Construction of a Trapezoidal Map

Outline:

I. Randomized incremental construction

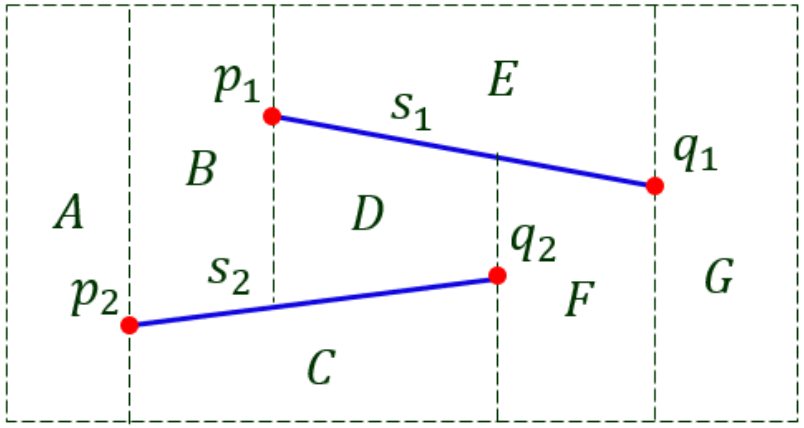
II. Trapezoidal map modification

III. Search structure update

IV. Analysis of expected query time

Search Structure

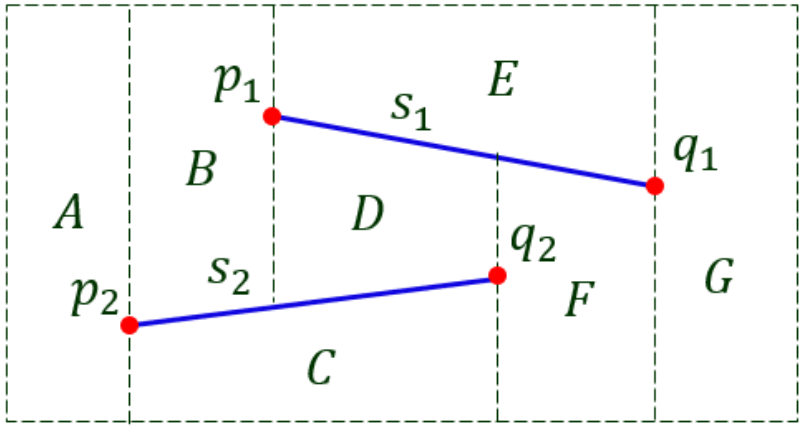
Trapezoidal map $T(S)$



Doubly-connected edge list (DCEL)

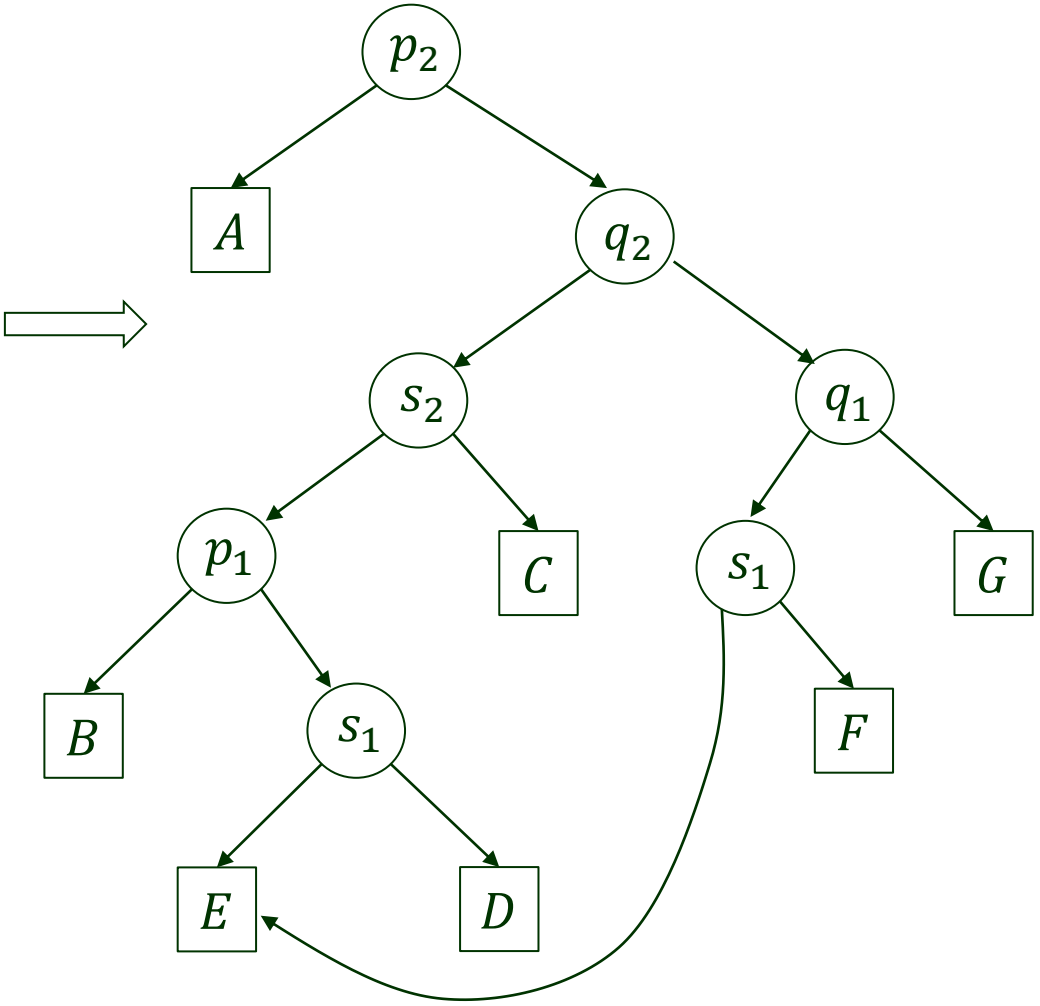
Search Structure

Trapezoidal map $T(S)$



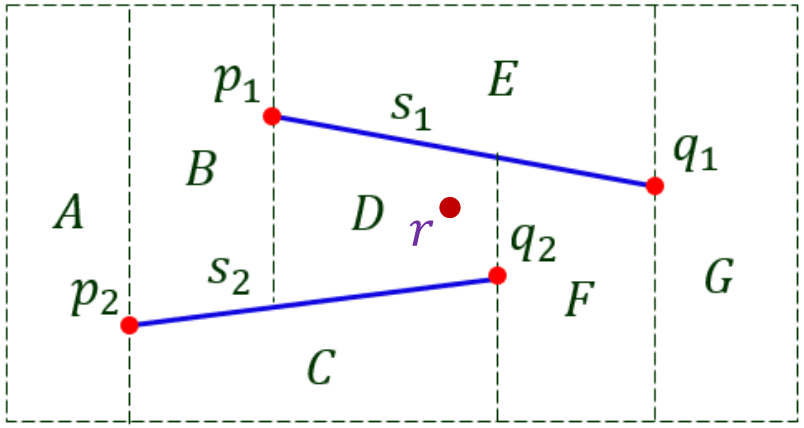
Doubly-connected edge list (DCEL)

Search structure D



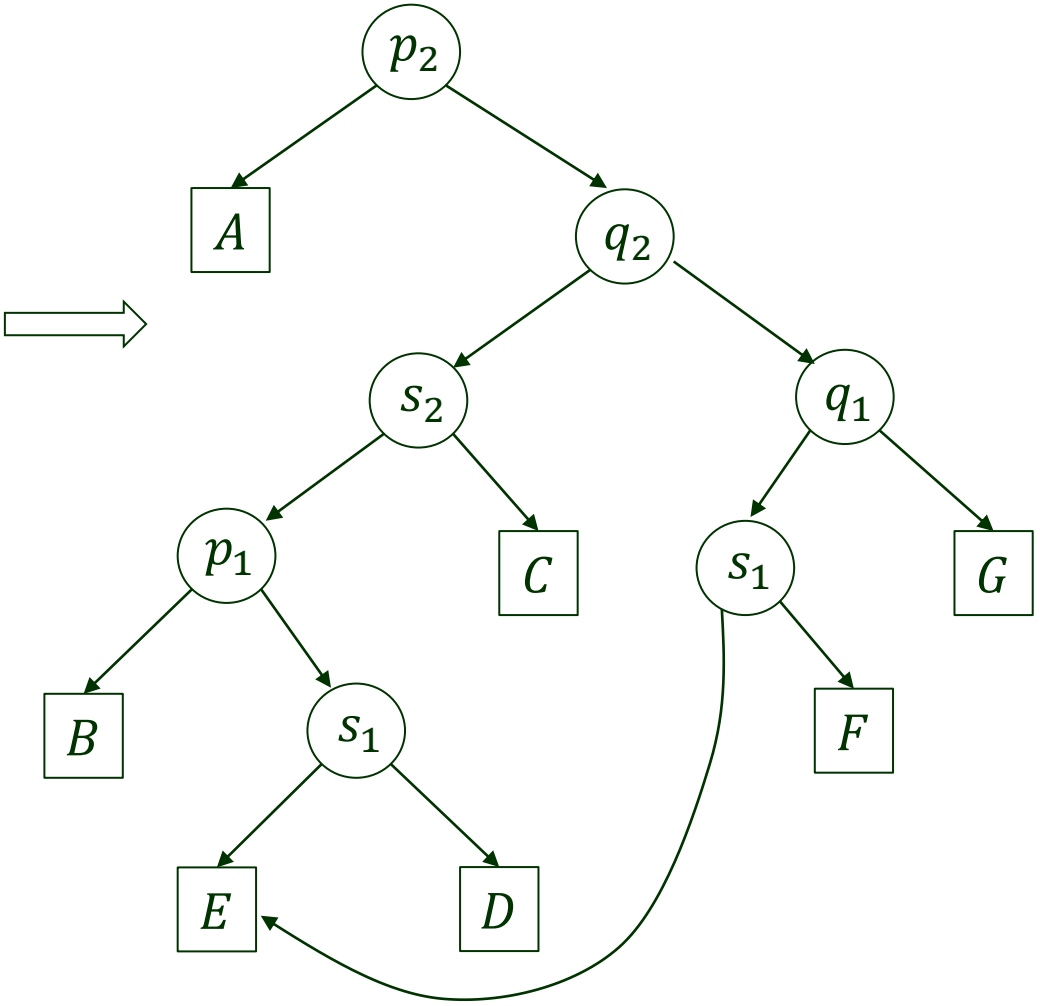
Search Structure

Trapezoidal map $T(S)$



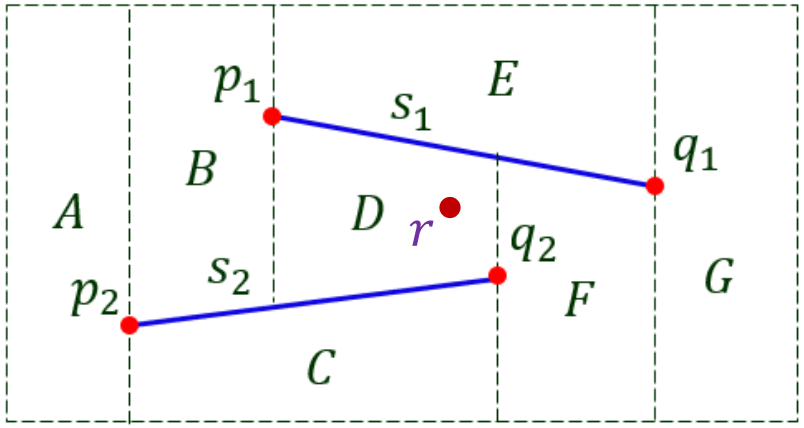
Doubly-connected edge list (DCEL)

Search structure D



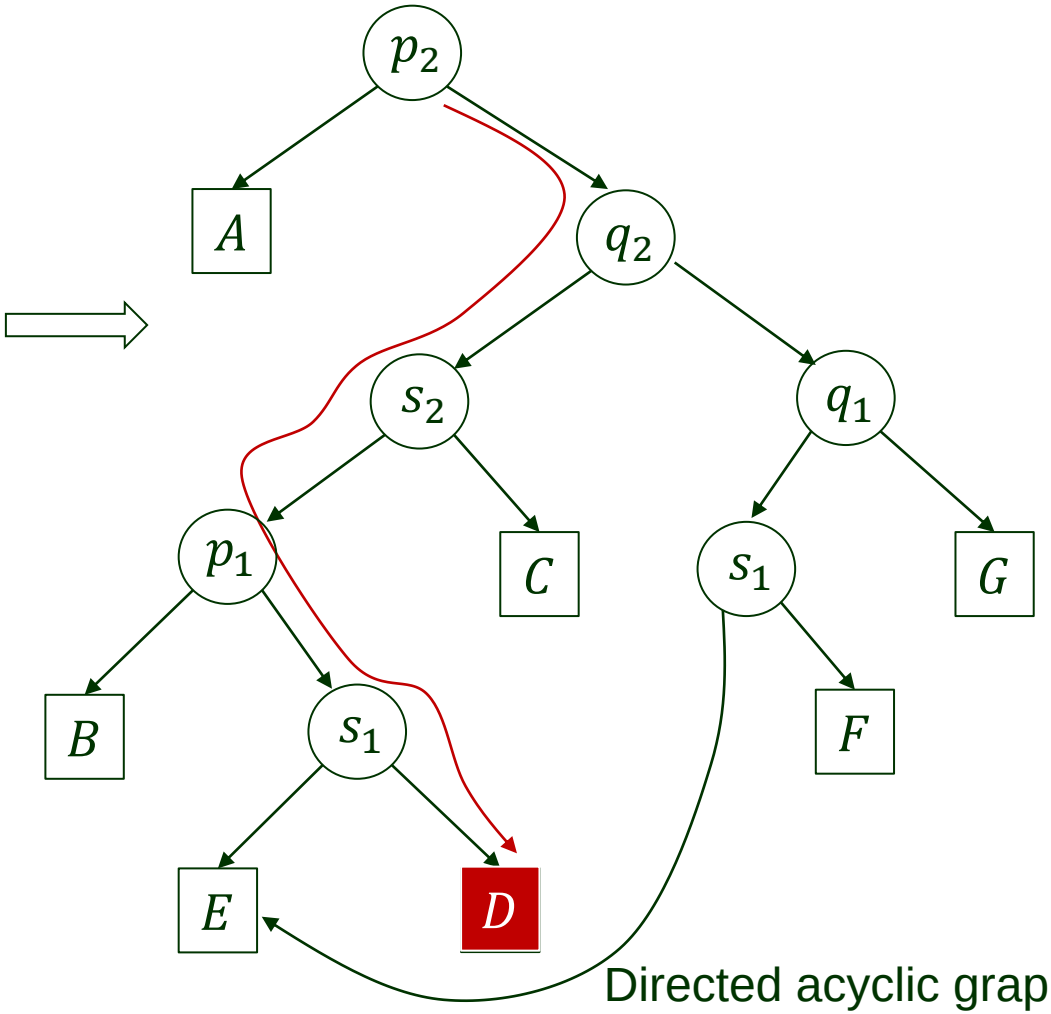
Search Structure

Trapezoidal map $T(S)$



Doubly-connected edge list (DCEL)

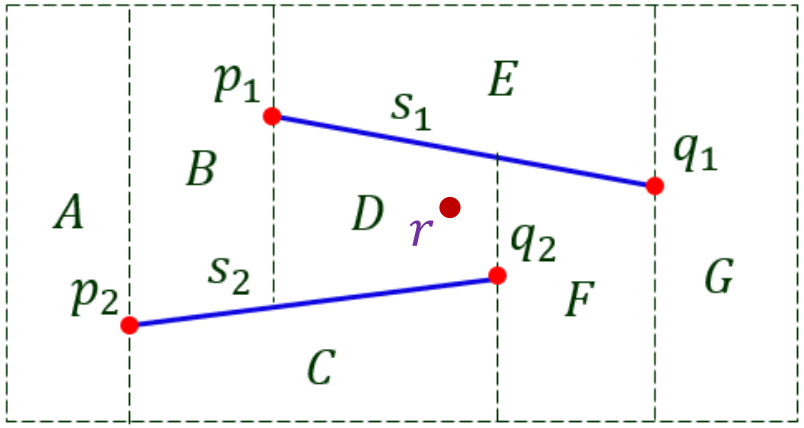
Search structure D



Directed acyclic graph (DAG)

Search Structure

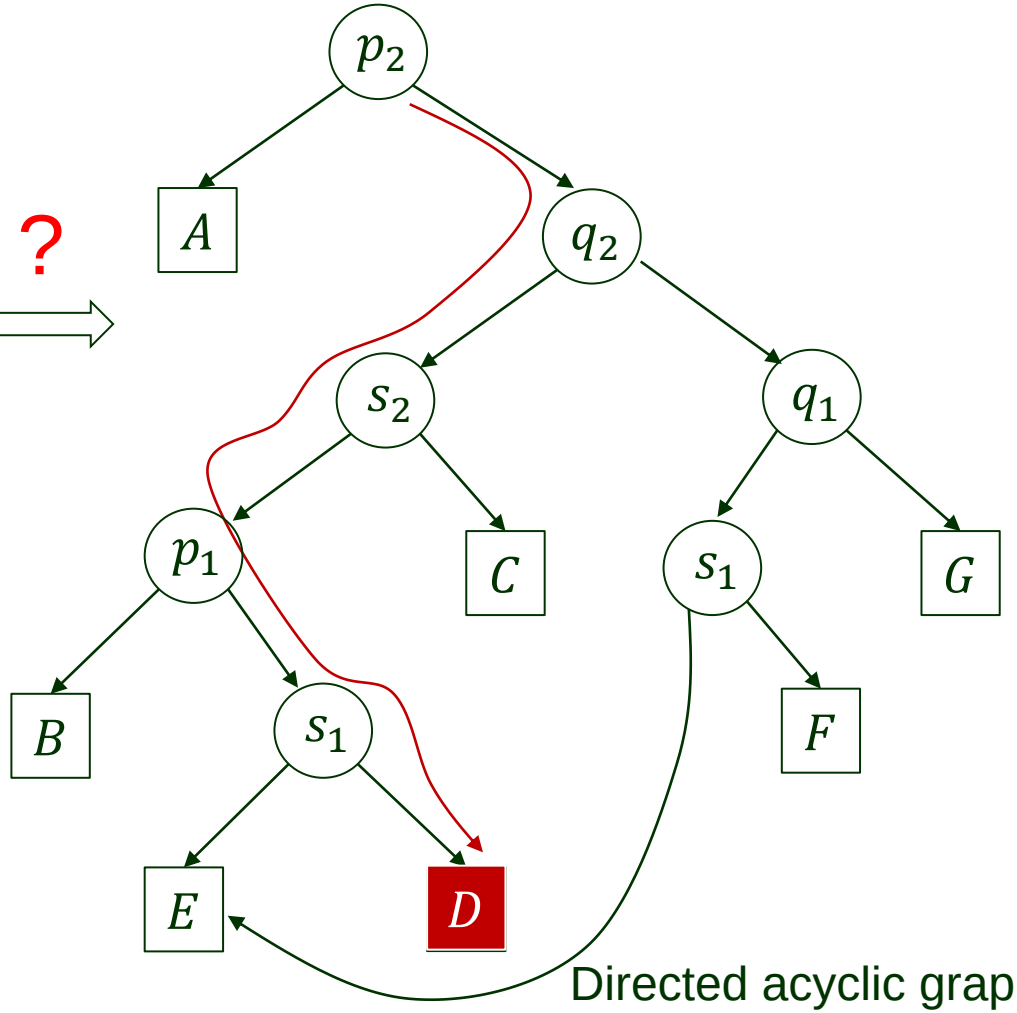
Trapezoidal map $T(S)$



Doubly-connected edge list (DCEL)

?

Search structure D



Directed acyclic graph (DAG)

I. Randomized Incremental Construction of D

- Randomly order segments: $s_1, s_2, \ldots, s_n$.
- Add one segment at a time.

I. Randomized Incremental Construction of D

- Randomly order segments: $s_1, s_2, \ldots, s_n$.
- Add one segment at a time. $S_i = \{s_1, s_2, \ldots, s_i\}$

I. Randomized Incremental Construction of D

- Randomly order segments: $s_1, s_2, \ldots, s_n$.
- Add one segment at a time. $S_i = \{s_1, s_2, \ldots, s_i\}$
- After each addition, update the search structure and trapezoidal map.

I. Randomized Incremental Construction of D

- Randomly order segments: $s_1, s_2, \ldots, s_n$.
- Add one segment at a time. $S_i = \{s_1, s_2, \ldots, s_i\}$
- After each addition, update the search structure and trapezoidal map.

```
for i ← 1 to n
    T(S_i) → T(S_{i+1})
```

I. Randomized Incremental Construction of D

- Randomly order segments: $s_1, s_2, \ldots, s_n$.
- Add one segment at a time. $S_i = \{s_1, s_2, \ldots, s_i\}$
- After each addition, update the search structure and trapezoidal map.

for $i \leftarrow 1$ to n
  $T(S_i) \rightarrow T(S_{i+1})$

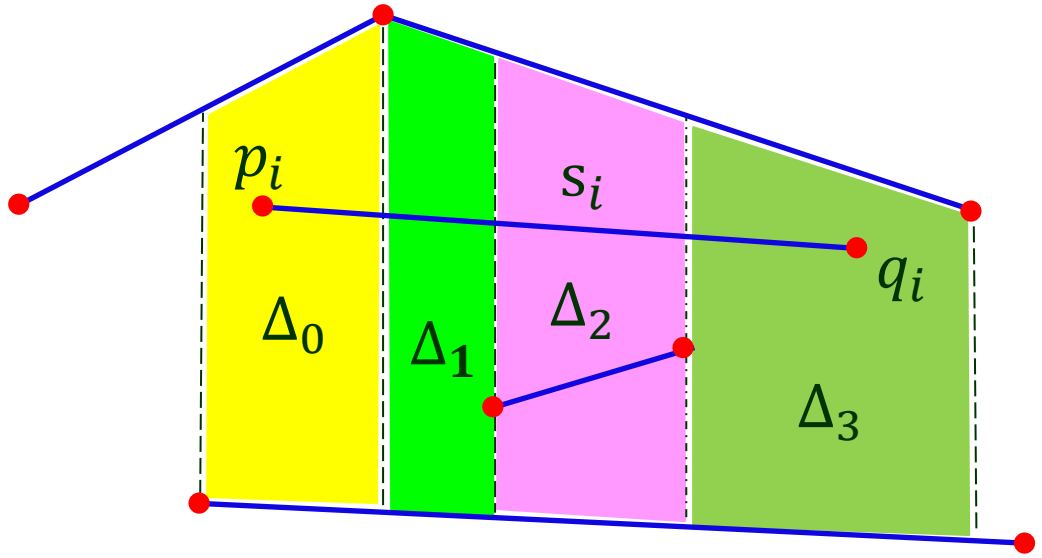
Loop invariants: a) T is the trapezoidal map for S_i;
b) D is a search structure for T.

Algorithm

1. Determine the bounding box R.
2. Initialize the trapezoidal map $T(S)$.
3. Initialize the search structure D.
4. Compute a random permutation $s_1, s_2, \ldots, s_n$ of segments.
5. for $i \leftarrow 1$ to n
6. do Find trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$ properly intersected by s_i.
7. Replace with new trapezoids due to intersection by s_i.
8. Remove leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from D.
9. Create the leaves for the new trapezoids.

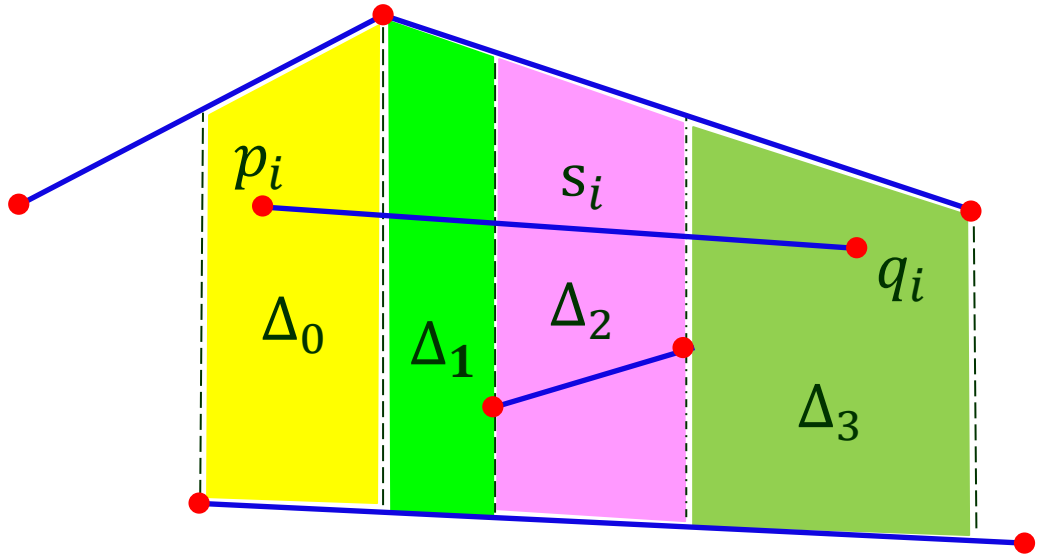
II. Trapezoidal Map Modification

How to add s_i?



II. Trapezoidal Map Modification

How to add s_i?



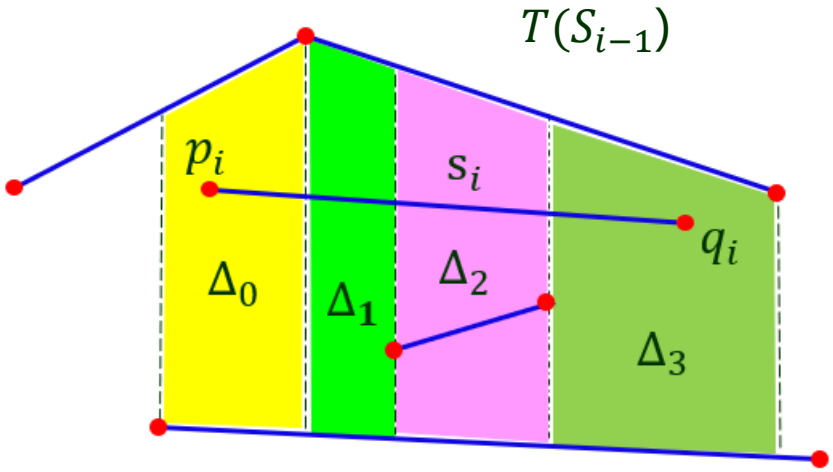
For every trapezoid $\Delta \in T(S_{i-1})$

$\Delta \notin T(S_i)$ iff Δ is intersected by s_i.

Intersected Trapezoids

Find trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$ intersected by s_i

- Locate Δ_0.

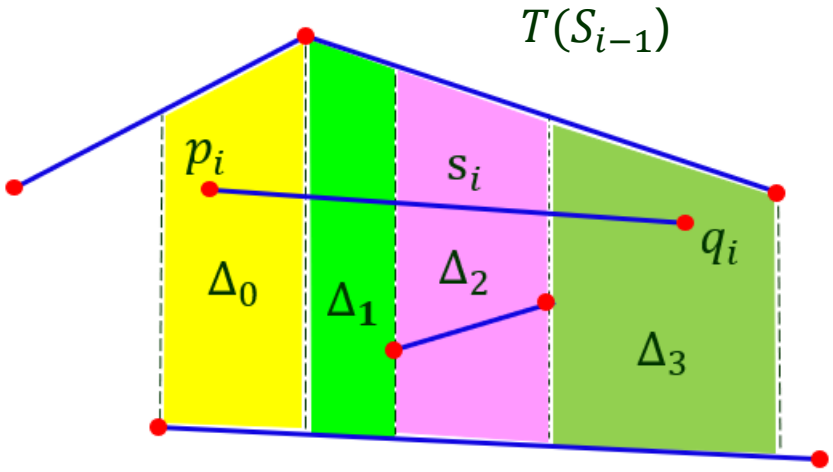


Intersected Trapezoids

Find trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$ intersected by s_i

- Locate Δ_0.

  Point location problem with left endpoint p_i of s_i in $T(S_{i-1})$.



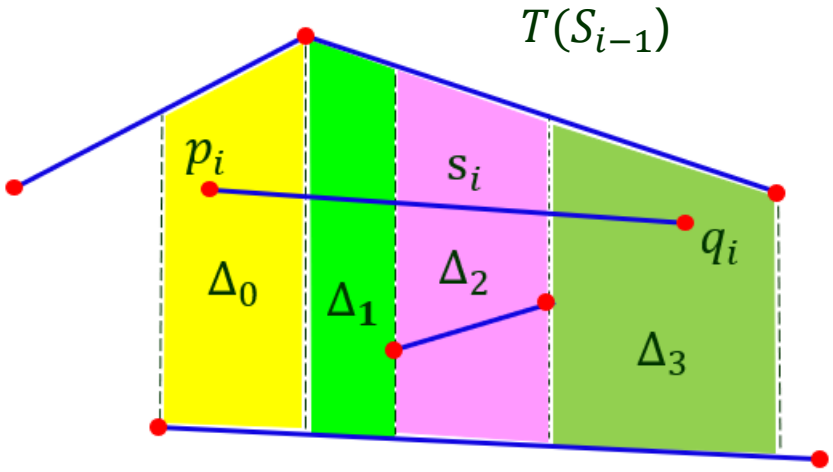
Intersected Trapezoids

Find trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$ intersected by s_i

- Locate Δ_0.

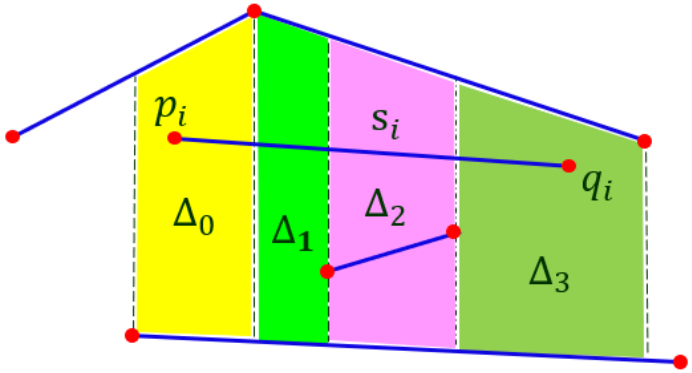
  Point location problem with left endpoint p_i of s_i in $T(S_{i-1})$.

  Use the current search structure $D(S_{i-1})$ to perform query with p_i.



The Right Neighbor

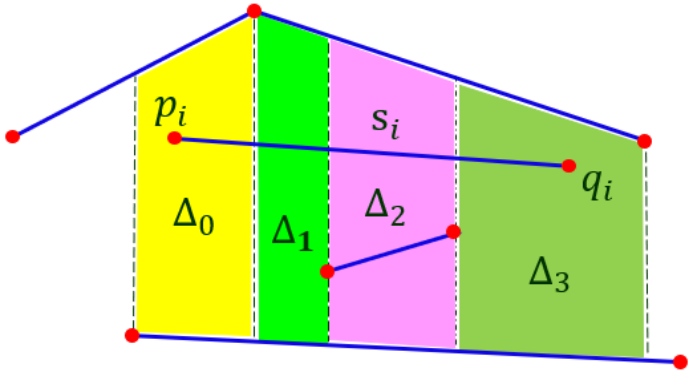
- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k



The Right Neighbor

- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

  Δ_{j+1} is a right neighbor of Δ_j.

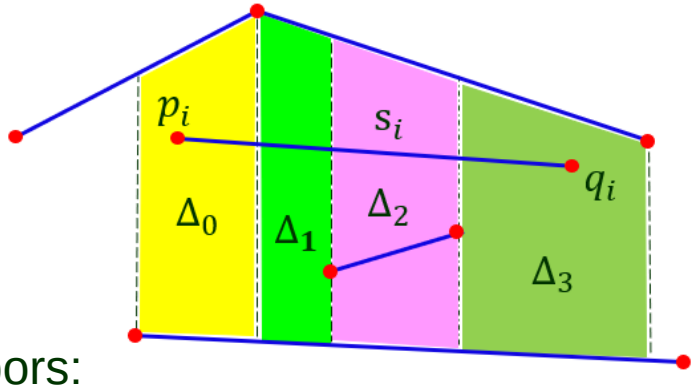


The Right Neighbor

- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

  Δ_{j+1} is a right neighbor of Δ_j.

  - In the case that there are two right neighbors:



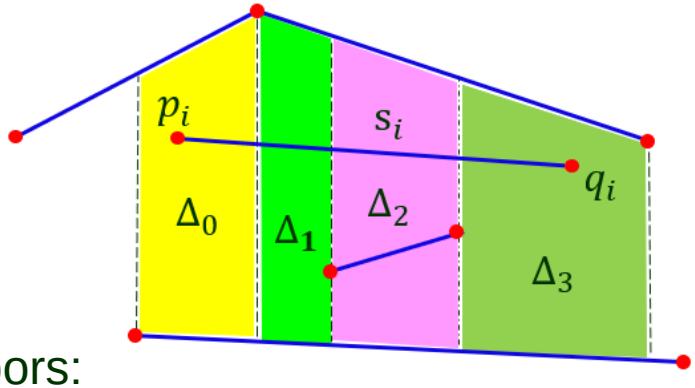
The Right Neighbor

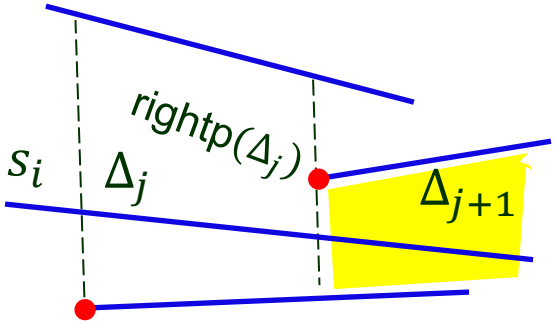
- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

  Δ_{j+1} is a right neighbor of Δ_j.

  - In the case that there are two right neighbors:

    - s_i below rightp(Δ_j)





The Right Neighbor

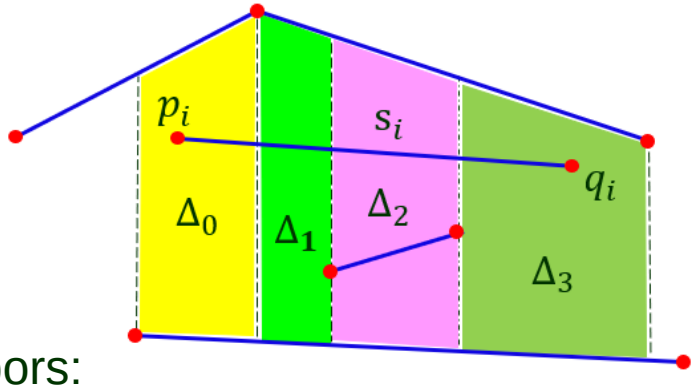
- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

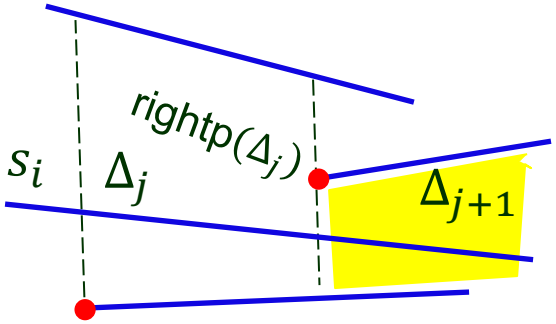
  Δ_{j+1} is a right neighbor of Δ_j.

  - In the case that there are two right neighbors:

    - s_i below rightp(Δ_j)

      $\Rightarrow \Delta_{j+1}$ is lower right neighbor.





The Right Neighbor

- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

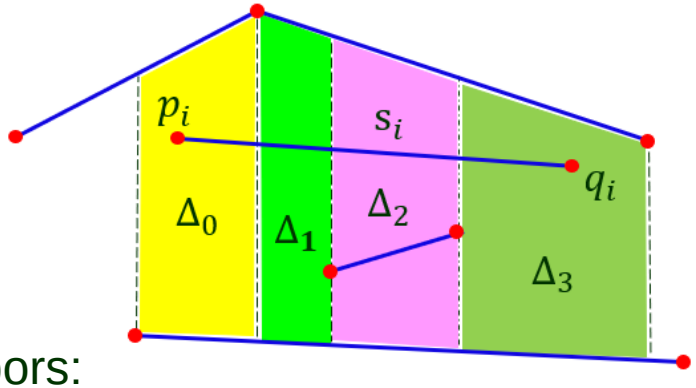
  Δ_{j+1} is a right neighbor of Δ_j.

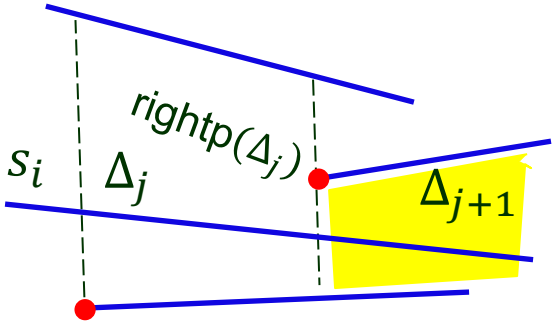
  - In the case that there are two right neighbors:

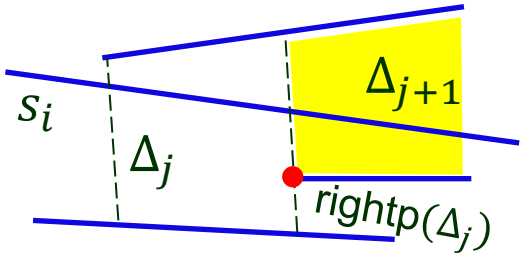
    - s_i below rightp(Δ_j)

      $\Rightarrow \Delta_{j+1}$ is lower right neighbor.

    - s_i above rightp(Δ_j)







The Right Neighbor

- $\Delta_0 \Rightarrow$ locate $\Delta_1 \Rightarrow \cdots \Rightarrow$ locate Δ_k

  Δ_{j+1} is a right neighbor of Δ_j.

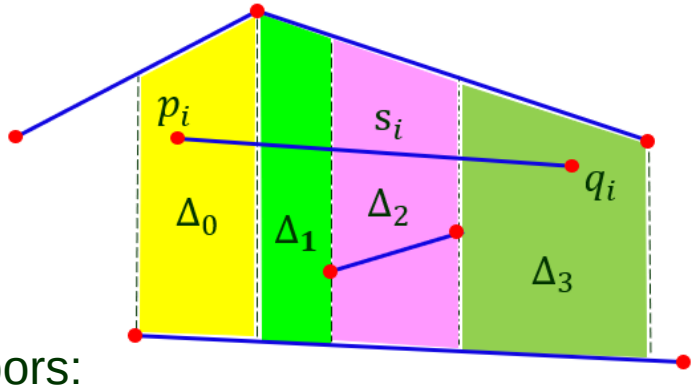
  - In the case that there are two right neighbors:

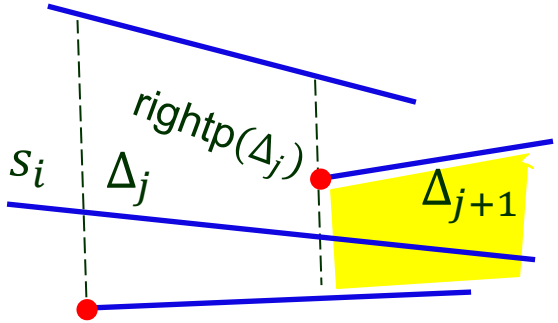
    - s_i below rightp(Δ_j)

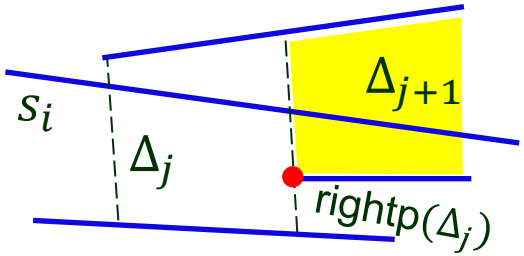
      $\Rightarrow \Delta_{j+1}$ is lower right neighbor.

    - s_i above rightp(Δ_j)

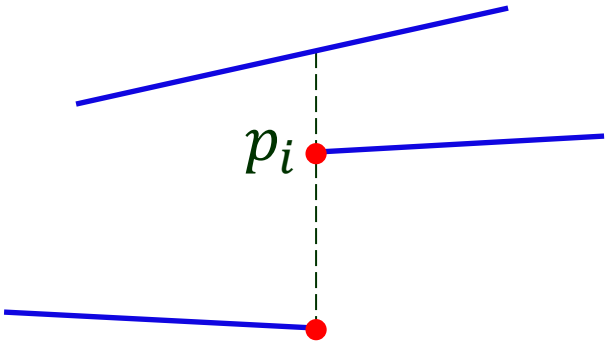
      $\Rightarrow \Delta_{j+1}$ is upper right neighbor.





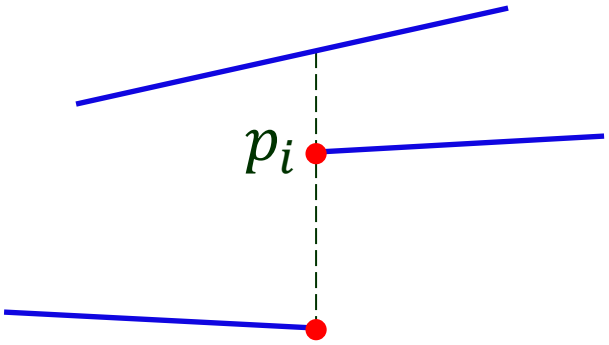


Two Degeneracies



- p_i lies on the vertical line of an x-node in $T(S_{i-1})$.

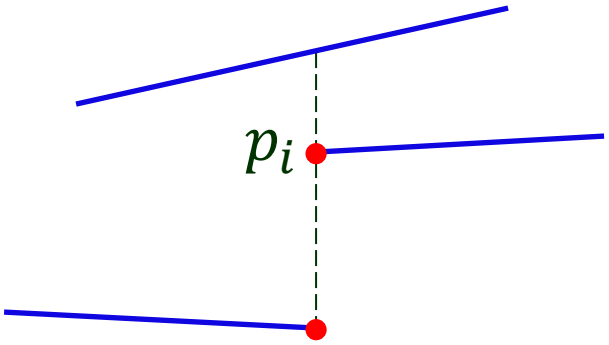
Two Degeneracies



- p_i lies on the vertical line of an x-node in $T(S_{i-1})$.

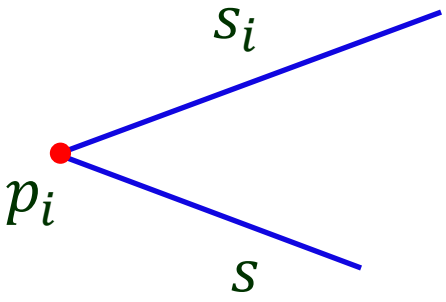
Treat it as lying "to the right" of the x-node.

Two Degeneracies



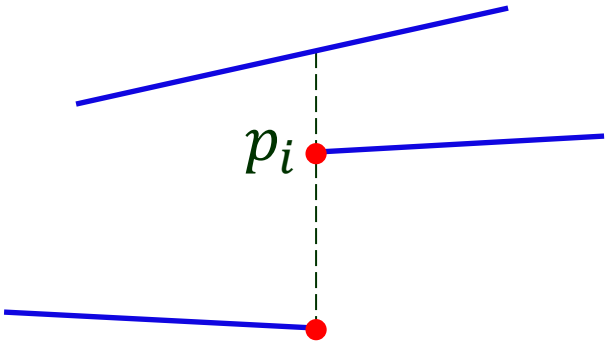
- p_i lies on the vertical line of an x-node in $T(S_{i-1})$.

Treat it as lying "to the right" of the x-node.



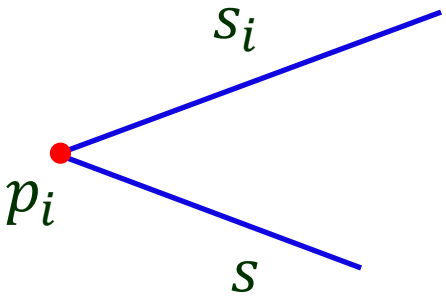
- p_i is the left endpoint of a segment s of a y-node.

Two Degeneracies



- p_i lies on the vertical line of an x-node in $T(S_{i-1})$.

Treat it as lying "to the right" of the x-node.



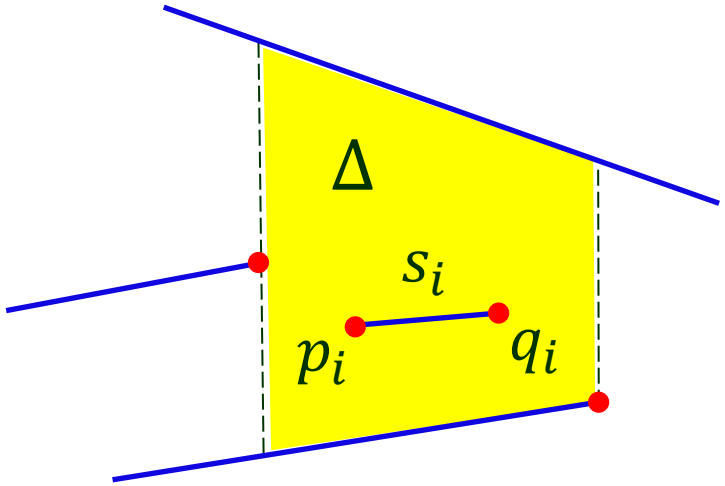
- p_i is the left endpoint of a segment s of a y-node.

Compare the slopes of s and s_i to determine "above" or "below".

III. Updating T and D (Simple Case)

s_i contained in Δ_0

- Non-degenerate situation: 4 new trapezoids.

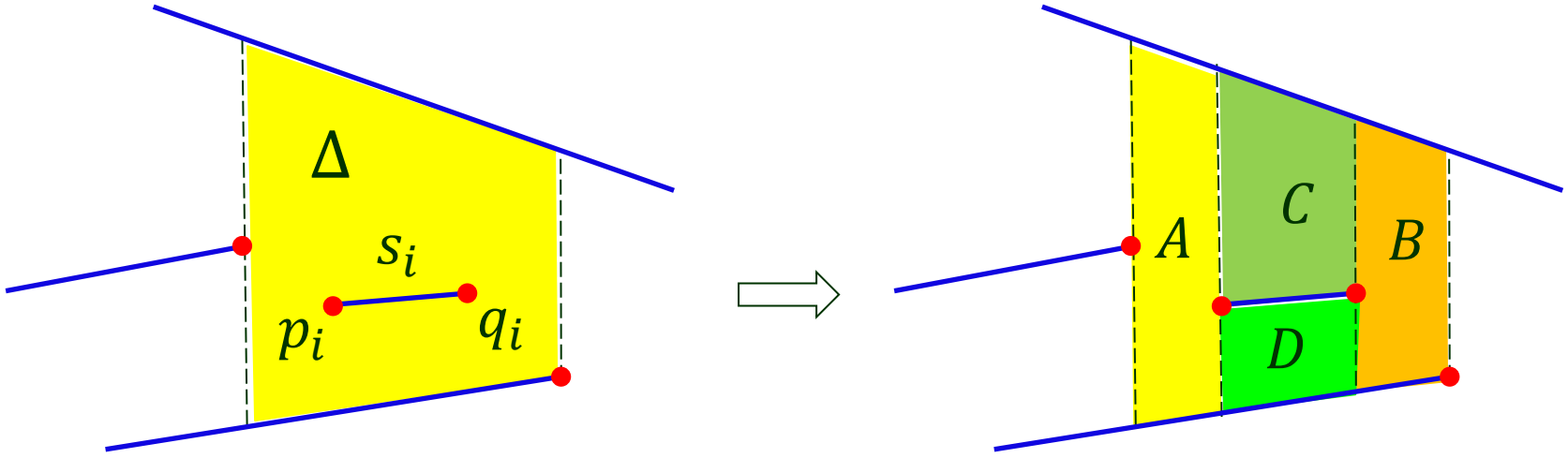


$T(S_{i-1})$

III. Updating T and D (Simple Case)

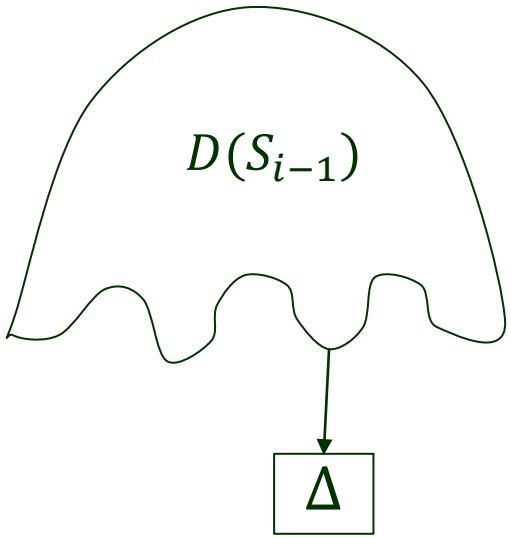
s_i contained in Δ_0

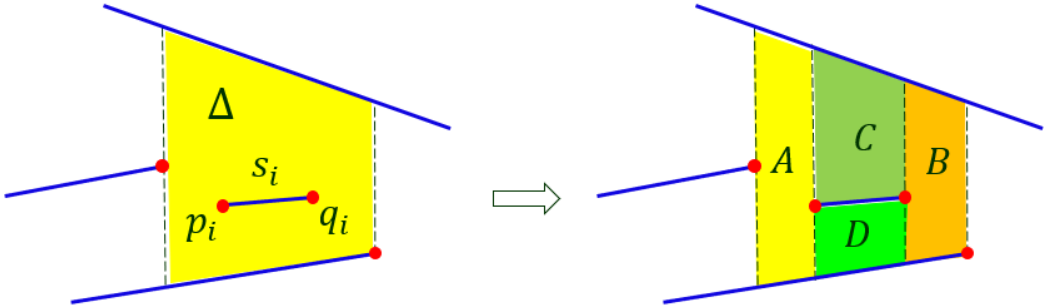
- Non-degenerate situation: 4 new trapezoids.



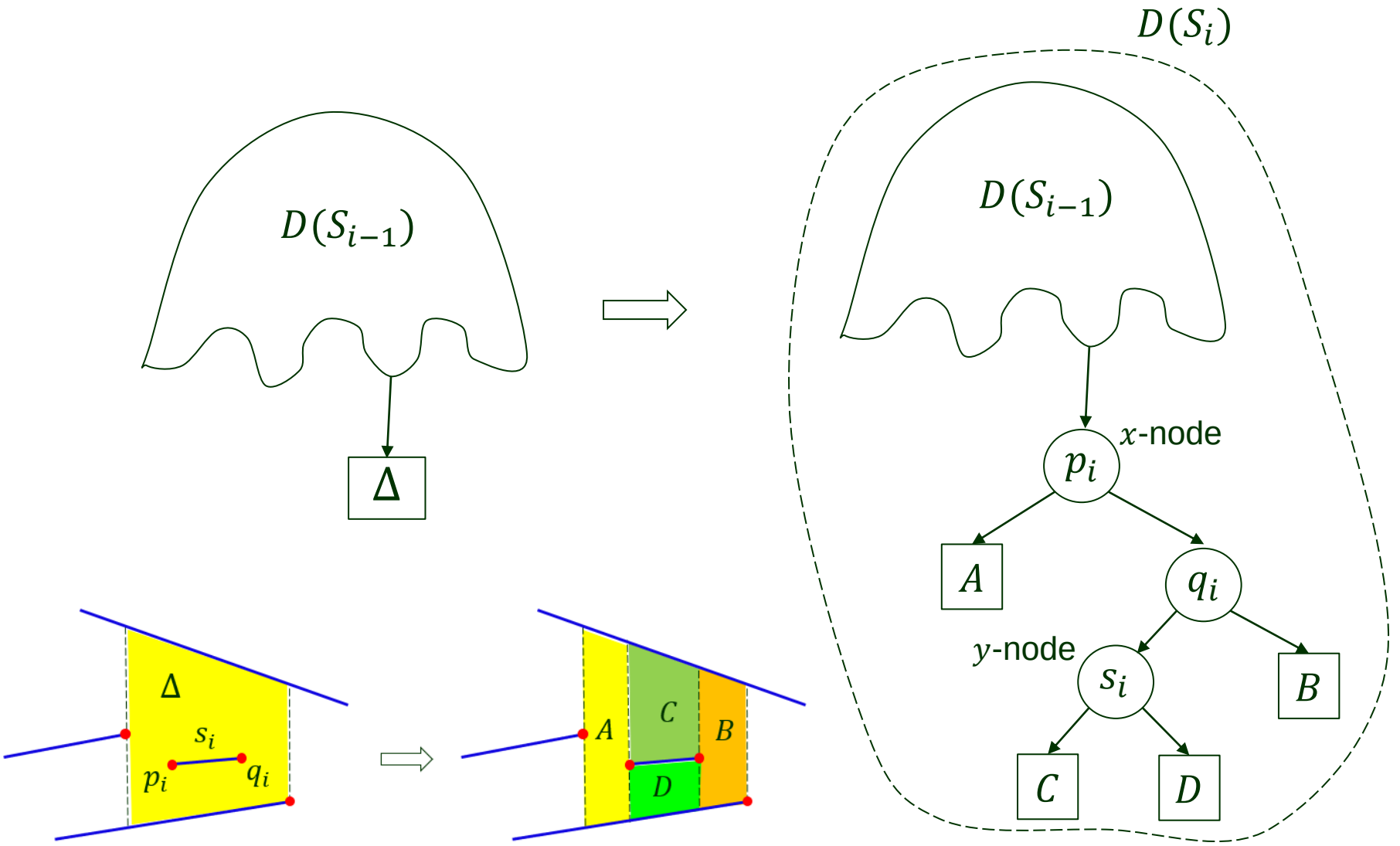
$T(S_{i-1})$ $T(S_i)$

Simple Update (cont'd)

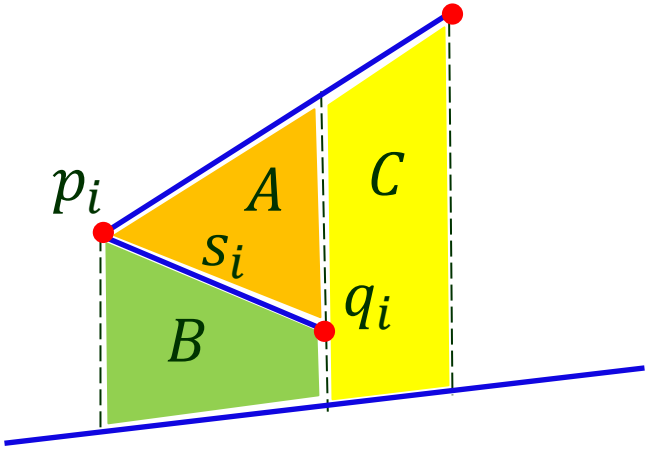




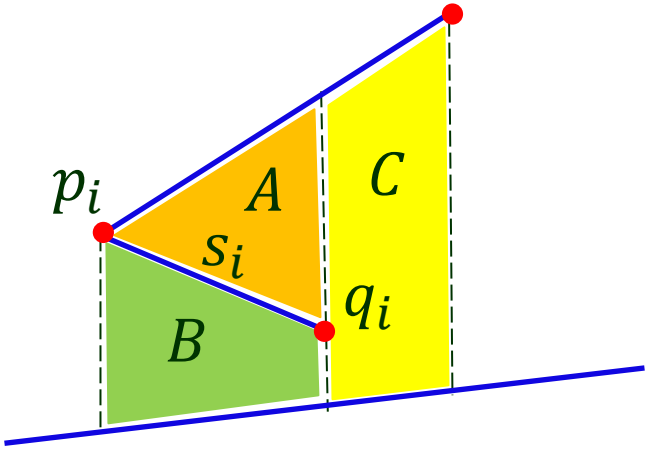
Simple Update (cont'd)

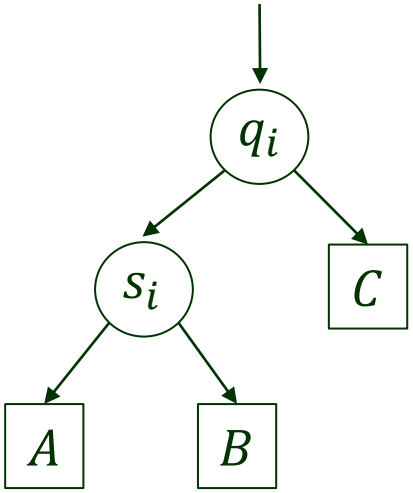


Degenerate Situations

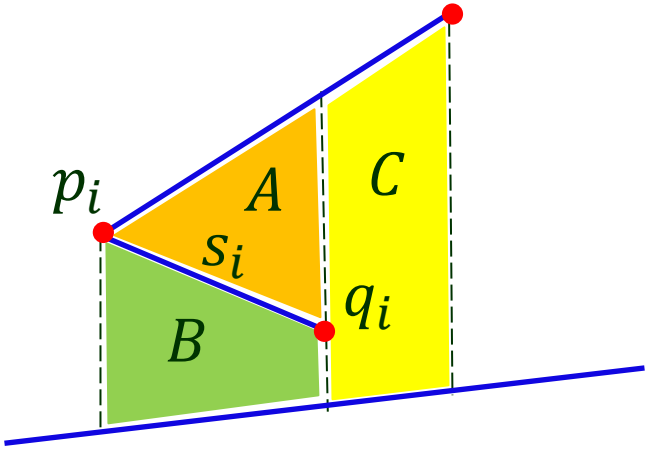


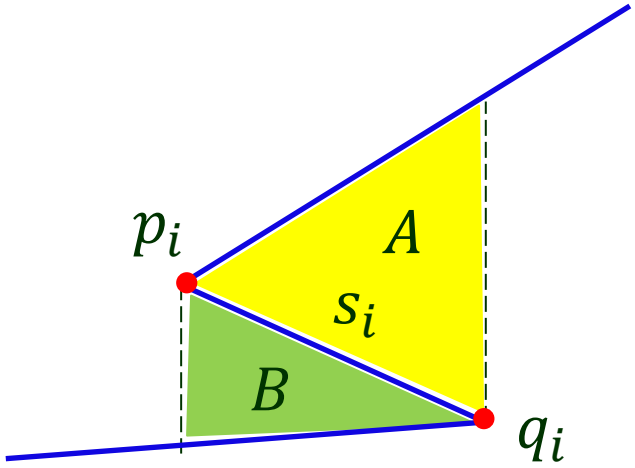
Degenerate Situations

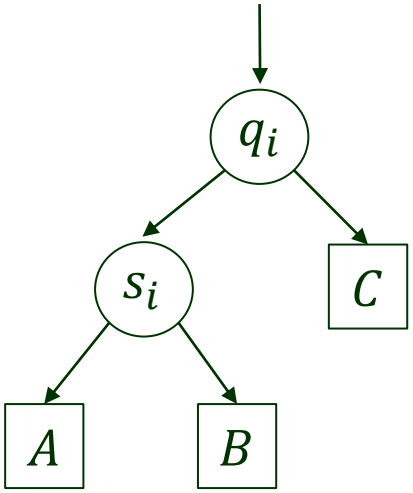




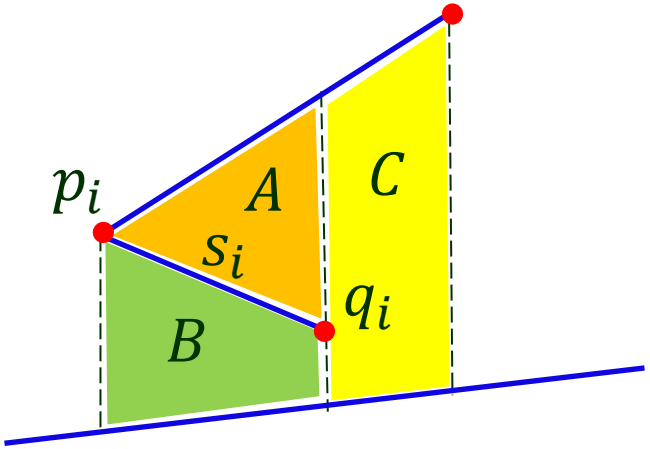
Degenerate Situations

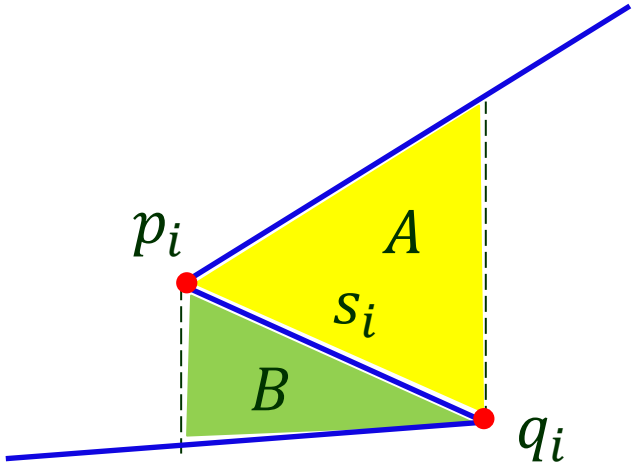


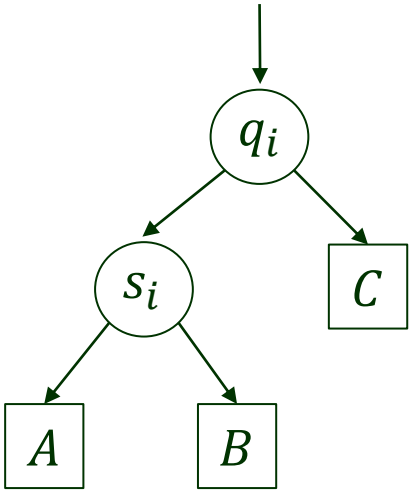


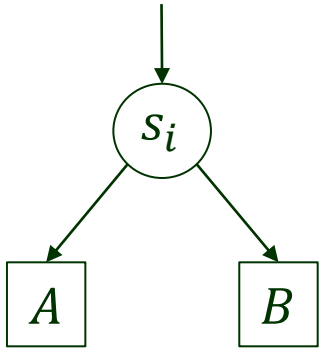


Degenerate Situations



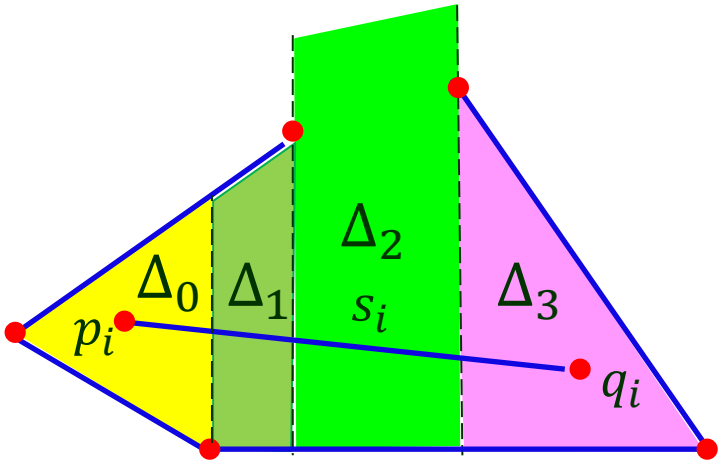






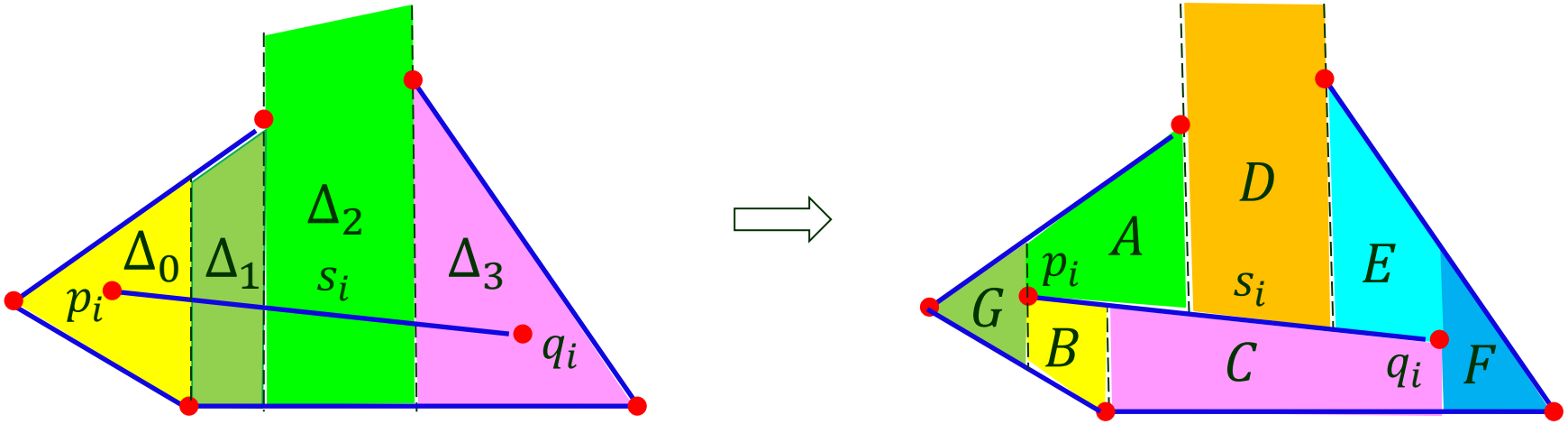
Updating T and D (General Case)

s_i intersects ≥ 2 trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$.

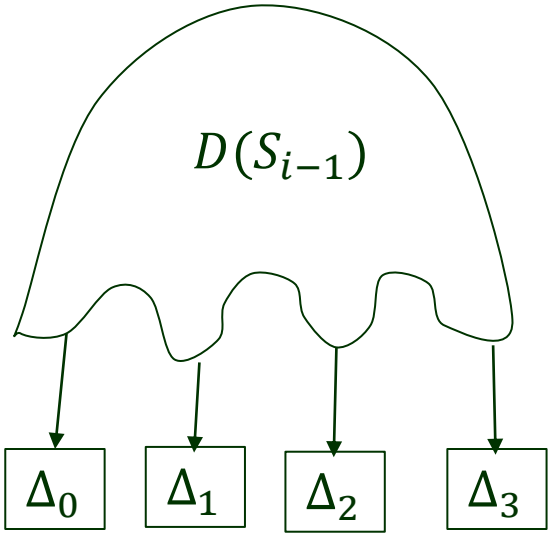


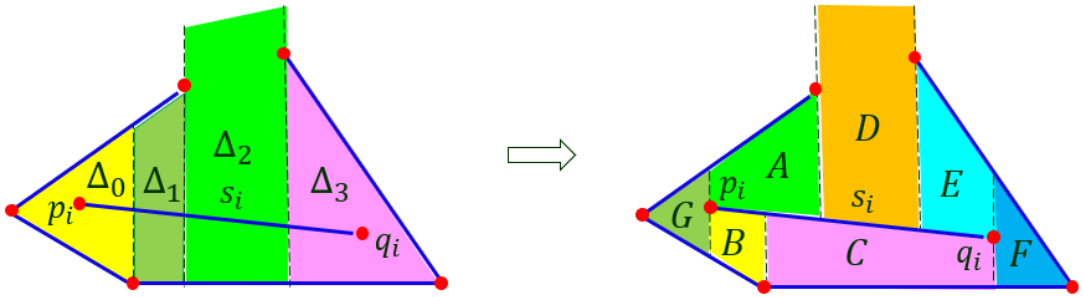
Updating T and D (General Case)

s_i intersects ≥ 2 trapezoids $\Delta_0, \Delta_1, \ldots, \Delta_k$.

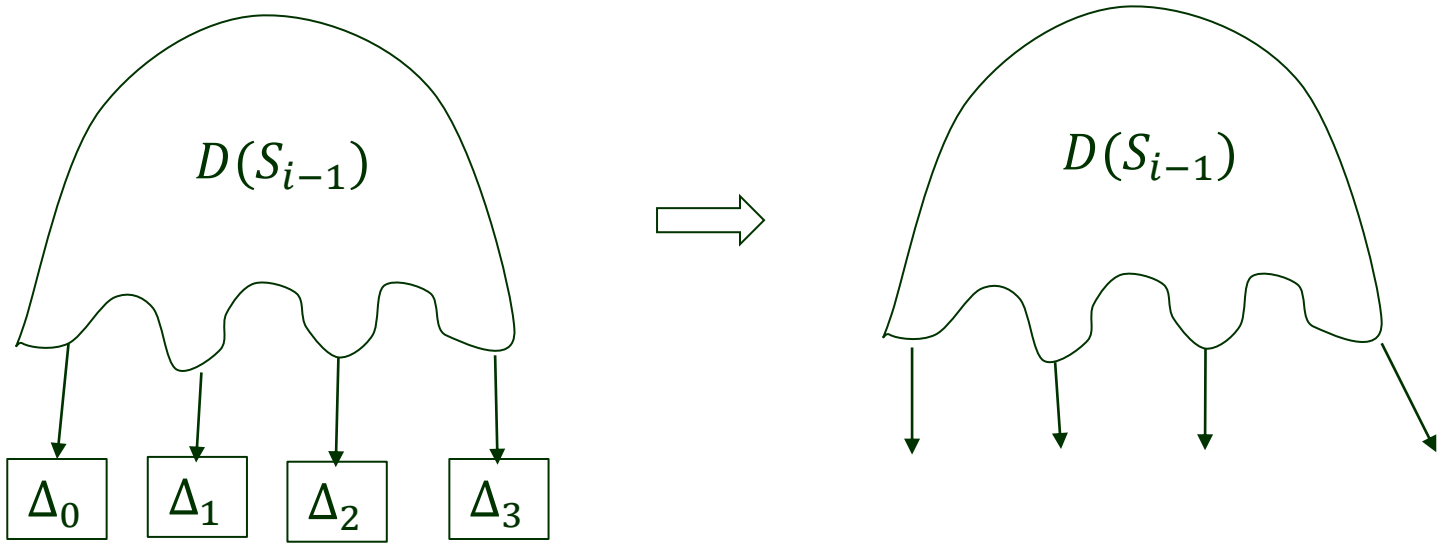


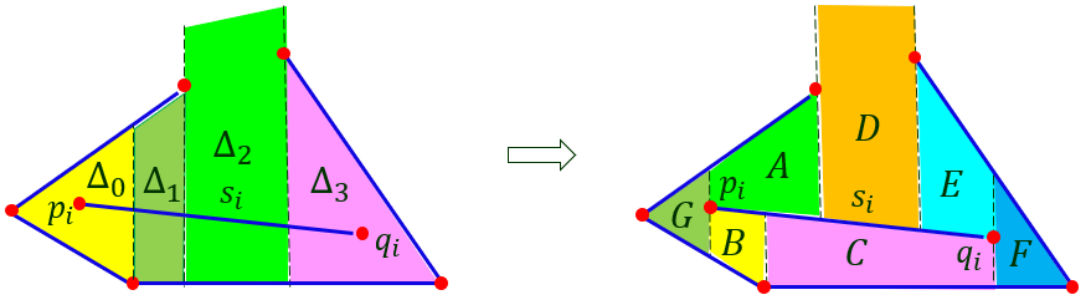
General Update (cont'd)



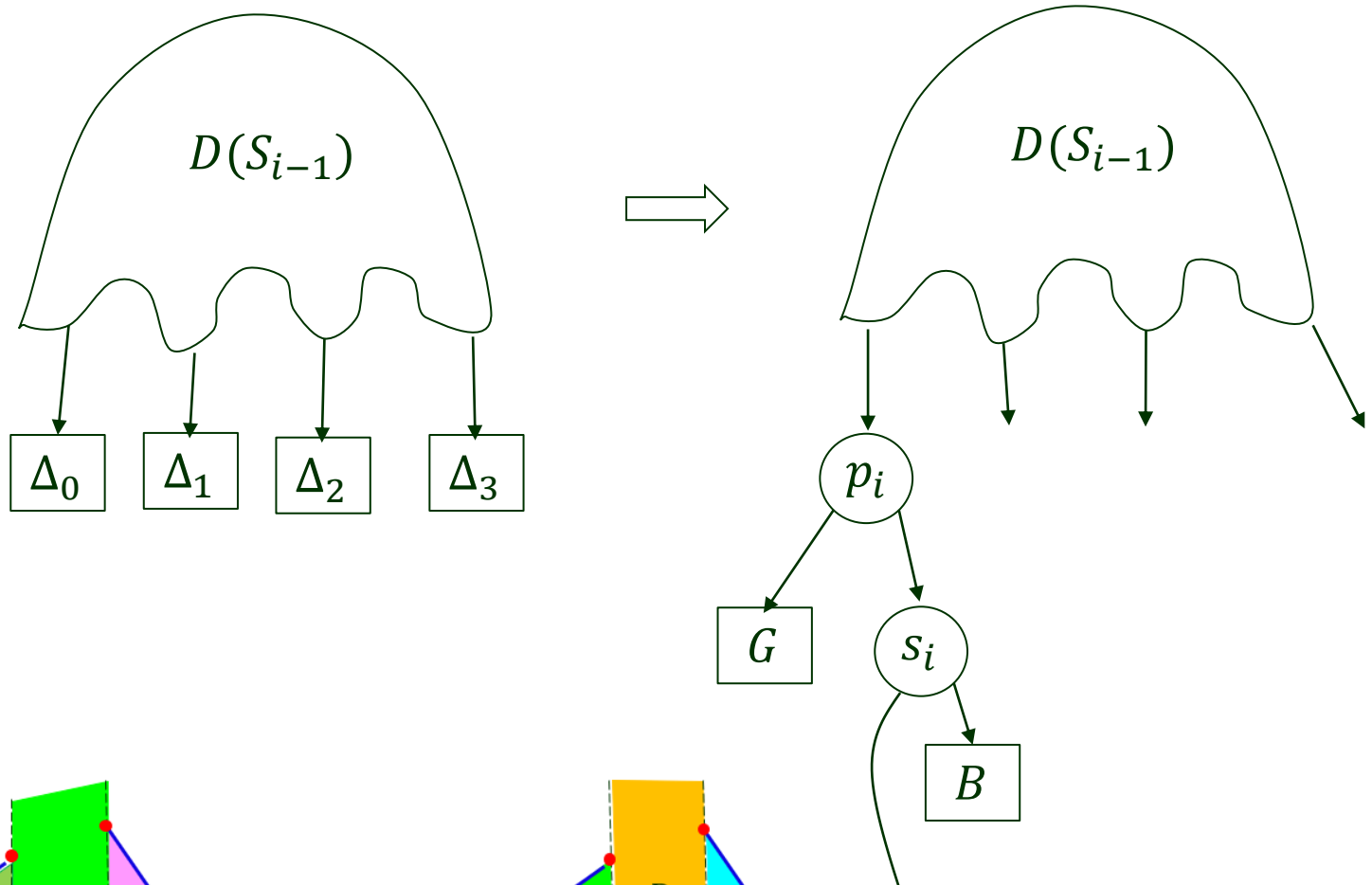


General Update (cont'd)

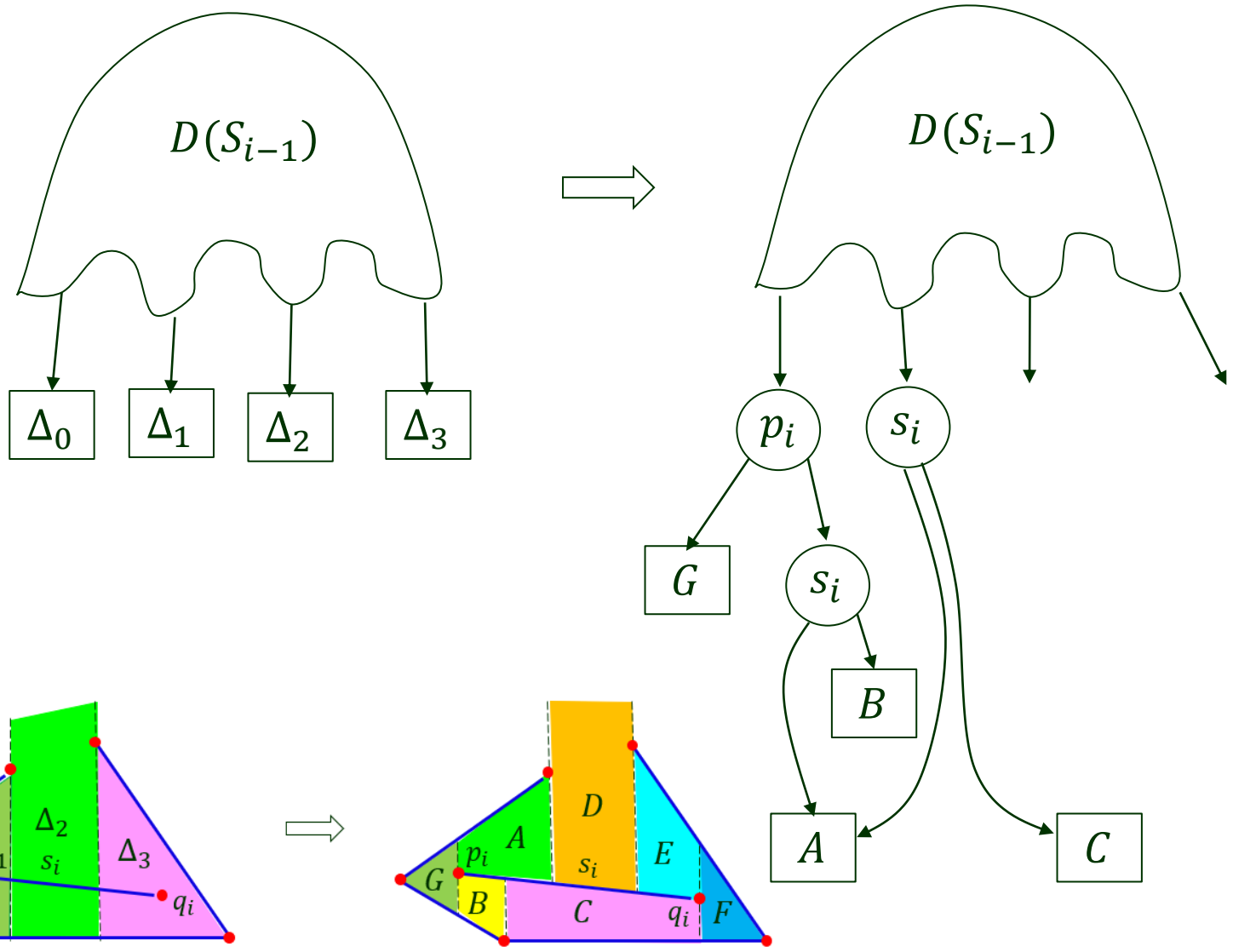




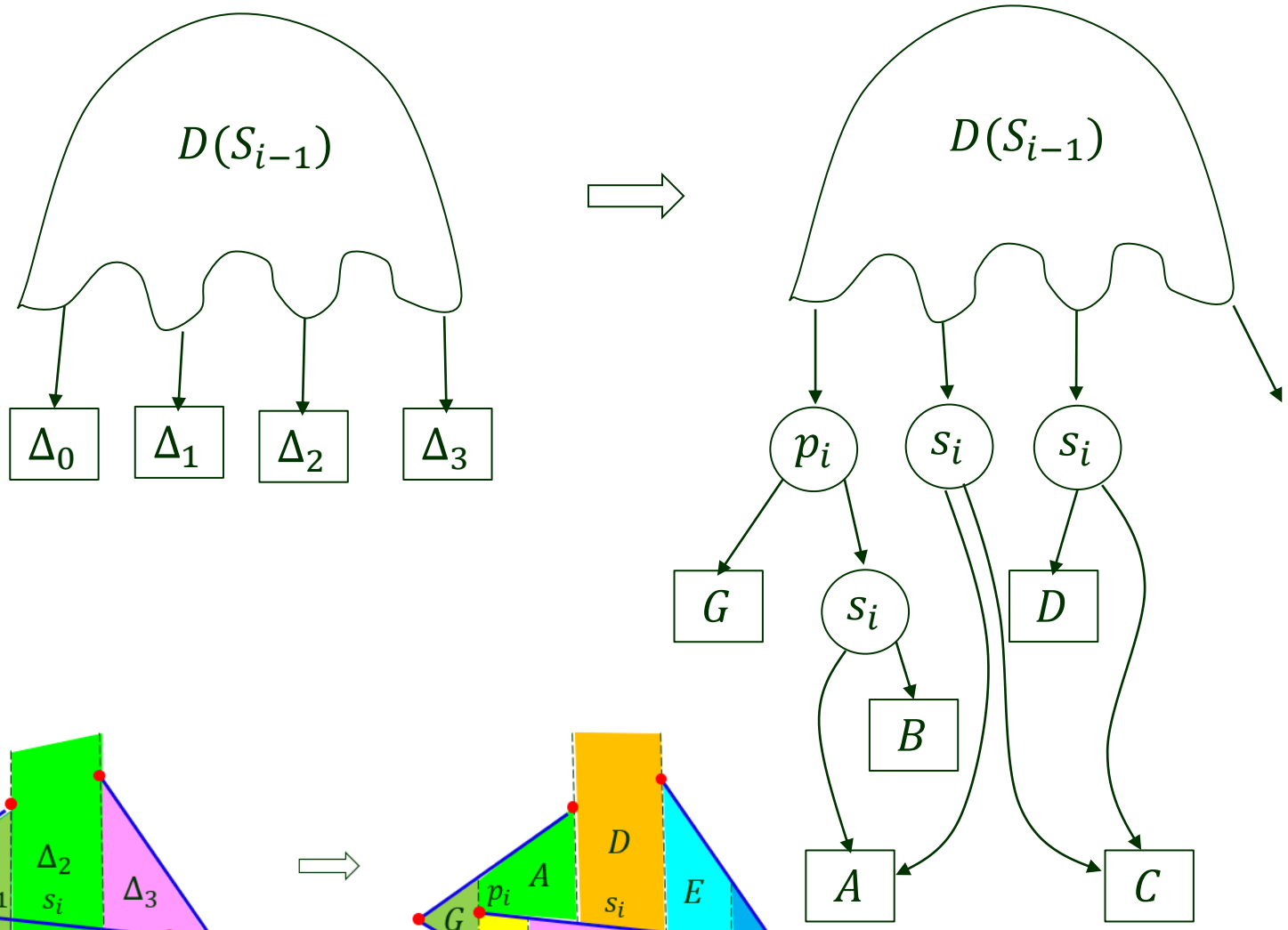
General Update (cont'd)



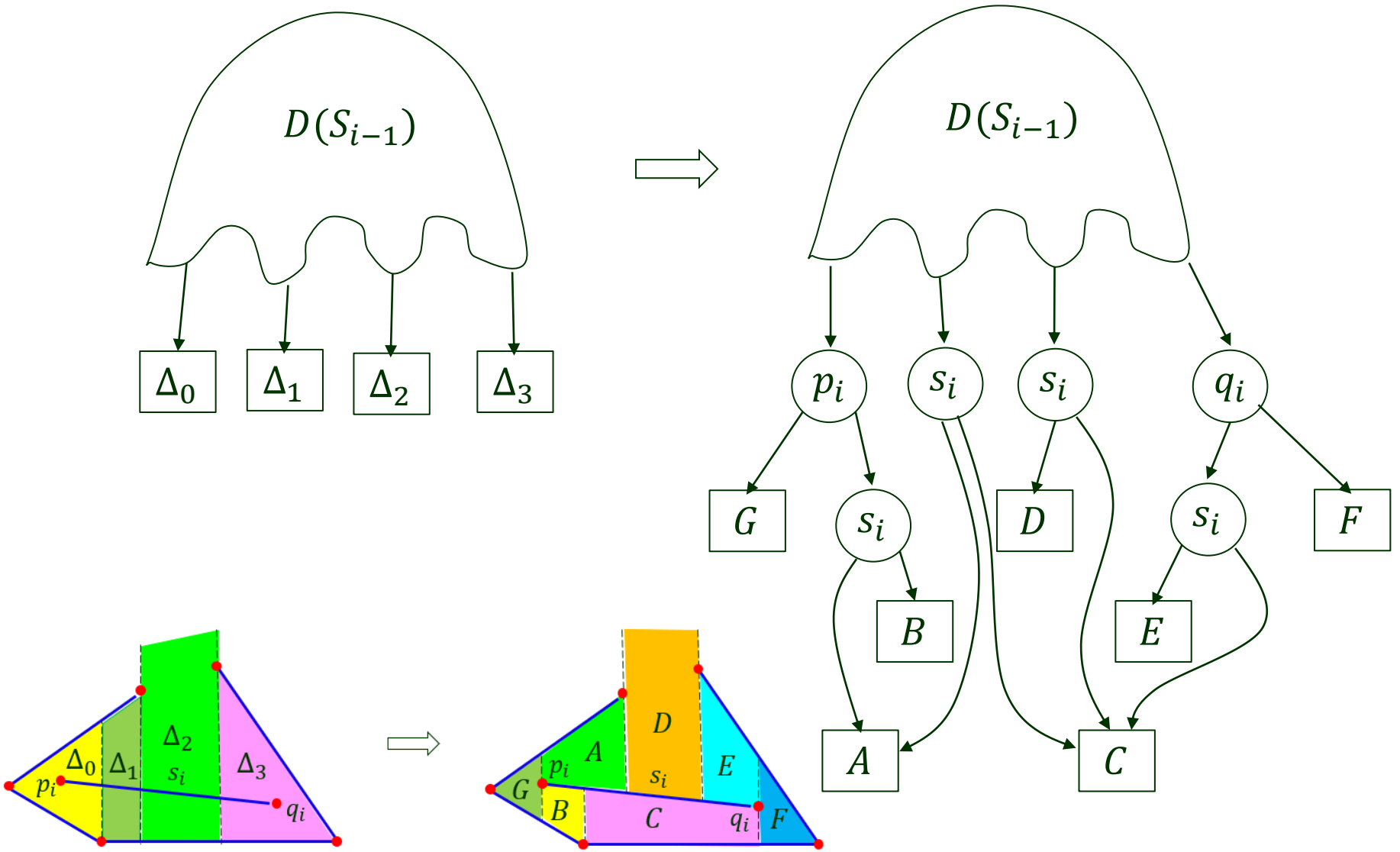
General Update (cont'd)



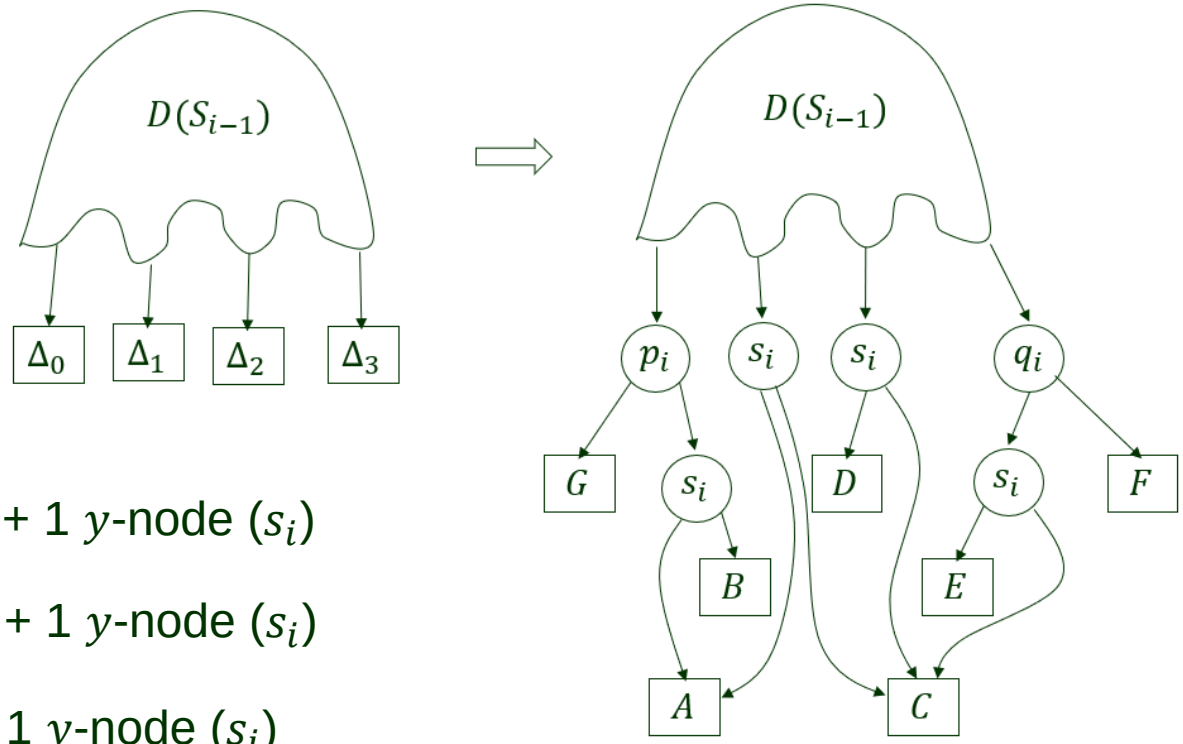
General Update (cont'd)



General Update (cont'd)



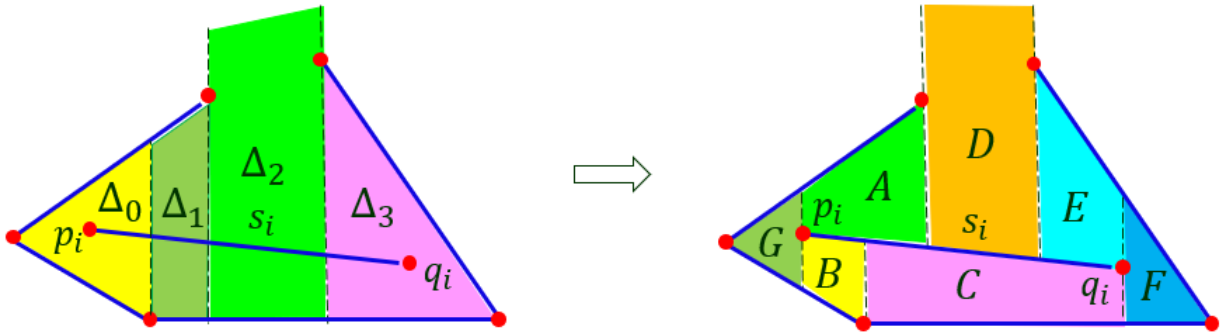
Search Structure Update



Δ_0: replaced with 1 x-node (p_i) + 1 y-node (s_i)

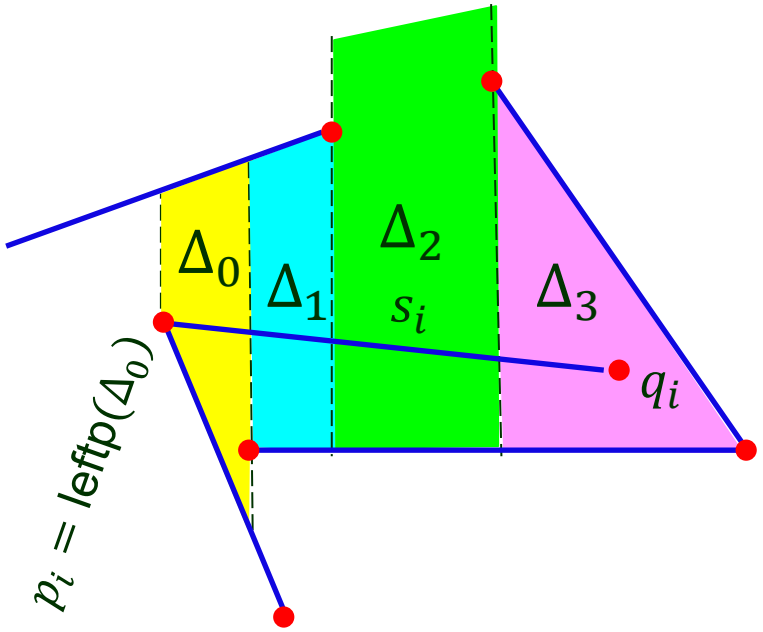
Δ_k: replaced with 1 x-node (q_i) + 1 y-node (s_i)

$\Delta_j, 1 \leq j \leq k-1$: replaced with 1 y-node (s_i)



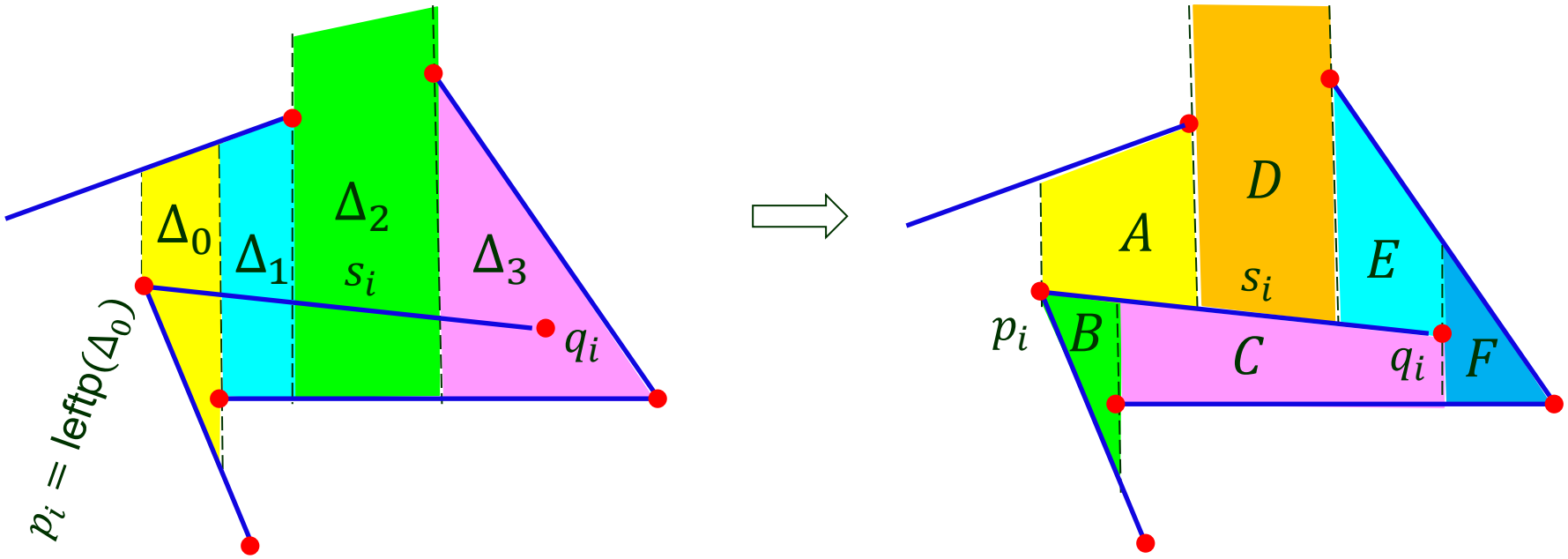
Degeneracies

- p_i coincides with leftp(Δ_0).



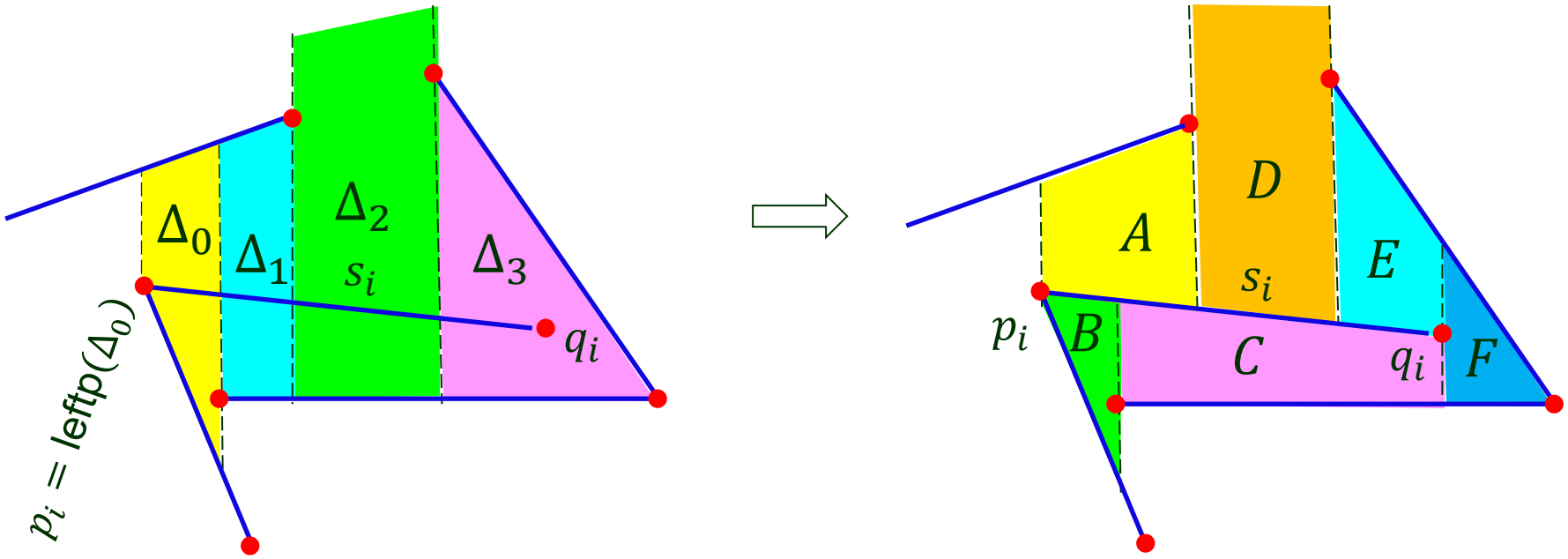
Degeneracies

♣ p_i coincides with leftp(Δ_0).



Degeneracies

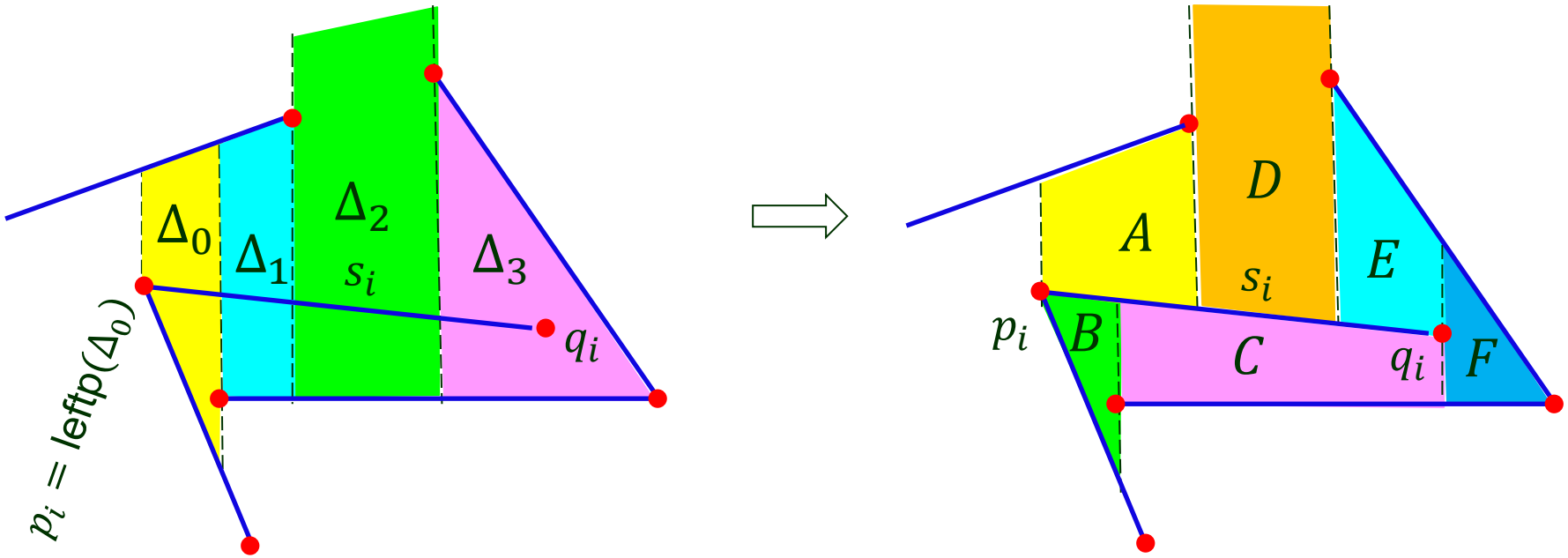
♣ p_i coincides with leftp(Δ_0).



Δ_0: replaced with 1 y-node (s_i)

Degeneracies

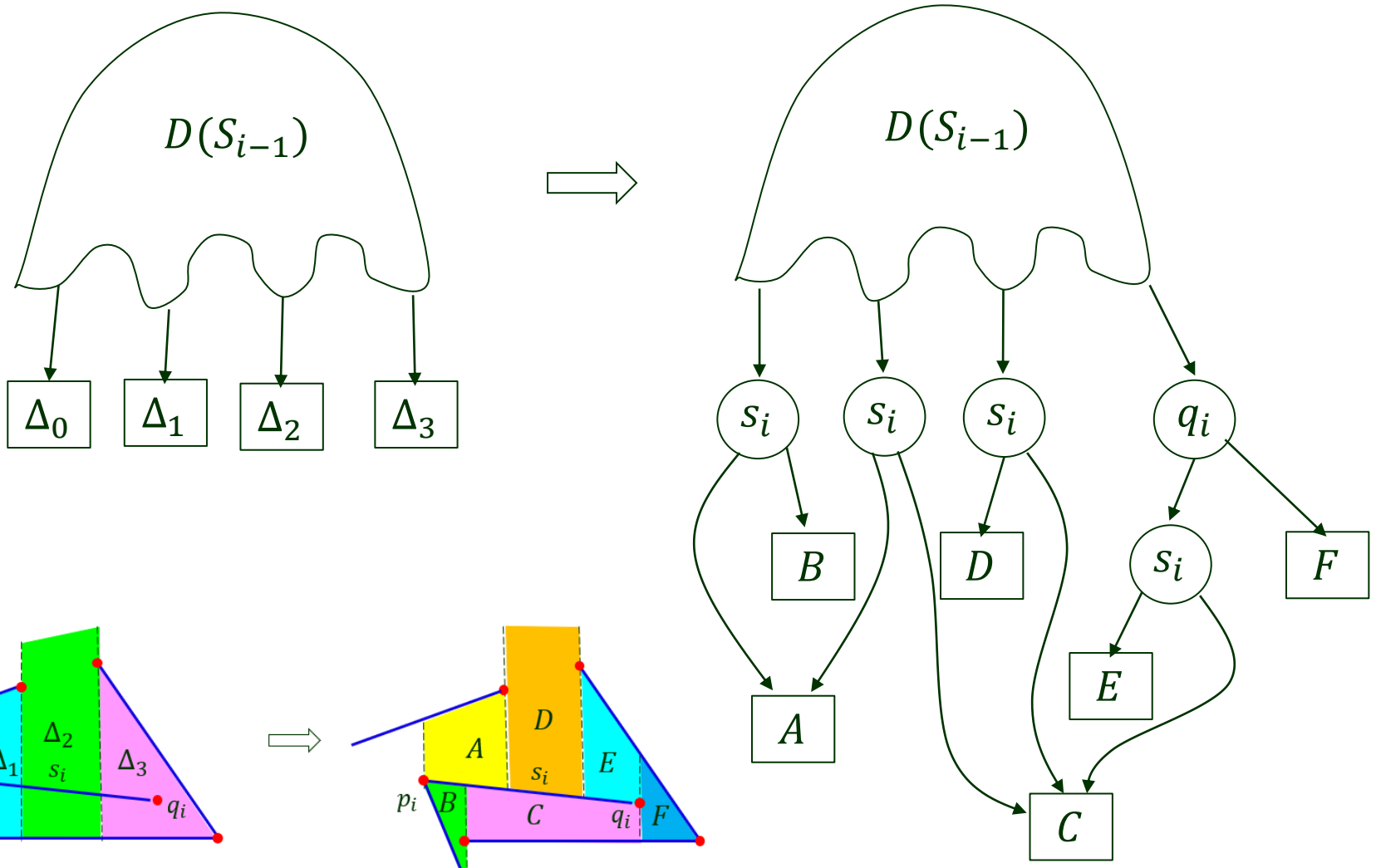
♣ p_i coincides with leftp(Δ_0).



Δ_0: replaced with 1 y-node (s_i)

♣ q_i coincides with rightp(Δ_k) – similarly handled.

Degeneracies (cont'd)



IV. Expected Query Time

Let q be a query point (fixed).

IV. Expected Query Time

Let q be a query point (fixed).

- query time $\backsim$ length of path γ traversed in D
- Bound the length of γ.

IV. Expected Query Time

Let q be a query point (fixed).

- query time $\backsim$ length of path γ traversed in D

- Bound the length of γ.

  Every node on γ was created in some iteration during construction.

IV. Expected Query Time

Let q be a query point (fixed).

- query time $\backsim$ length of path γ traversed in D

- Bound the length of γ.

  Every node on γ was created in some iteration during construction.

  $X_i, 1 \leq i \leq n$: # nodes on P created in iteration i.

IV. Expected Query Time

Let q be a query point (fixed).

- query time $\backsim$ length of path γ traversed in D

- Bound the length of γ.

Every node on γ was created in some iteration during construction.

$X_i, 1 \leq i \leq n$: # nodes on P created in iteration i.

random variable depending on the
random order of the segments

IV. Expected Query Time

Let q be a query point (fixed).

- query time $\backsim$ length of path γ traversed in D

- Bound the length of γ.

  Every node on γ was created in some iteration during construction.

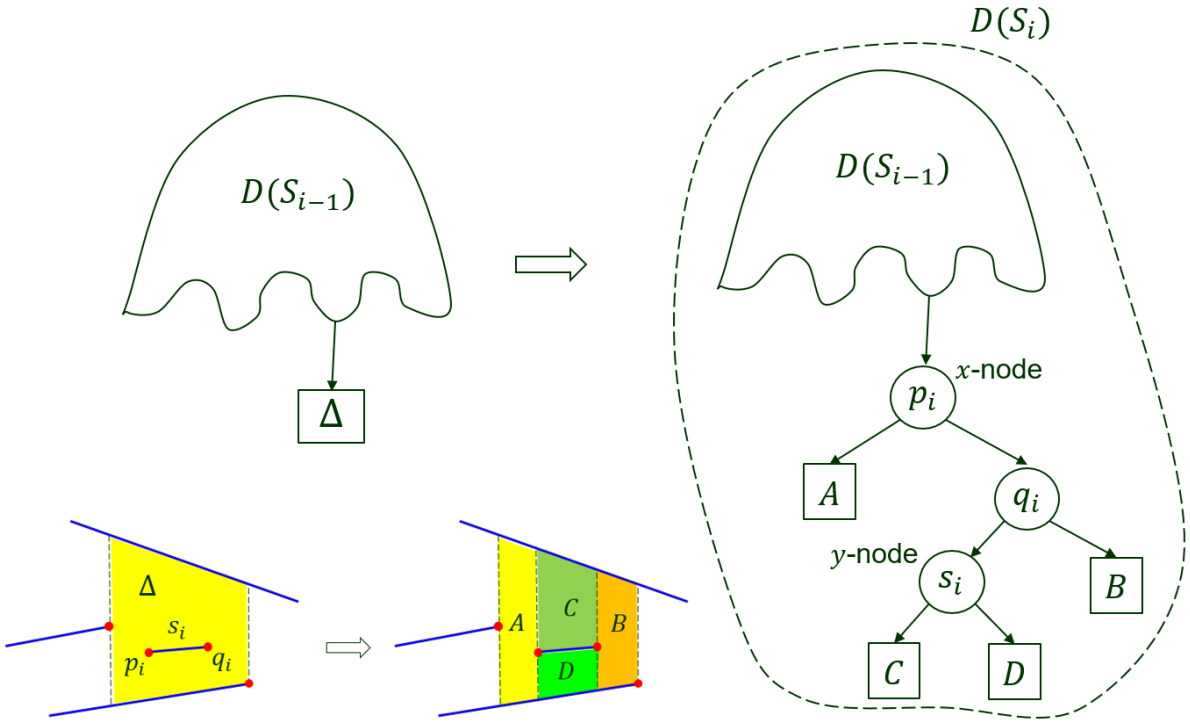
  $X_i, 1 \leq i \leq n$: # nodes on P created in iteration i.

  random variable depending on the random order of the segments

Expected path length: $E\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} E(X_i)$

Nodes Added in One Iteration

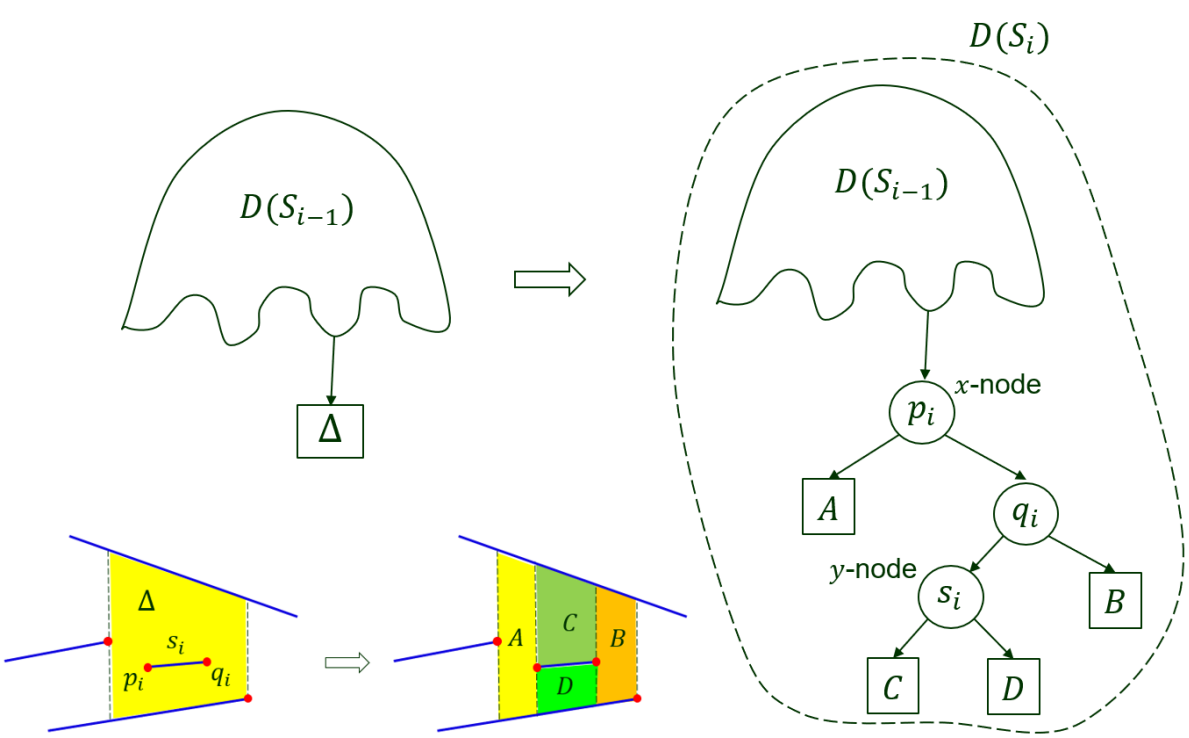
Fact Any iteration adds ≤ 3 nodes to the search path γ.



The case where 3 nodes are added along a path.

Nodes Added in One Iteration

Fact Any iteration adds ≤ 3 nodes to the search path γ.

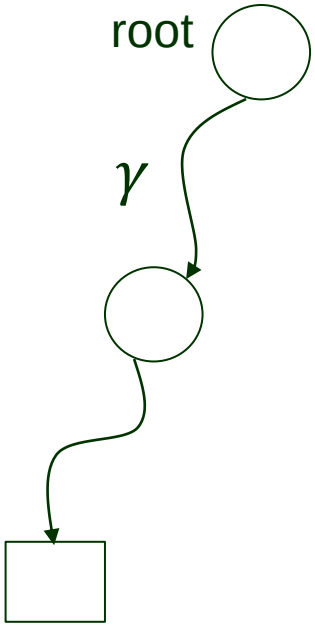


$$X_i \leq 3$$

The case where 3 nodes are added along a path.

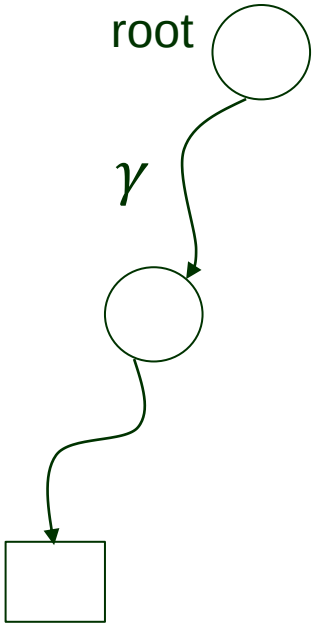
Probability of Node Creation

P_i: probability that there is a node on the search path γ of the query point q which is created in iteration i.



Probability of Node Creation

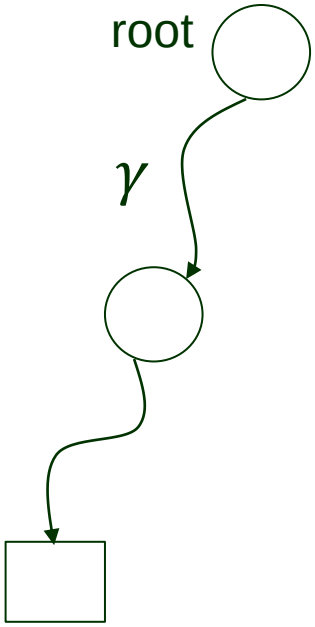
P_i: probability that there is a node on the search path γ of the query point q which is created in iteration i.



$$E(X_i) \leq 3P_i$$

Probability of Node Creation

P_i: probability that there is a node on the search path γ of the query point q which is created in iteration i.



$$E(X_i) \leq 3P_i$$

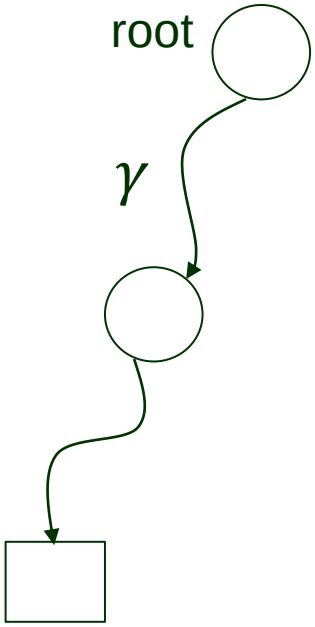
- Iteration i contributes at least one node to γ iff

$$\underbrace{\Delta_q(S_i)}_{\text{trapezoid containing } q \text{ in } T(S_i)} \neq \underbrace{\Delta_q(S_{i-1})}_{\text{trapezoid containing } q \text{ in } T(S_{i-1})}$$

where $S_i = \{s_1, s_2, \cdots, s_i\}$

Probability of Node Creation

P_i: probability that there is a node on the search path γ of the query point q which is created in iteration i.



$$E(X_i) \leq 3P_i$$

- Iteration i contributes at least one node to γ iff

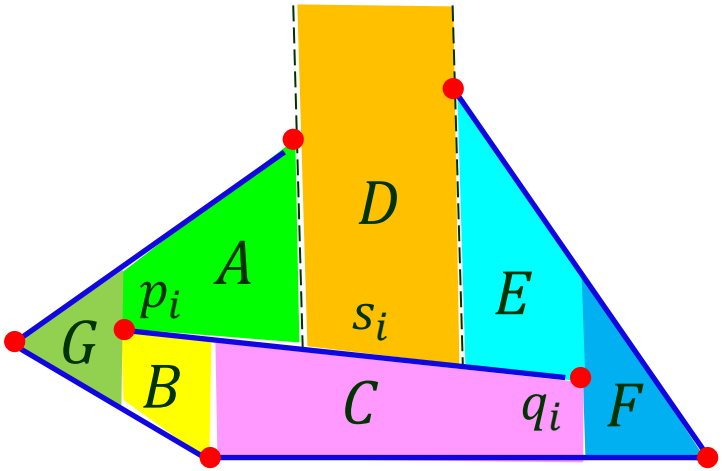
$$\underbrace{\Delta_q(S_i)}_{\text{trapezoid containing } q \text{ in } T(S_i)} \neq \underbrace{\Delta_q(S_{i-1})}_{\text{trapezoid containing } q \text{ in } T(S_{i-1})}$$

where $S_i = \{s_1, s_2, \cdots, s_i\}$

$$P_i = \Pr(\Delta_q(S_i) \neq \Delta_q(S_{i-1}))$$

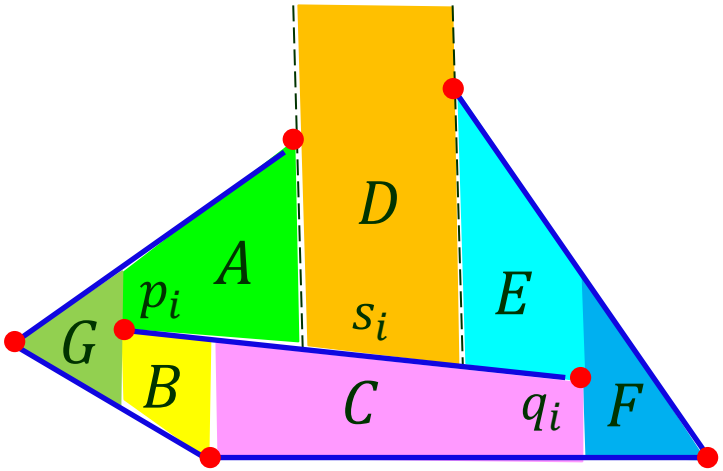
Cont'd

$$\Delta_q(S_i) \neq \Delta_q(S_{i-1})$$



Cont'd

$\Delta_q(S_i) \neq \Delta_q(S_{i-1}) \implies \Delta_q(S_i)$ is created in iteration i



Cont'd

$\Delta_q(S_i) \neq \Delta_q(S_{i-1}) \implies \Delta_q(S_i)$ is created in iteration i

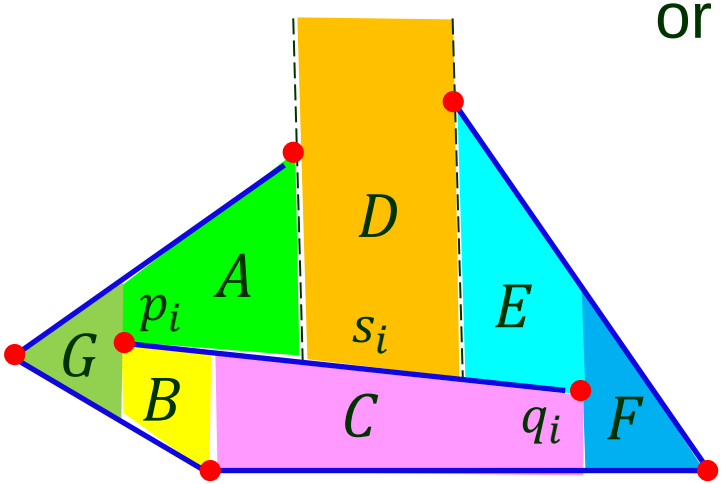
denoting
$\Delta = \Delta_q(S_i)$

$\implies$ either top$(\Delta) = s_i$

or bottom$(\Delta) = s_i$

or leftp$(\Delta) = p_i$ or q_i

or rightp$(\Delta) = p_i$ or q_i



Cont'd

$\Delta_q(S_i) \neq \Delta_q(S_{i-1}) \implies \Delta_q(S_i)$ is created in iteration i

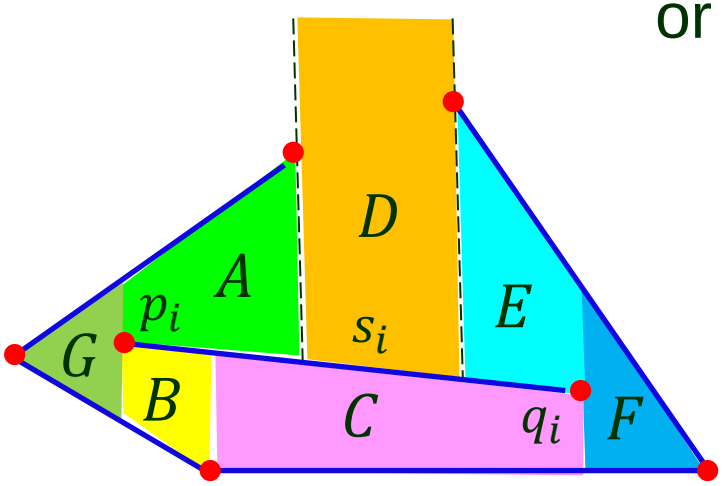
denoting
$\Delta = \Delta_q(S_i)$

$\implies$ either top$(\Delta) = s_i$

or bottom$(\Delta) = s_i$

or leftp$(\Delta) = p_i$ or q_i

or rightp$(\Delta) = p_i$ or q_i



$\Delta_q(S_i)$ as a trapezoid in $T(S_i)$ does not depend on the order of segments in S_i.

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.

Backward Analysis

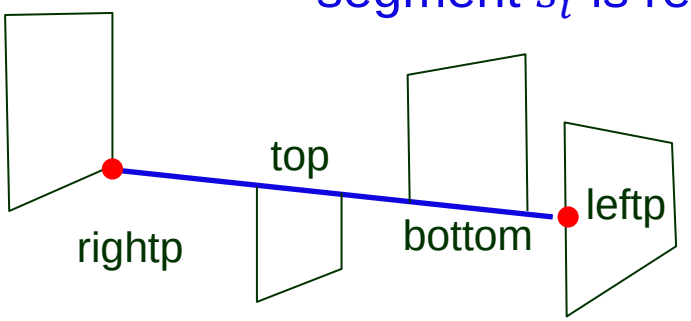
Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.

⇕ iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.

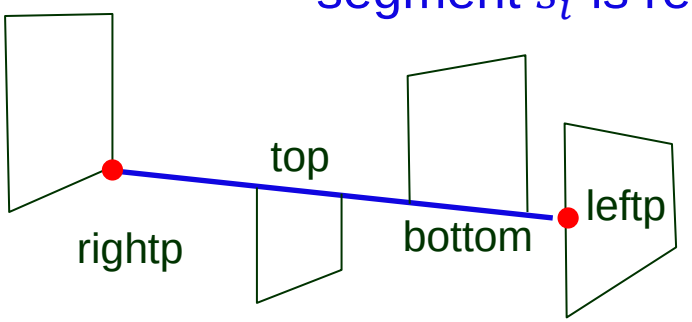


iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.



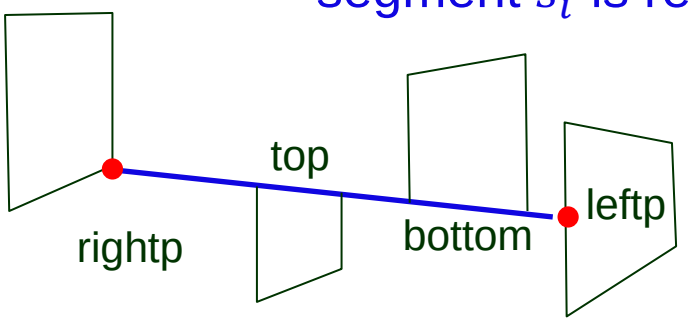
iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

- Disappearance of top(Δ)

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.



⇕ iff

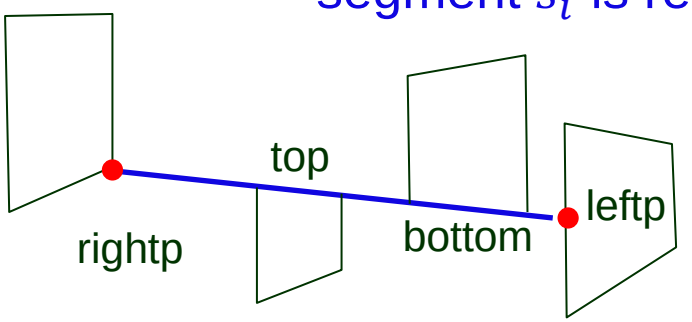
One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

- Disappearance of top(Δ)

  Every segment in S_i is equally likely to be s_i.

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.



⇕ iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

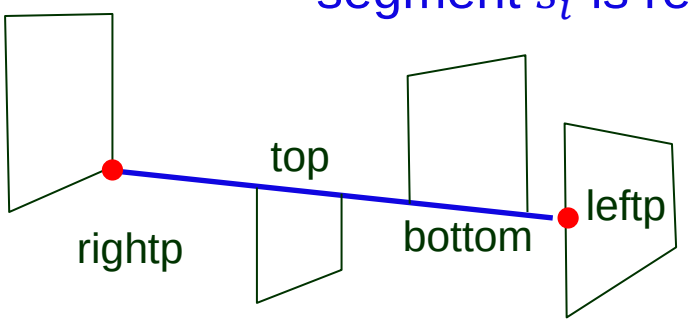
- Disappearance of top(Δ)

  Every segment in S_i is equally likely to be s_i.

  ⇨ s_i happens to be top(Δ) with probability $\frac{1}{i}$, i.e.,

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.



⇕ iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

- Disappearance of top(Δ)

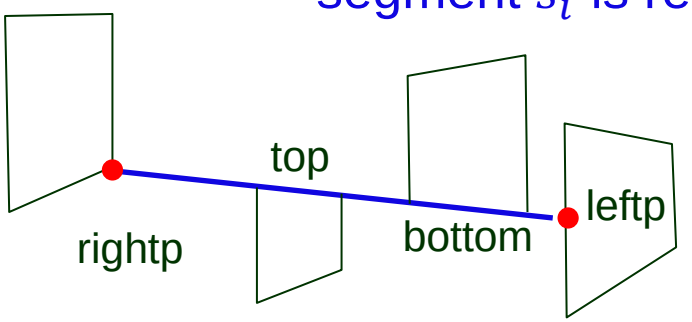
Every segment in S_i is equally likely to be s_i.

⇨ s_i happens to be top(Δ) with probability $\frac{1}{i}$, i.e.,

$$\Pr(s_i = \text{top}(\Delta)) = \frac{1}{i}$$

Backward Analysis

Idea Look at the probability that $\Delta = \Delta_q(S_i)$ disappears when the segment s_i is removed.



⇕ iff

One of top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) disappears when s_i is removed.

- Disappearance of top(Δ)

  Every segment in S_i is equally likely to be s_i.

  ⇨ s_i happens to be top(Δ) with probability $\frac{1}{i}$, i.e.,

  $$\Pr(s_i = \text{top}(\Delta)) = \frac{1}{i}$$

- Disappearance of bottom(Δ)

  $$\Pr(s_i = \text{bottom}(\Delta)) = \frac{1}{i}$$

Analysis (cont'd)

- Disappearance of leftp(Δ)

Analysis (cont'd)

- Disappearance of leftp(Δ)

  ⇨ s_i is the only segment in S_i with leftp(Δ) as the end point

Analysis (cont'd)

- Disappearance of leftp(Δ)

  ⇨ s_i is the only segment in S_i with leftp(Δ) as the end point

  ⇨ probability $\leq \frac{1}{i}$

Analysis (cont'd)

- Disappearance of leftp(Δ)

  ⇨ s_i is the only segment in S_i with leftp(Δ) as the end point

  ⇨ probability $\leq \frac{1}{i}$

- Disappearance of rightp(Δ)

  Similarly, probability $\leq \frac{1}{i}$

Analysis (cont'd)

- Disappearance of leftp(Δ)

  ⇨ s_i is the only segment in S_i with leftp(Δ) as the end point

  ⇨ probability $\leq \frac{1}{i}$

- Disappearance of rightp(Δ)

  Similarly, probability $\leq \frac{1}{i}$

$$P_i = \Pr\left(\Delta_q(S_i) \neq \Delta_q(S_{i-1})\right)$$
$$= \Pr\left(\Delta_q(S_i) \notin T(S_{i-1})\right)$$
$$\leq \frac{4}{i}$$

Expected Query Time

$$E\left(\sum_{i=1}^{n} X_i\right) \leq \sum_{i=1}^{n} 3P_i$$

Expected Query Time

$$E\left(\sum_{i=1}^{n} X_i\right) \leq \sum_{i=1}^{n} 3P_i$$

$$\leq 12 \sum_{i=1}^{n} \frac{1}{i}$$

Expected Query Time

$$E\left(\sum_{i=1}^{n} X_i\right) \leq \sum_{i=1}^{n} 3P_i$$

$$\leq 12 \sum_{i=1}^{n} \frac{1}{i}$$

$$= 12\, H_n$$

$H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the nth Harmonic number.

Expected Query Time

$$E\left(\sum_{i=1}^{n} X_i\right) \leq \sum_{i=1}^{n} 3P_i$$

$$\leq 12 \sum_{i=1}^{n} \frac{1}{i}$$

$$= 12\, H_n$$

$H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the nth Harmonic number.

$$\ln n \leq H_n \leq \ln n + 1$$

Expected Query Time

$$E\left(\sum_{i=1}^{n} X_i\right) \leq \sum_{i=1}^{n} 3P_i$$

$$\leq 12 \sum_{i=1}^{n} \frac{1}{i}$$

$$= 12\, H_n \quad = O(\log n)$$

$H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the nth Harmonic number.

$$\ln n \leq H_n \leq \ln n + 1$$

Time and Space

Theorem a) The trapezoidal map $T(S)$ and search structure D are computed in $O(n \log n)$ expected time.
// not discussed

b) Expected size of D is $O(n)$. // not discussed

c) Expected query time is $O(\log n)$.