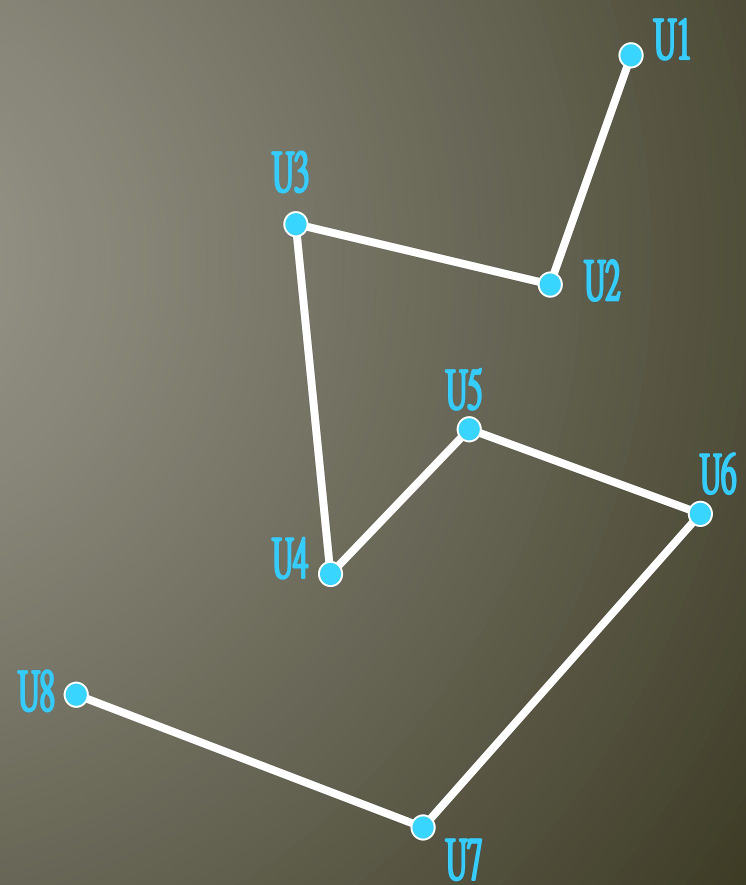
The Chain Method

- **Definition :**

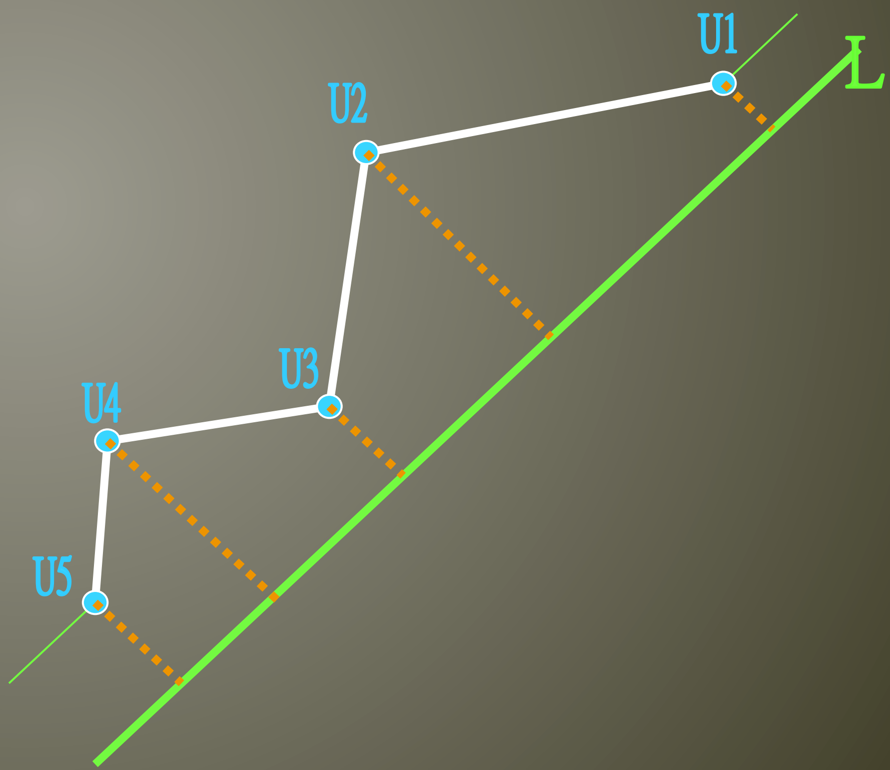
  A chain C = (u_1, … , u_p) is a planar straight-line graph with vertex set { u_1, … , u_p } and edge set { (u_i, u_{i+1}) : i = 1, …, p-1 }



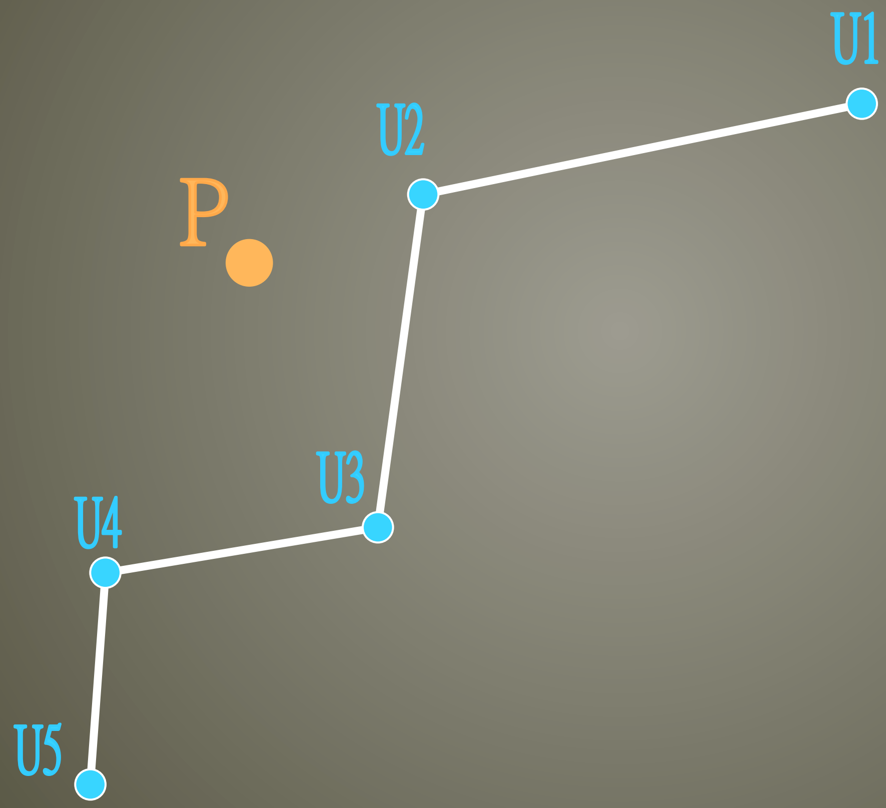
The Monotone Chain

- **Definition :**

  A chain C = (u_1, …, u_p) is said to be monotone with respect to a straight line L if a line orthogonal to L intersects C in exactly one point.

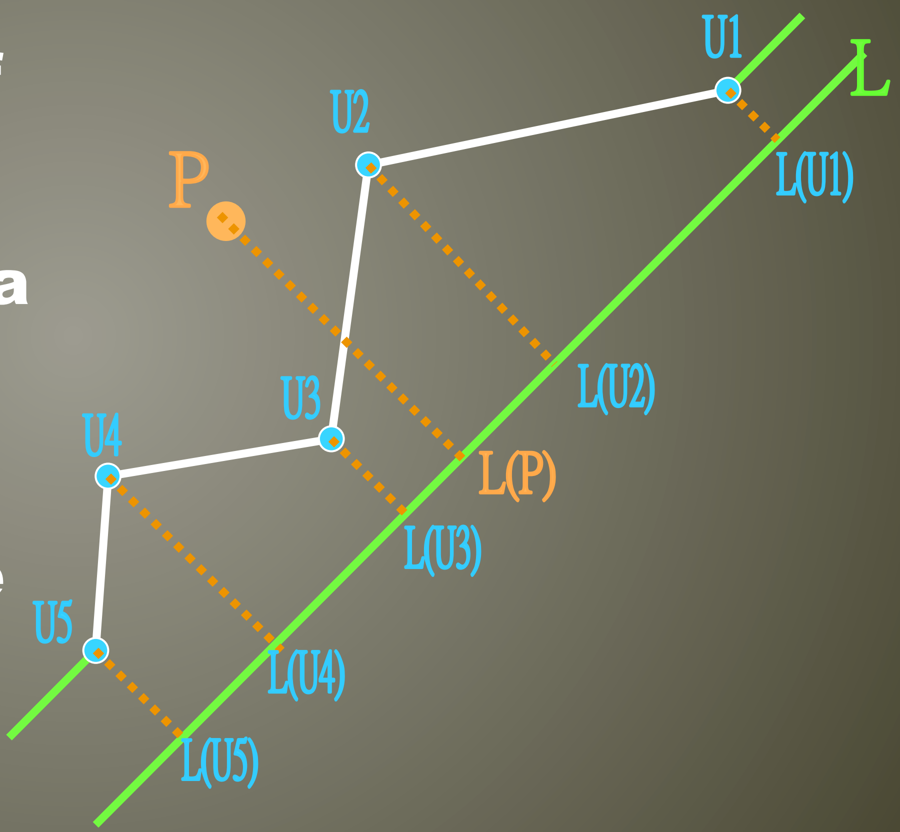


Where does the query point lie?



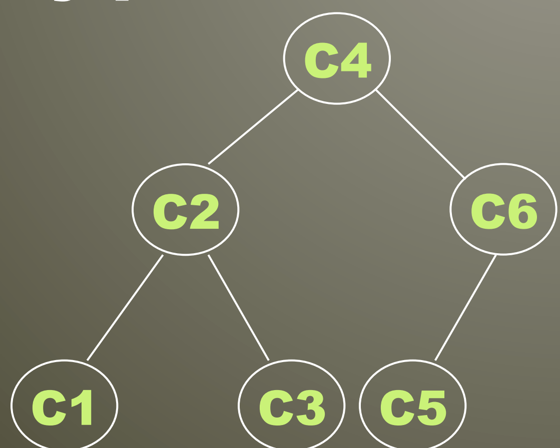
A query point lies ?

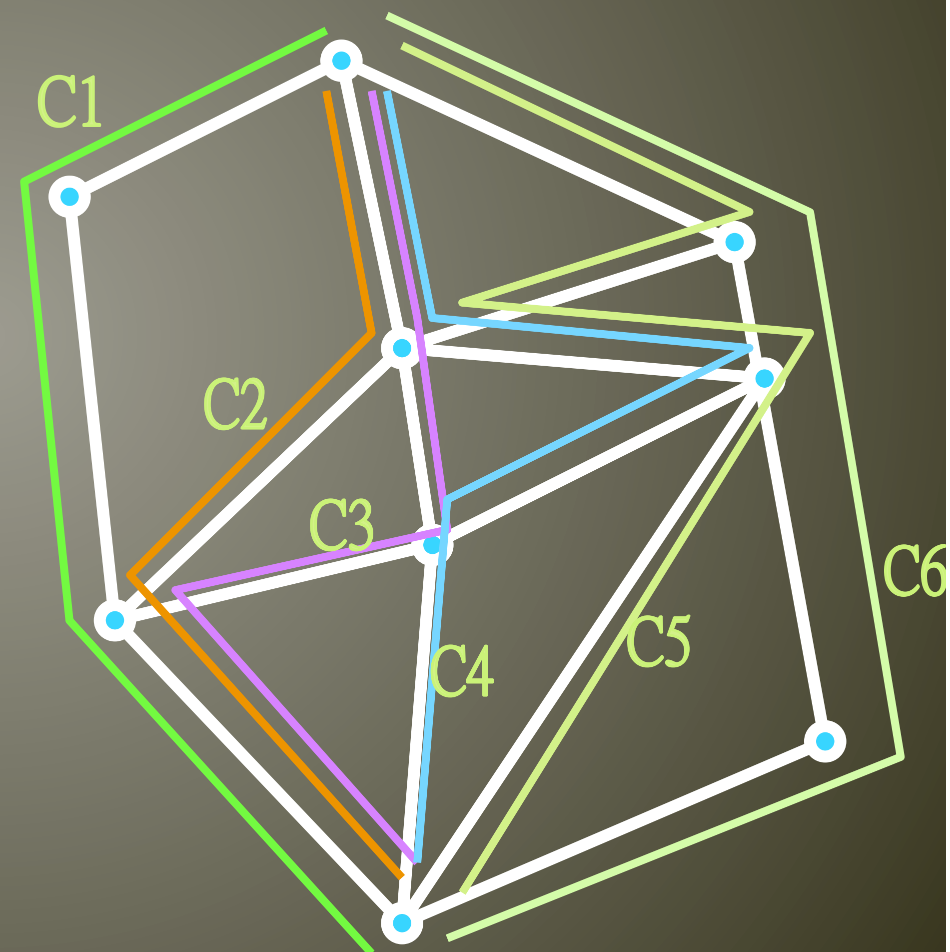
- The projection of P on L can be located with a binary search in a unique interval ($L(u_i), L(u_{i+1})$)
- Determine on which side of the line containing u_iu_{i+1} the query point lies.



The Chain Method

- Suppose there is a set of chains C = { C_1, …, C_r } of a PSLG. We can apply **bisection** to find in which region a query point lies.





The Chain Method (cont)

- If there are r chains in C and the longest chain has p vertices, then the search worst-case time is O(log p * log r)

Steps to Construct the Chains …

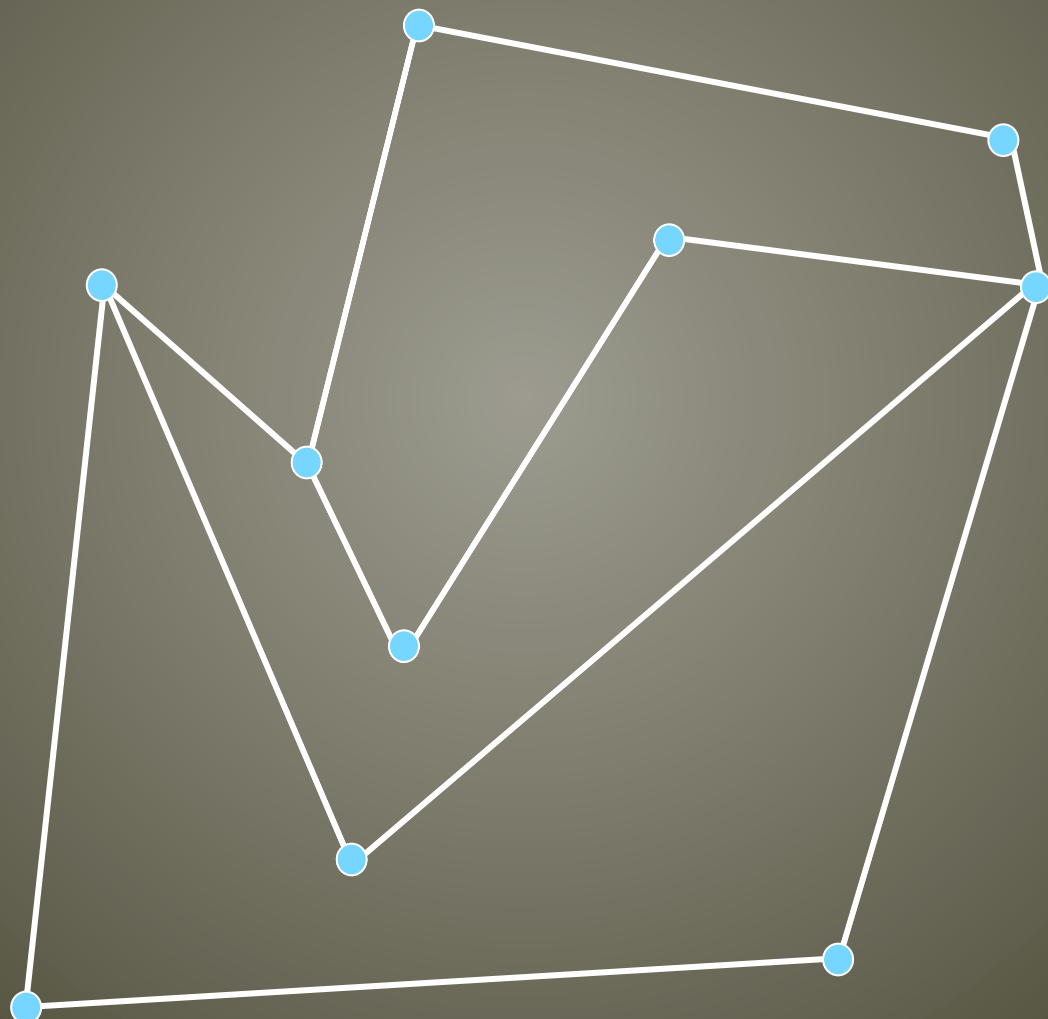
- First, regularize the PSLG
- Second, assign weights on the graph using *weight-balancing* algorithm
- Third, construct chains by traversing the graph

Definitions

(*for chains monotone w.r.t. y*)

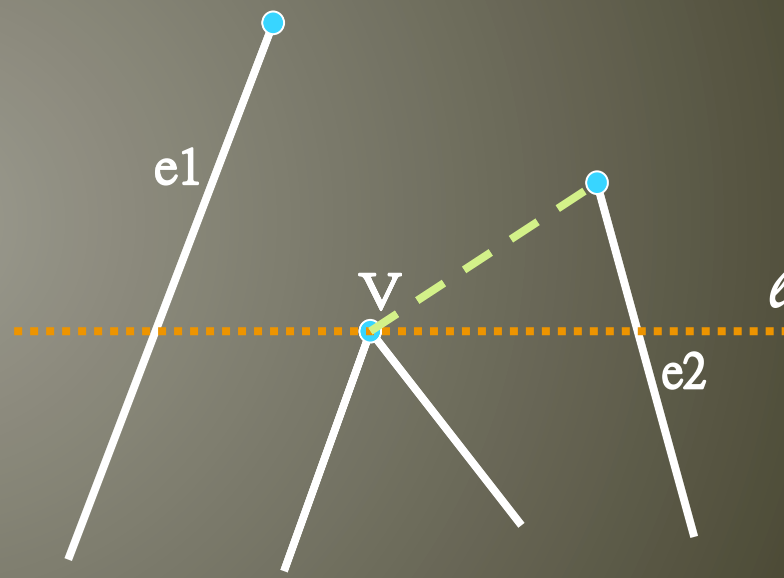
1. A vertex v_j is said to be *regular* if there are vertices $y(v_i) < y(v_j) < y(v_k)$ such that (v_i, v_j) and (v_i, v_k) are edges of G.
2. Graph G is said to be regular if each v_j is regular for $1 < j < N$

Example: non regular

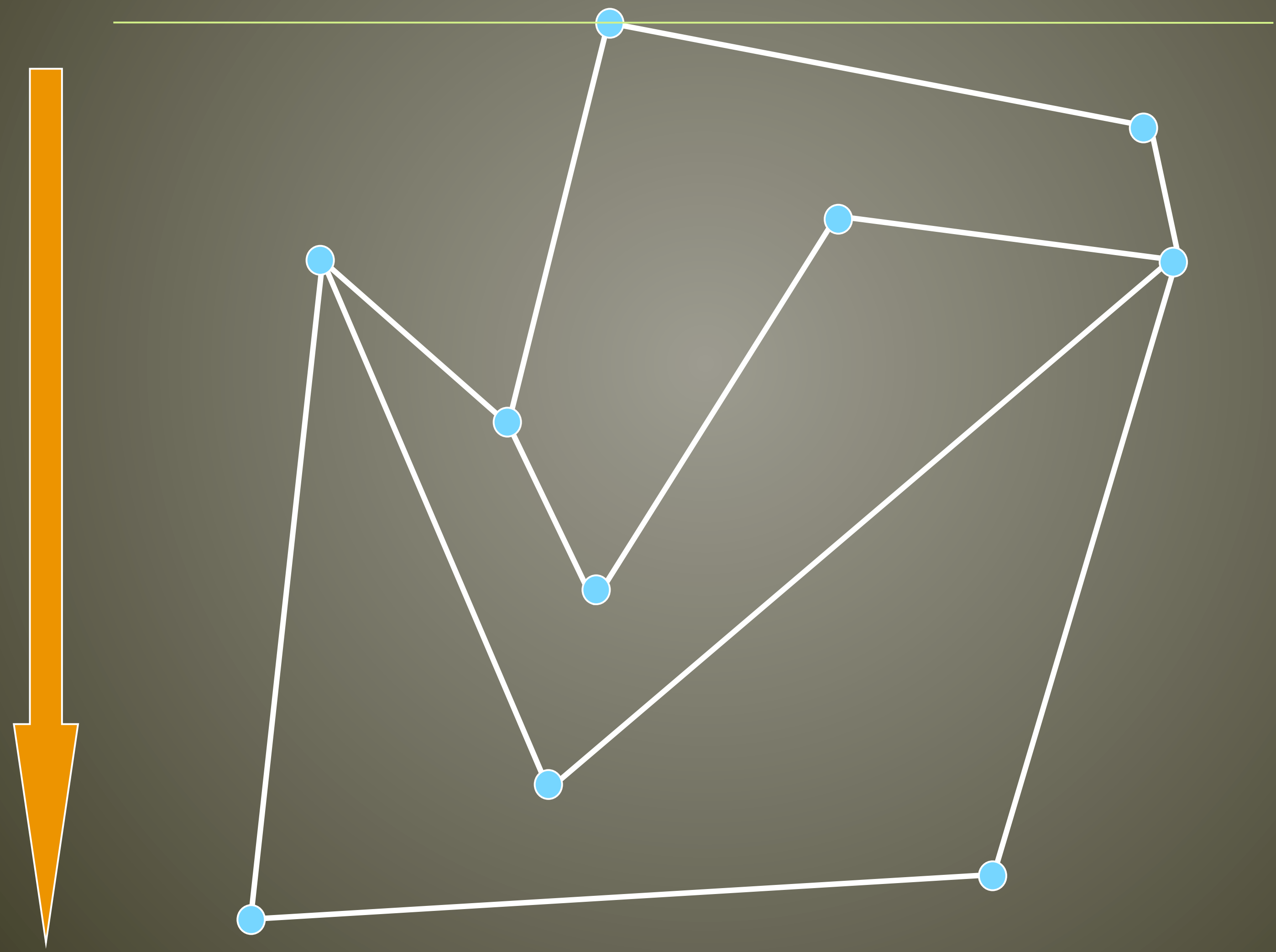


Regularize Nonregular Vertices (remove cusps)

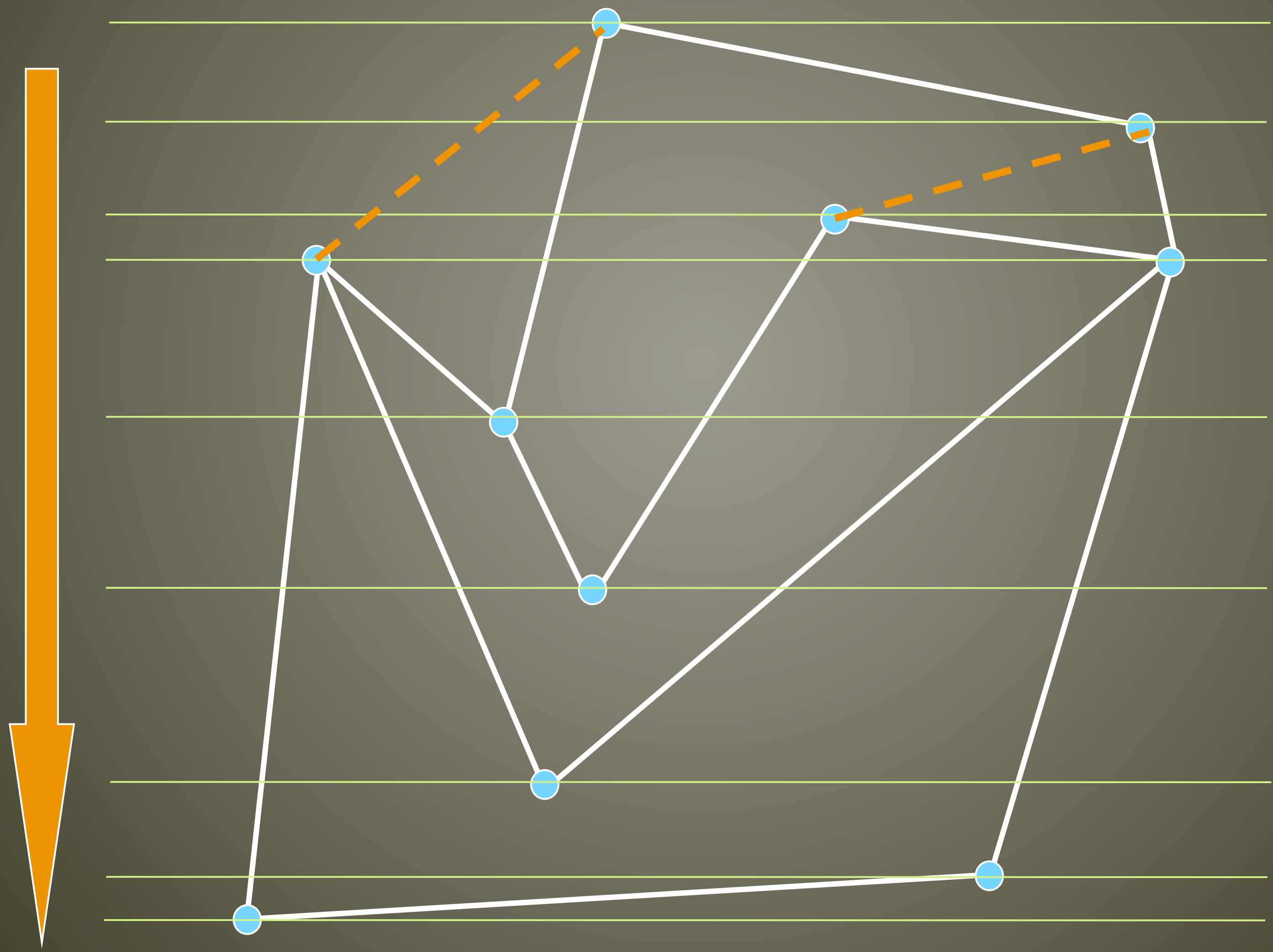
- **Definition: cusp**
- **2-pass sweep-line to remove cusps**
  - **Connect to the vertex closest to (above) ℓ .**



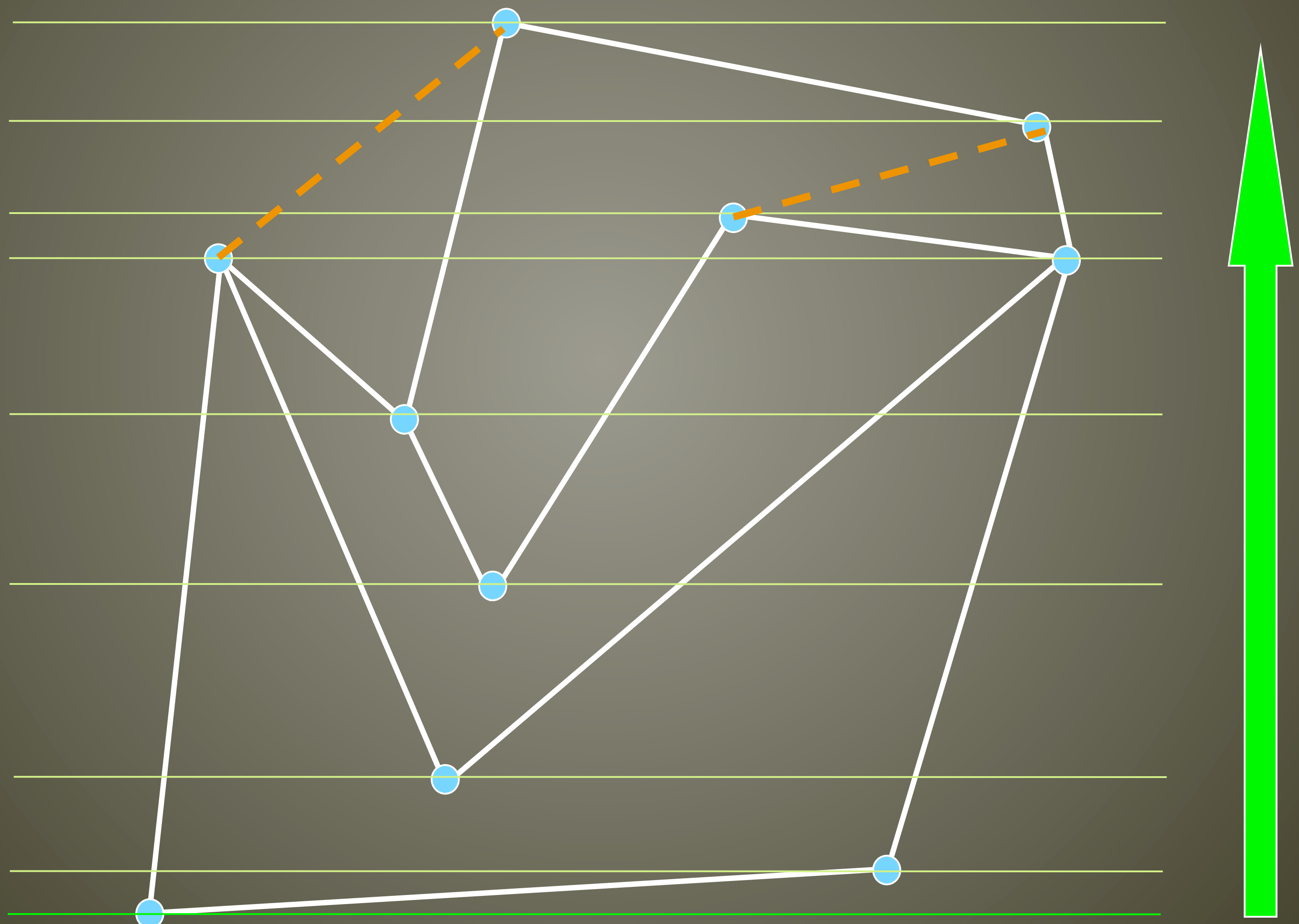
Example: cusp removal



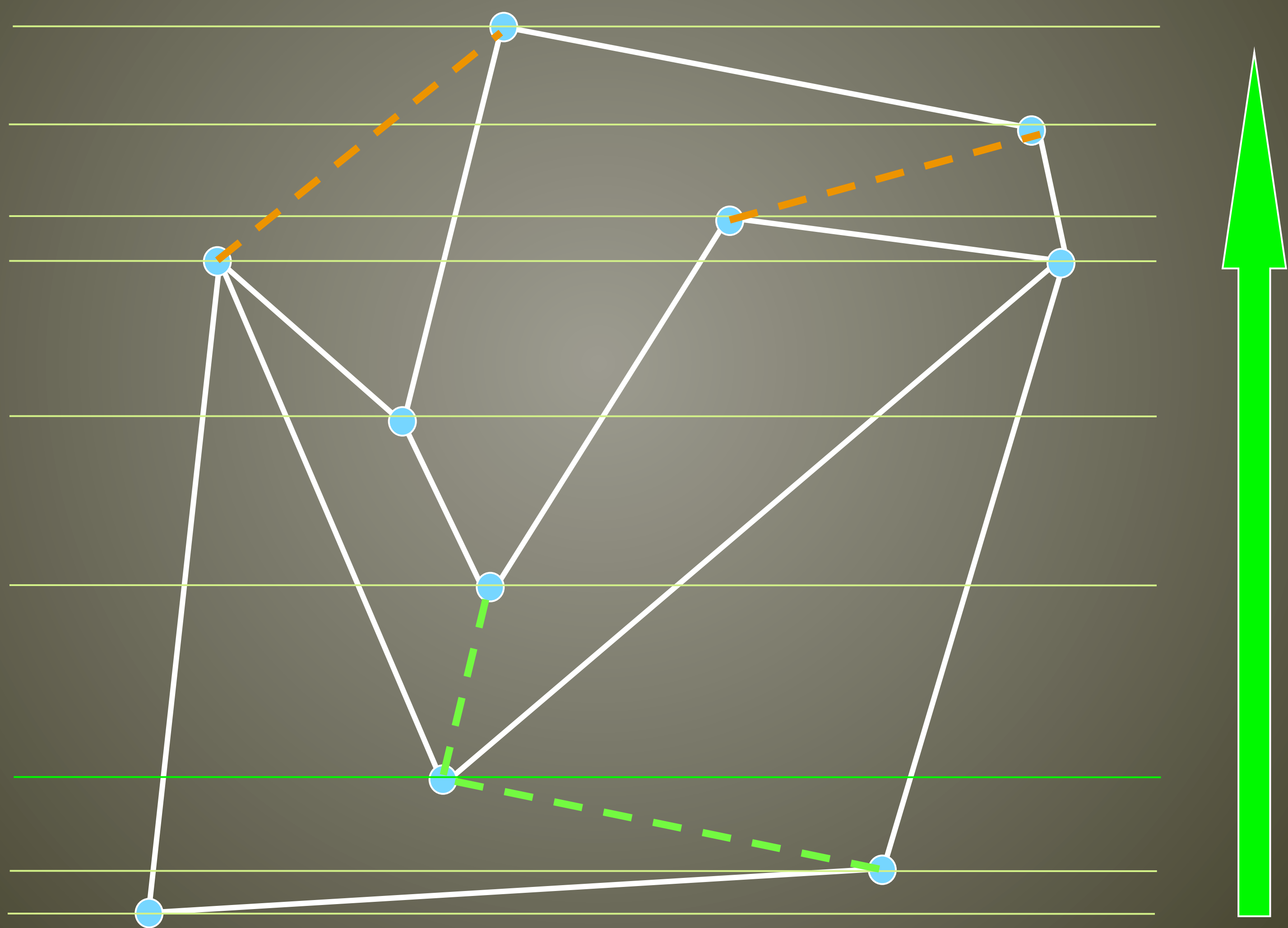
Example: cusp removal



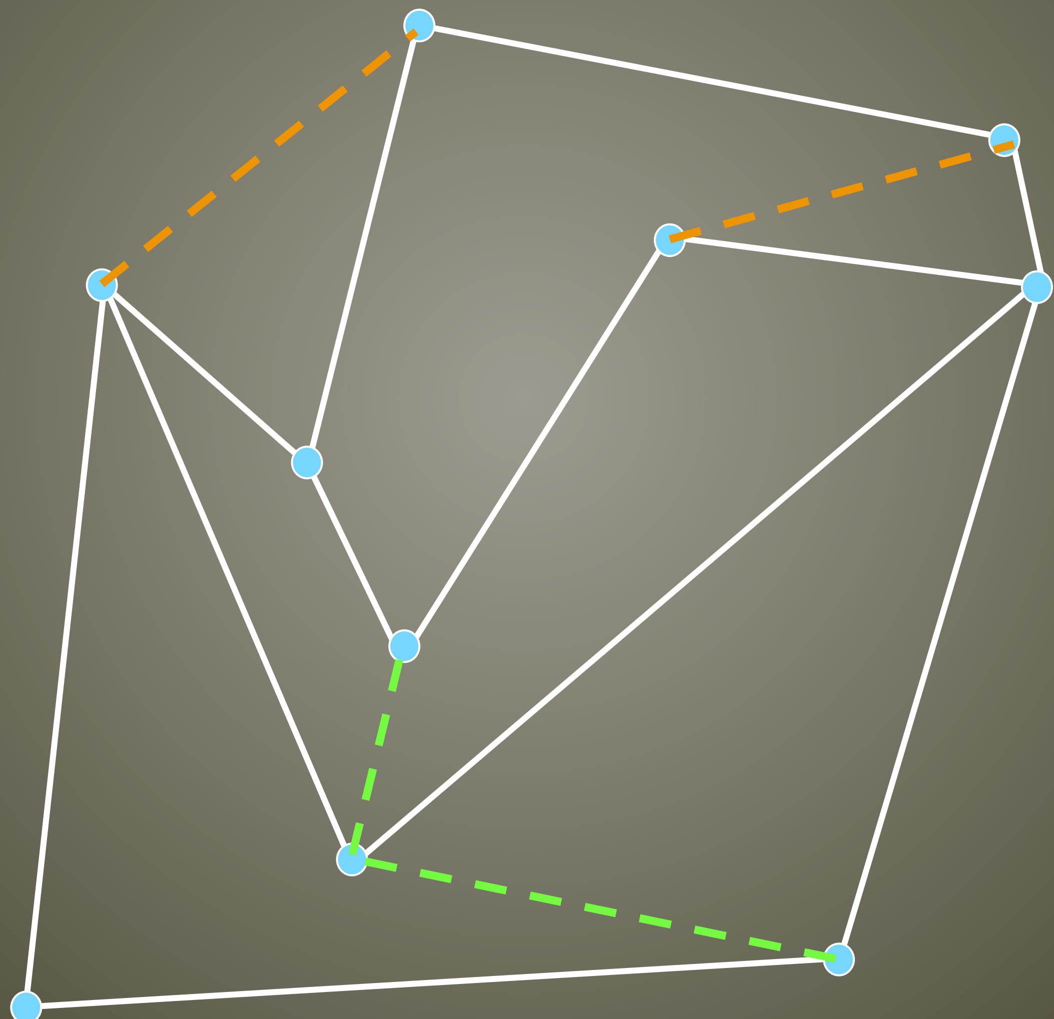
Example: cusp removal



Example: cusp removal



Example: cusp removal



Weight Assignment of Edges

- All edges satisfy :

1. Each edge has positive weight
2. For each V_j($j \neq 1, N$),

   $W_{in}(V_j) = W_{out}(V_j)$

PS.

$$Win(Vi) = \sum_{e \in IN(v)} W(e)$$

$$Vin(Vi) = | IN(v) |$$

$$Wout(Vi) = \sum_{e \in OUT(v)} W(e)$$

$$Vout(Vi) = | OUT(v) |$$

WEIGHT-BALANCING REGULAR PSLG

1. Initialization

```
for each edge e do W(e) = 1
```

2. First pass

for(i = 2; i <= N-1; i++) {

$W_{in}(V_i)$ = sum of weight of incoming edges of V_i

if ($W_{in}(V_i) > V_{out}(V_i)$) {

d_1 = leftmost outgoing edge of V_i

$W(d_1) = W_{in}(V_i) - V_{out}(V_i) + 1$

}

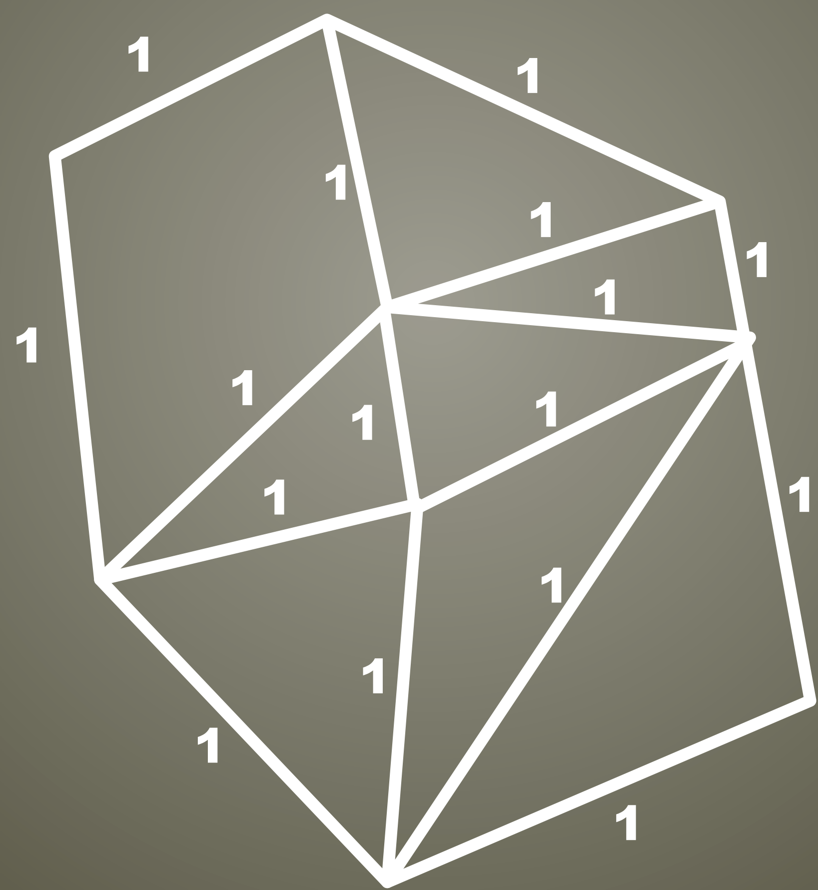
}

WEIGHT-BALANCING REGULAR PSLG (cont)

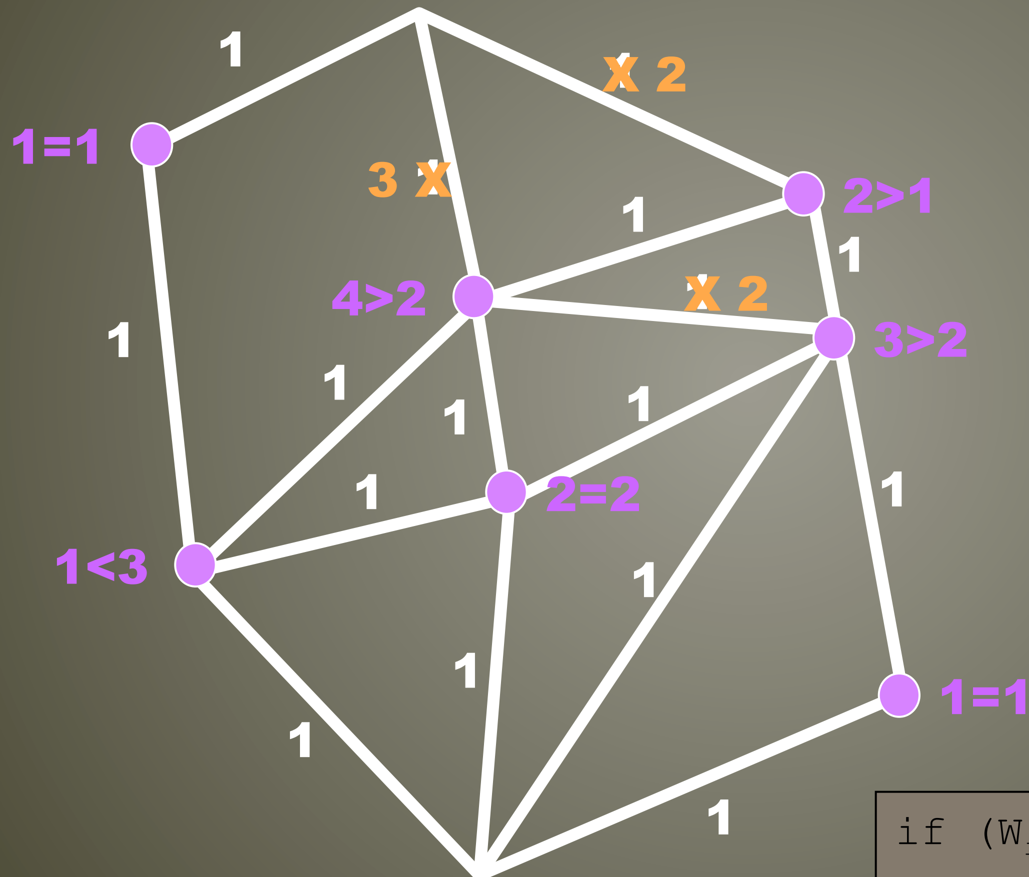
3. Second pass

```
For (i = N-1; i >= 2; i--){
   W_out(V_i) = sum of weight of outgoing edges of V_i
   if (W_out(V_i)  > W_in(V_i)) {
       d_2 = leftmost incoming edge of V_i
       W(d_2) = W_out(V_i) - W_in(V_i) + W(d_2)
    }
}
```

Example: Initialization

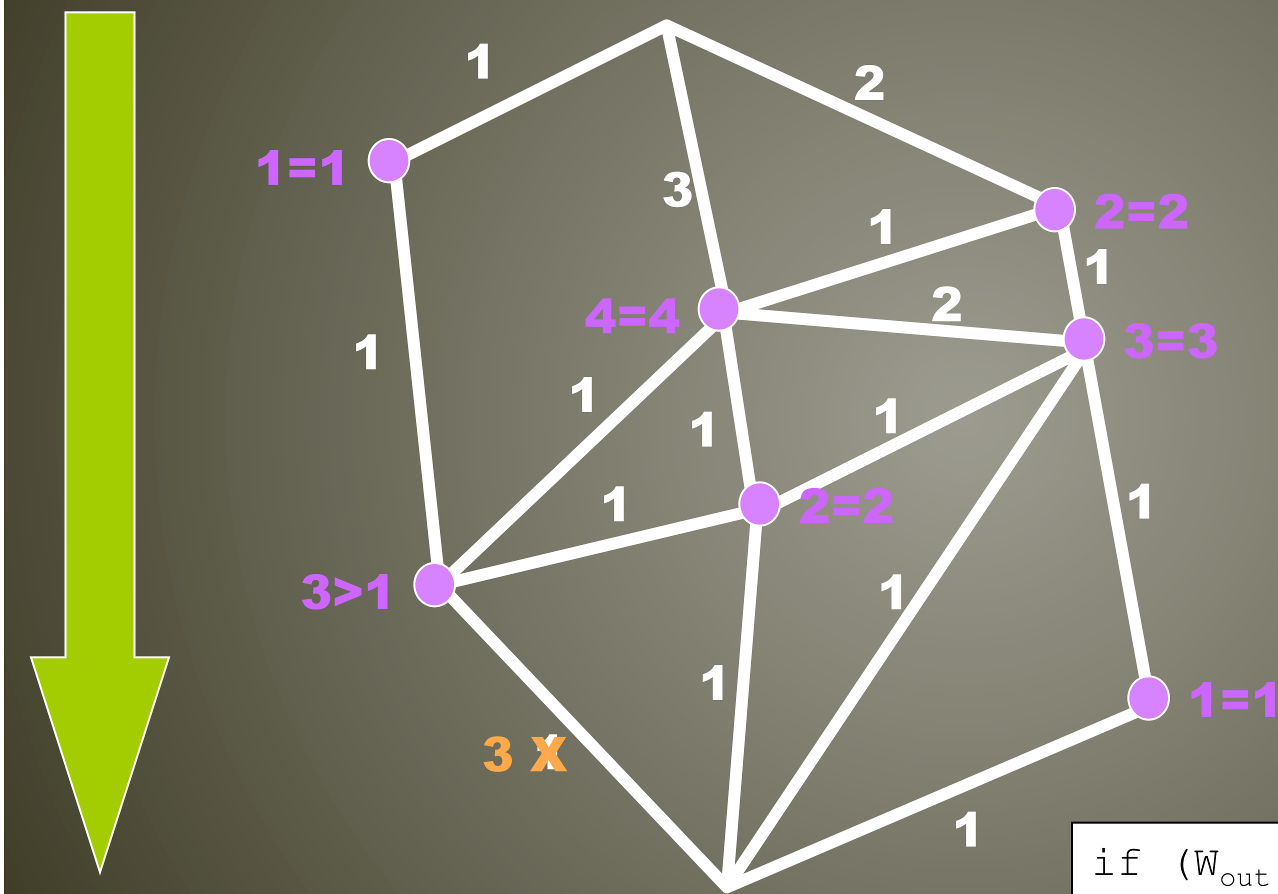


Example: 1^{st} pass




```
if (W_in(V_i) > V_out(V_i)) {
 d_1 = leftmost outgoing edge (V_i)
 W(d_1) = W_in(V_i) - V_out(V_i) + 1
```

Example: 2nd pass

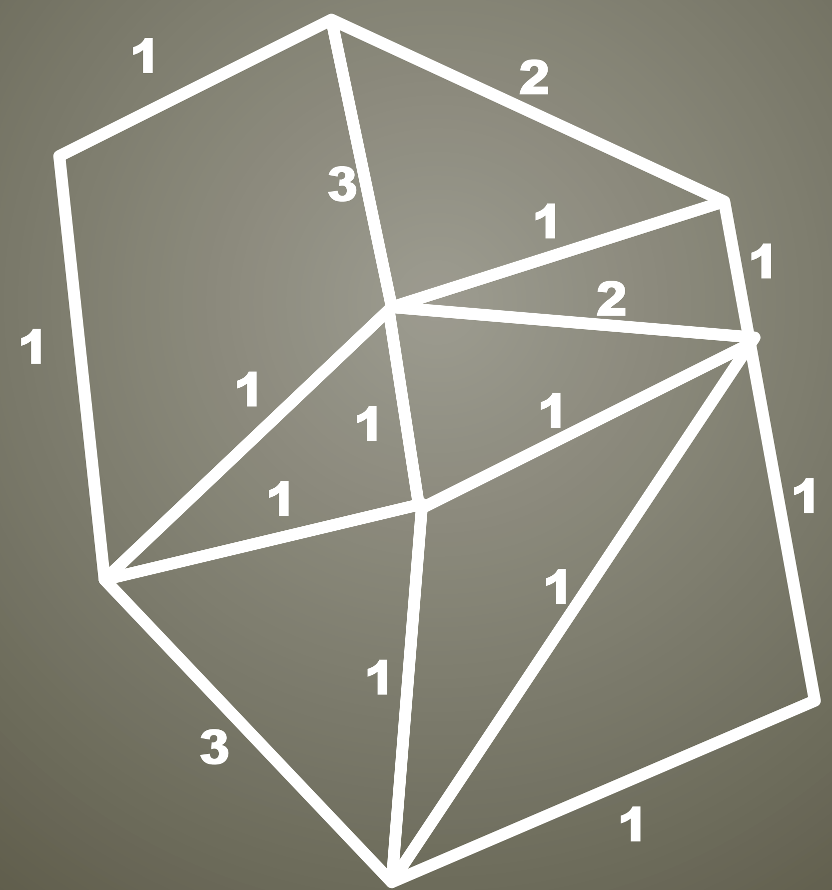


if $(W_{out}(V_i) > W_{in}(V_i))$ {

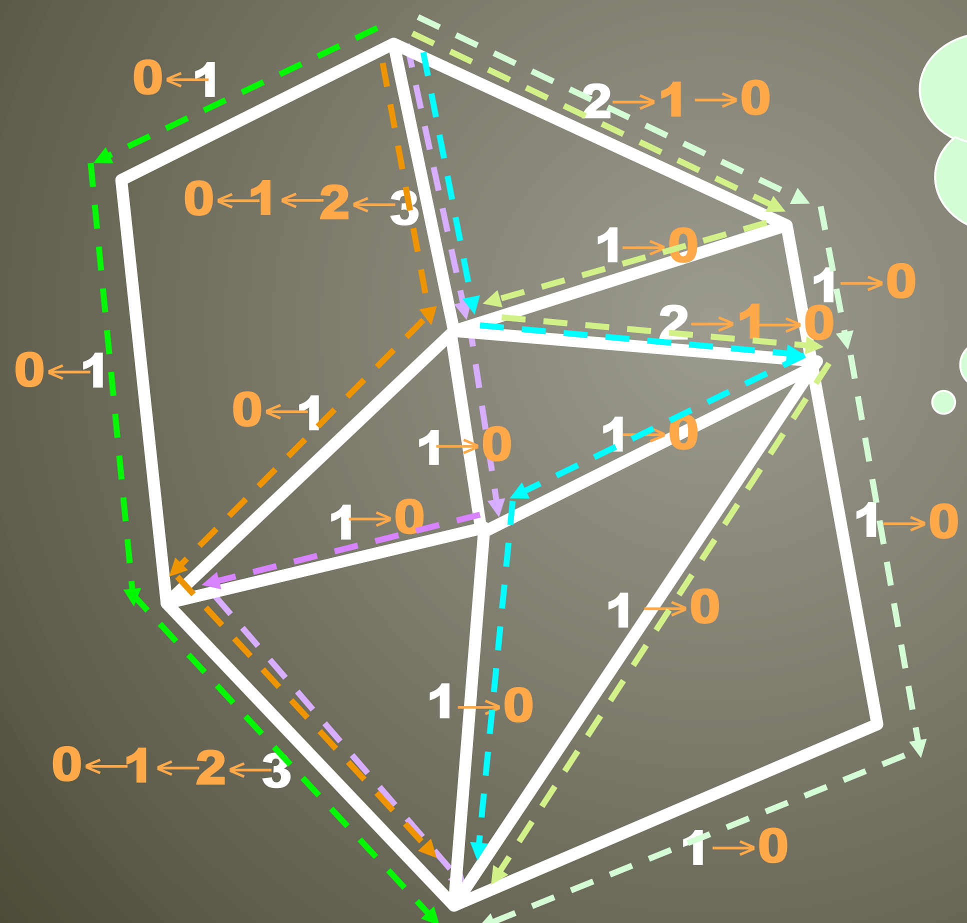
d_2 = leftmost incoming edge (V_i)

$W(d_2) = W_{out}(V_i) - W_{in}(V_i) + W(d_2)$

Example: final result

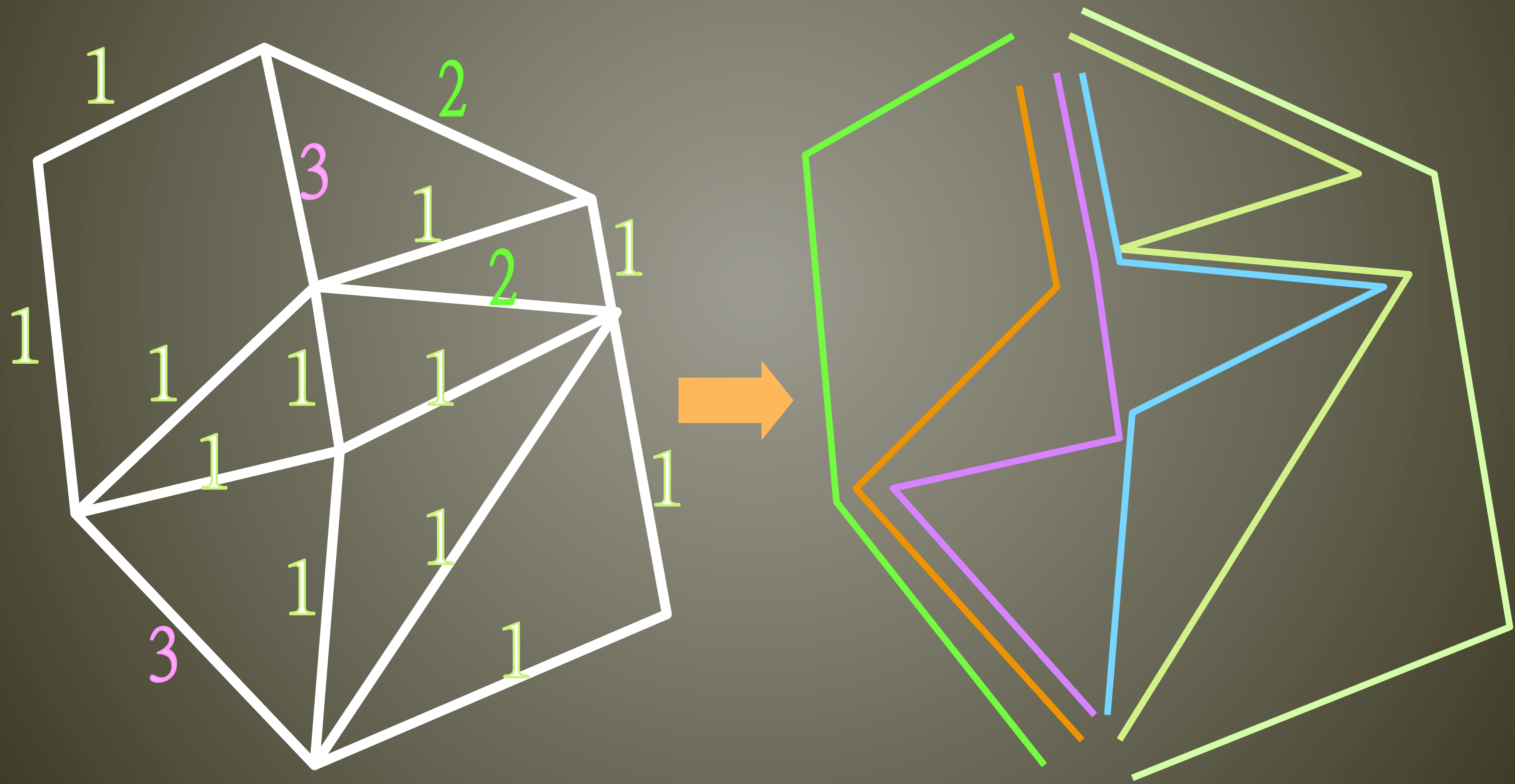


Ex: construct chains



Start from top; traverse the leftmost possible branch

Ex: Polygon and Monotone Chains



Theorem

- Time complexity : $O(\log^2 N)$
- Space complexity : $O(N)$
- Preprocessing time: $O(N \log N)$

1. Sorting N vertex of PSLG in O(NlogN)
2. Construct status structure, each node need O(logN)

Exercise

- Apply the chain method to determine which region the point P lies.