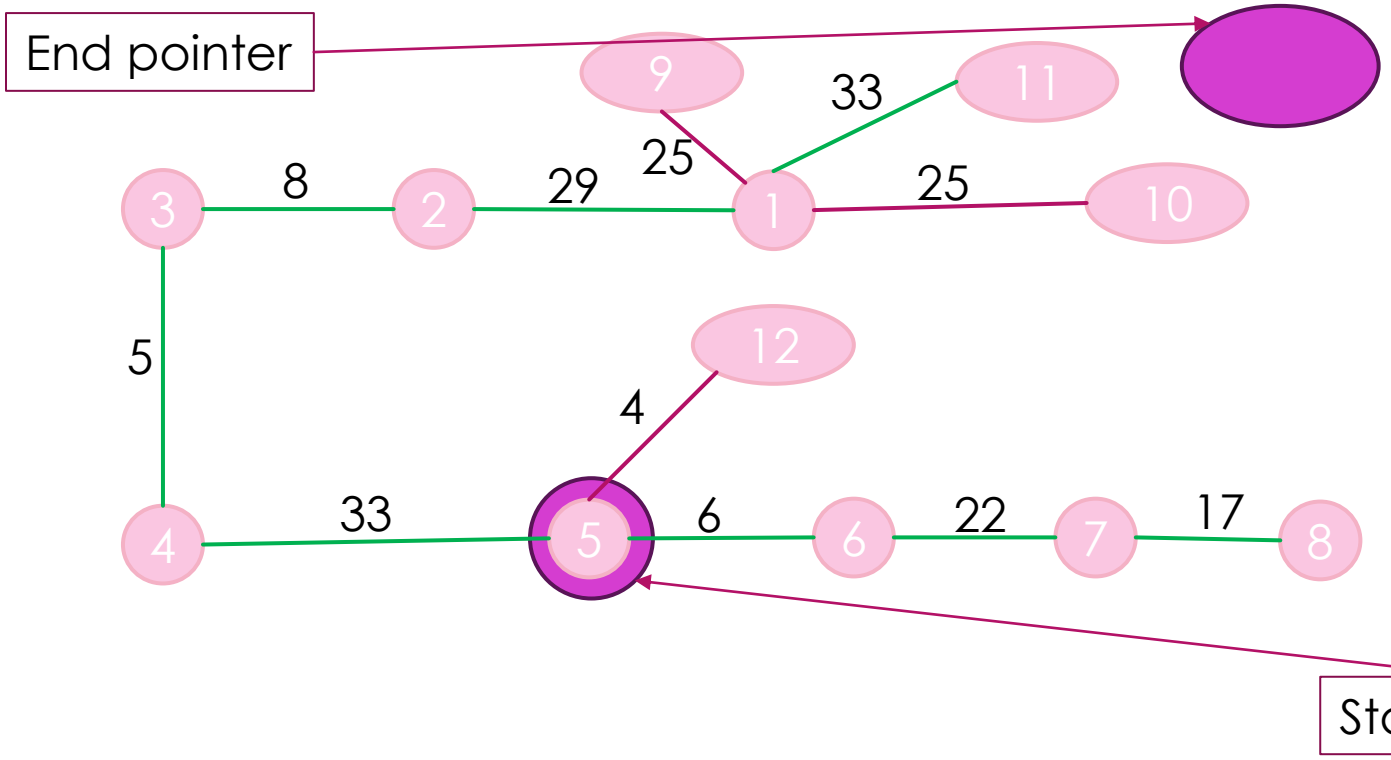
node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval does not meet the condition (total weight 108 >77), shrink the interval (move the start pointer), total weight = 75 <= 77. the result is min(45,max(78,153-153))=45.

Lab 3.B Step 3 Demo

node	8	7	6	5	4	3	2	1	11
dis	0	17	39	45	78	83	91	120	153



The current interval meets the condition (total weight 75 <= s = 77), expand the interval (move the end pointer), but the end pointer cannot move any further, **stop**.