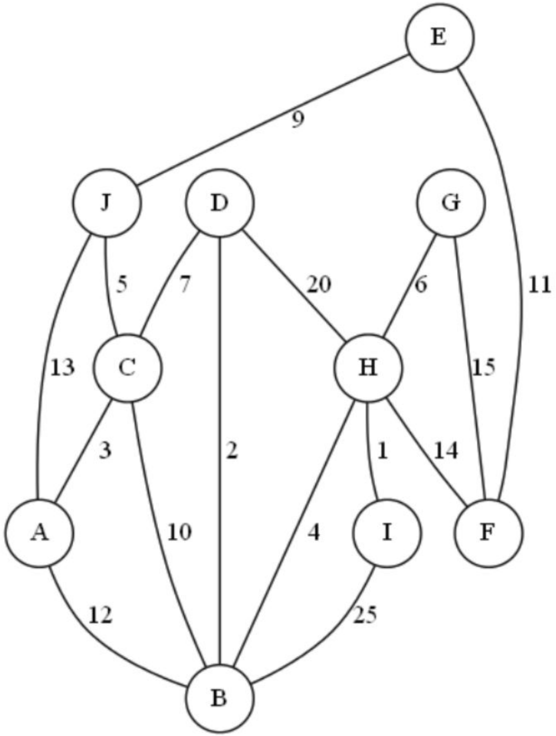
PSO 15

The Last One
(misc. review)

Question 1

(Prim's algorithm & MST)

(1) Run the Prim's algorithm on the following graph with starting vertex A.



(2) Let G be a connected undirected graph of 100 vertices and 300 edges. The total weight of an MST of G is 500. When the weight of each edge of G is increased by 5, what is the total weight of the MST of the updated graph?

Question 2

(More on the Dijkstra's algorithm)

Suppose you are a city planner, and you are worried that it takes too long for an ambulance to get from the university to the hospital. You've decided to build one new street to improve this particular route. You and your team have researched and gathered a list of potential streets that could be built; the task now is to choose one street that will improve this route the most.

To model this formally, let $G = (V, E)$ be a directed graph with positive edge weights (representing the lengths of streets), and let $s, t \in V$ be two vertices in the graph. We assume that E represents the edges/streets that are "already built". Let $E^{'}$ represents the edges/streets that you can add to the graph. Then the problem is to identify the single edge $e \in E^{'}$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.

Question 3

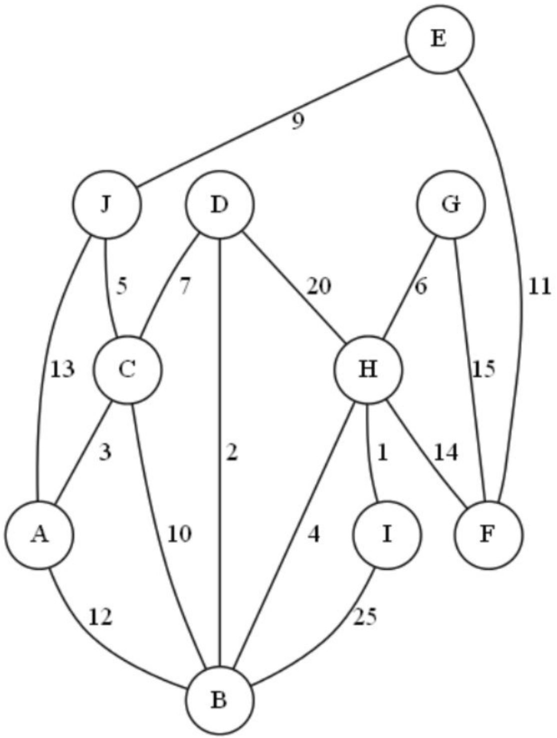
(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

(2) Insert $\{15, 21, 7, 24, 0, 26, 3, 28, 29\}$ (in the given order) into an initially empty 2-3 tree.

(3) Delete element 7 in the final 2-3 tree obtained in question (2).

(1) Run the Prim's algorithm on the following graph with starting vertex A.



Idea: use the cut property, where the cut is the iteratively growing tree

(2) Let G be a connected undirected graph of 100 vertices and 300 edges. The total weight of an MST of G is 500. When the weight of each edge of G is increased by 5, what is the total weight of the MST of the updated graph?

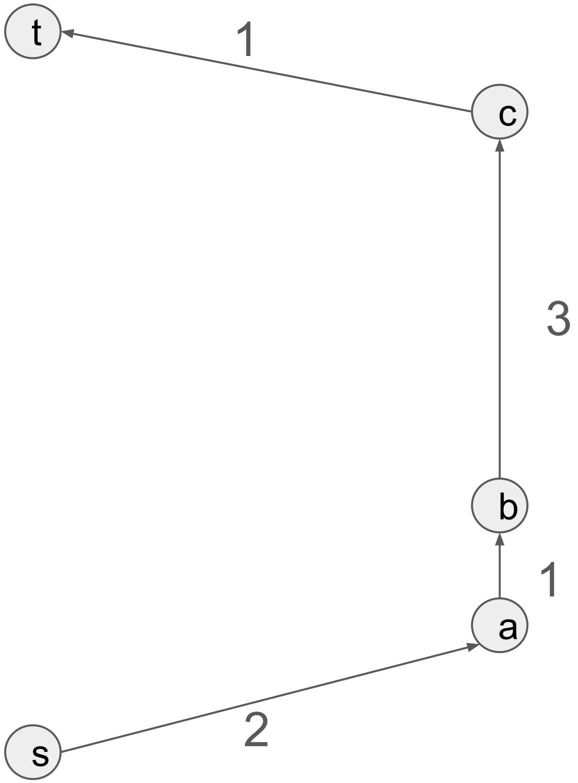
Question 2

(More on the Dijkstra's algorithm)

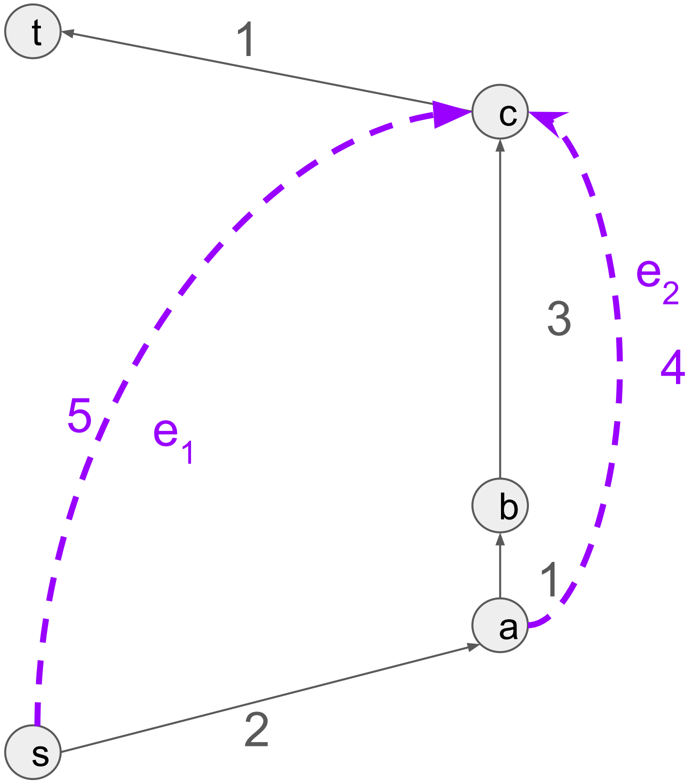
Suppose you are a city planner, and you are worried that it takes too long for an ambulance to get from the university to the hospital. You've decided to build one new street to improve this particular route. You and your team have researched and gathered a list of potential streets that could be built; the task now is to choose one street that will improve this route the most.

To model this formally, let $G = (V, E)$ be a directed graph with positive edge weights (representing the lengths of streets), and let $s, t \in V$ be two vertices in the graph. We assume that E represents the edges/streets that are"already built". Let E' represents the edges/streets that you can add to the graph. Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.

Reading time

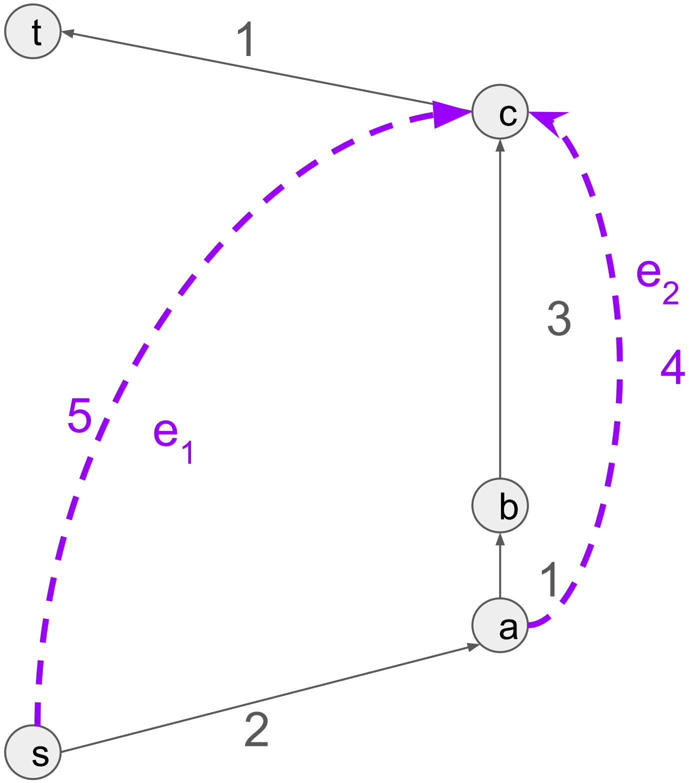


Let this be G = (V,E)



Let this be G = (V,E)

Let E’ be the purple dotted edges

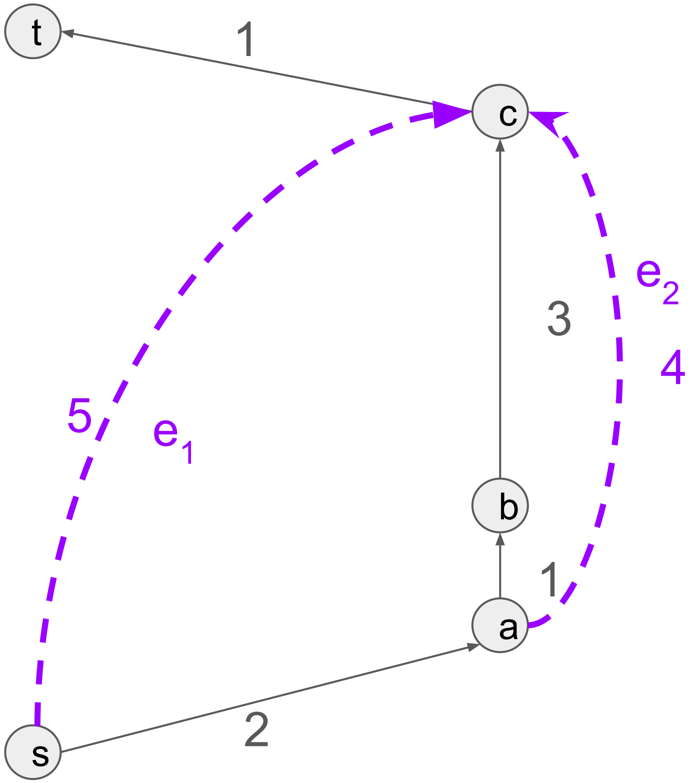


Let this be G = (V,E)

Let E' be the purple dotted edges

How do we choose between adding e_1 and e_2?

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



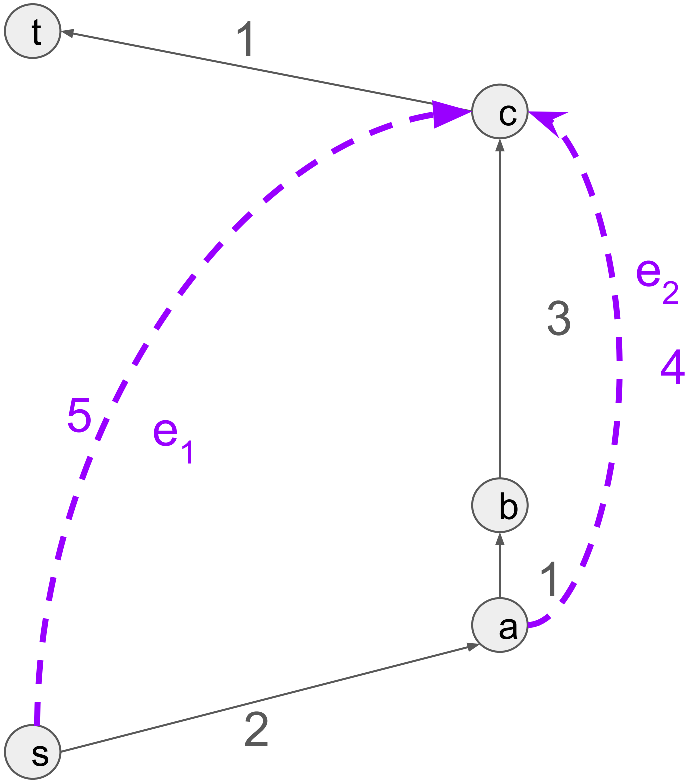
Let this be G = (V,E)

Let E’ be the purple dotted edges

How do we choose between adding e_1 and e_2?

Equivalent problem: choose e_i that *maximizes* savings

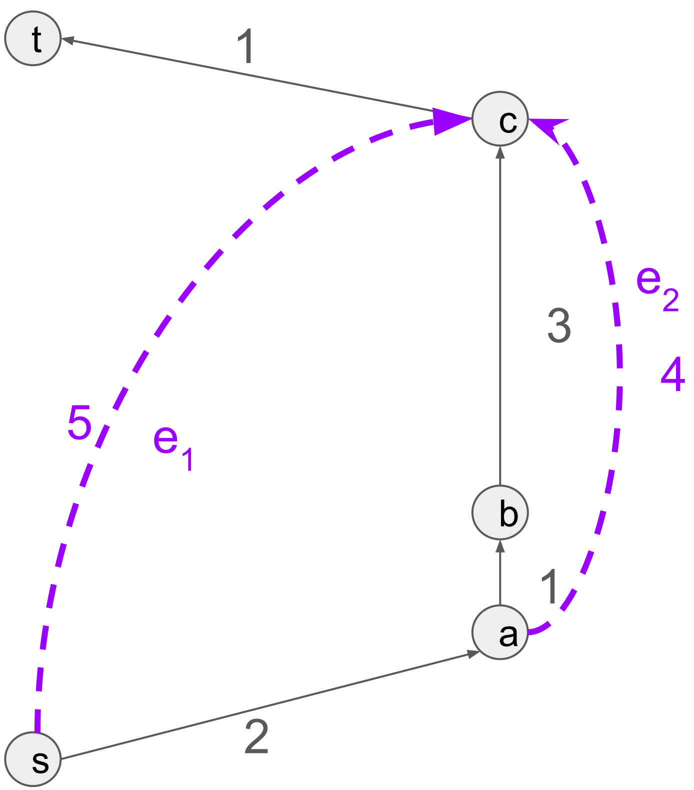
Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



Equivalent problem: choose e_i that *maximizes* savings

Let d_{st} be the original shortest path between s and t

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



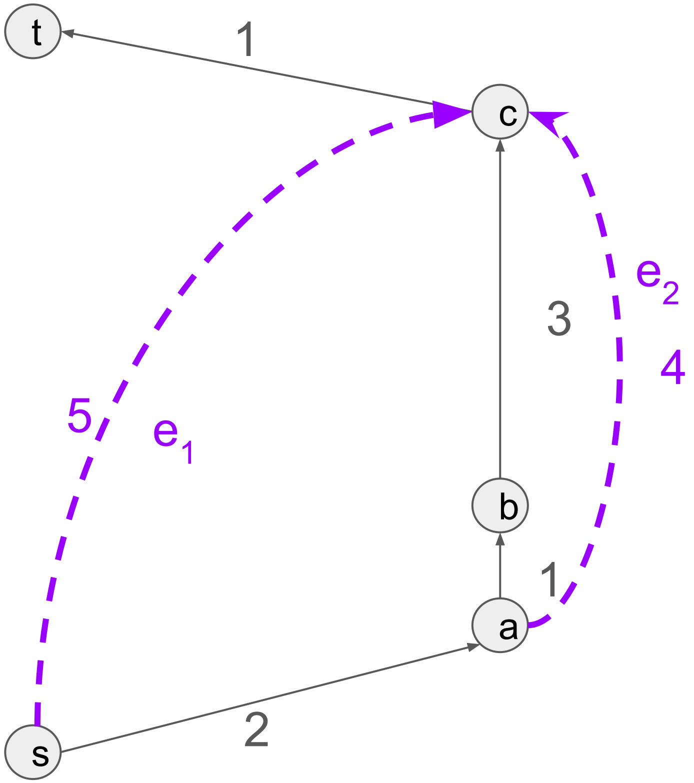
Equivalent problem: choose e_i that *maximizes* savings

Let d_{st} be the original shortest path between s and t

$\text{Savings}_{e1}=$

$\text{Savings}_{e2}=$

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



Equivalent problem: choose e_i that *maximizes* savings

Let d_{st} be the original shortest path between s and t

$\text{Savings}_{e1} = d_{st}$ - (new path w/ e_1)

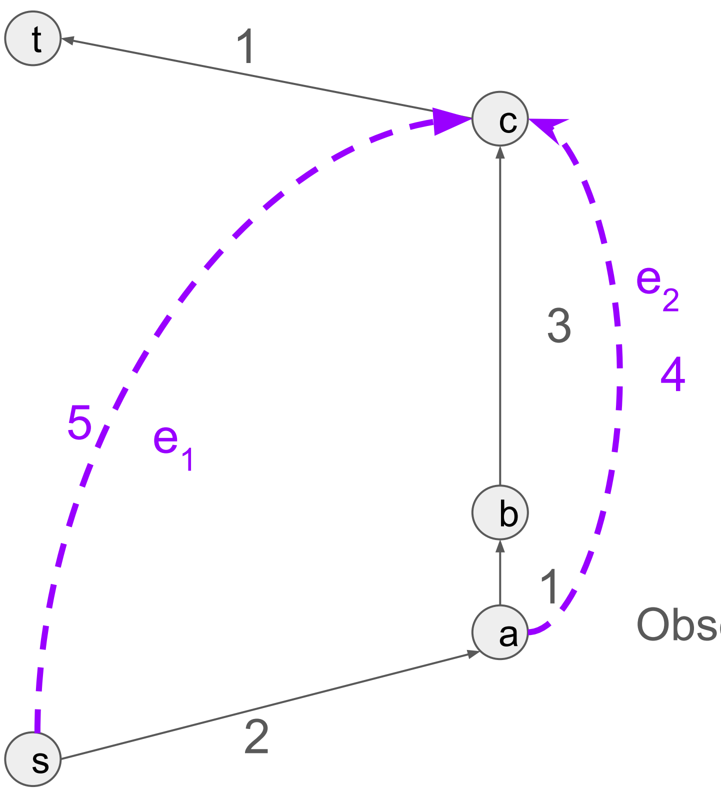
$\text{Savings}_{e2} = d_{st}$ - (new path w/ e_2)

New goal: compute (new path w/ e_i)

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.

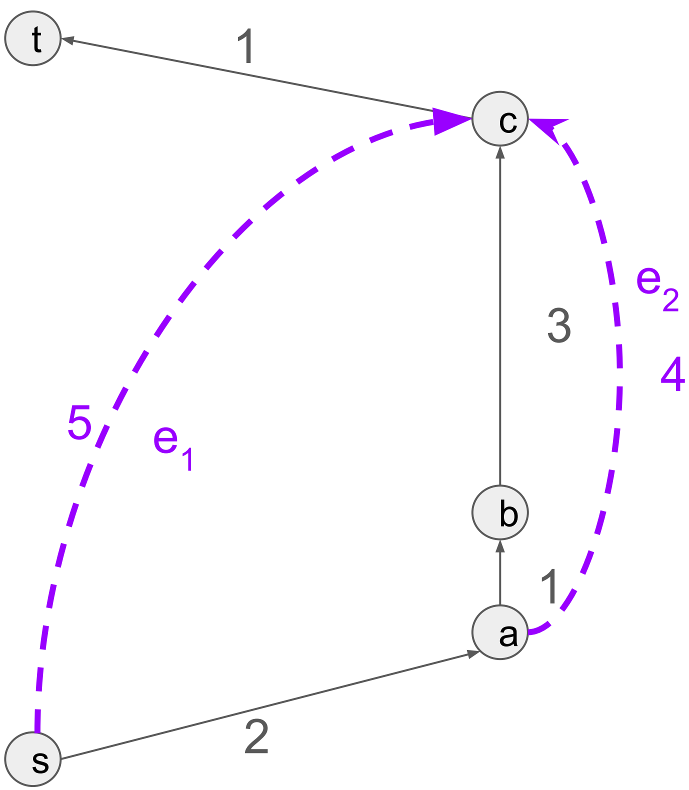
New goal: compute (new path w/ e_i)

(new path w/ e_1) =



Observation: e_1 only creates shorter paths to c

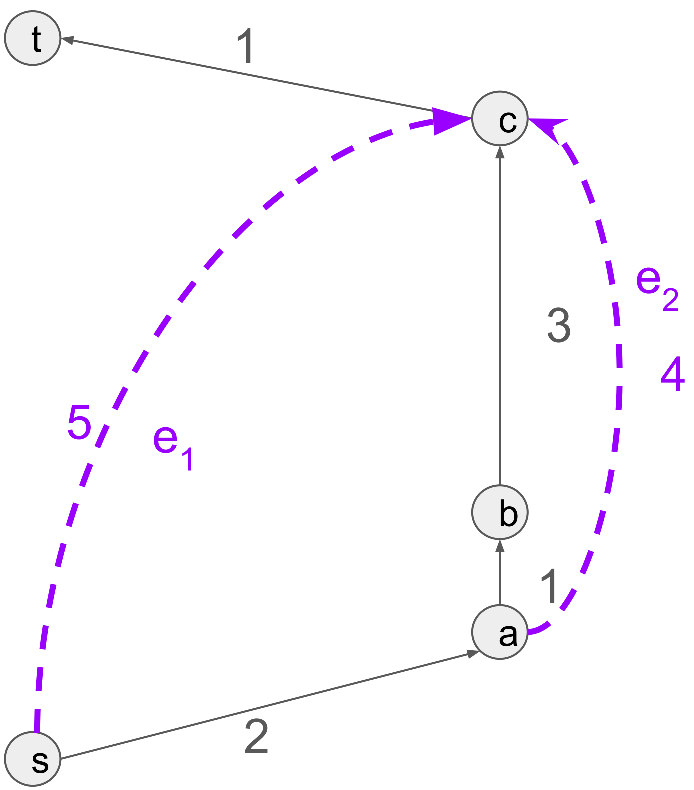
Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



New goal: compute (new path w/ e_i)

(new path w/ e_1) =
(new shortest path s to c) + d_{ct}

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.

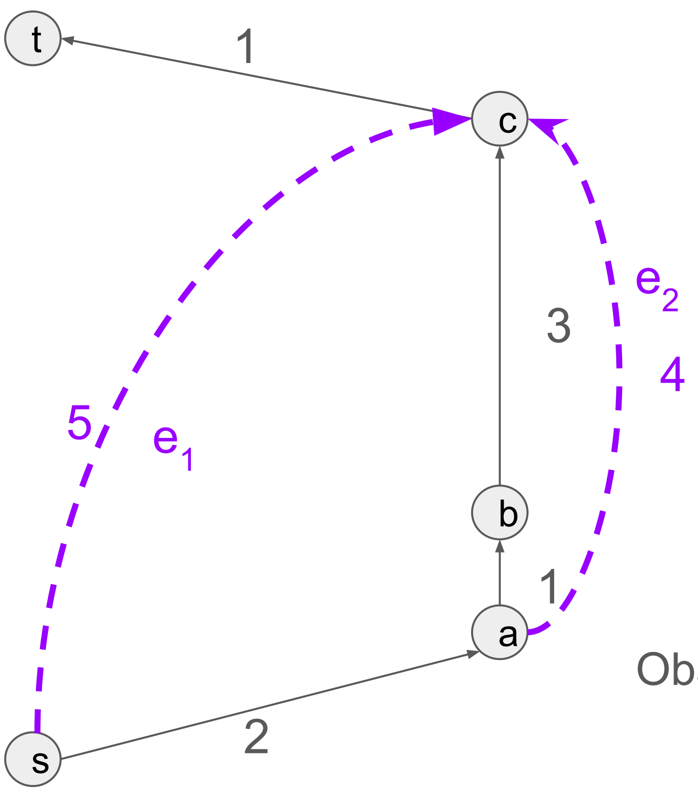


New goal: compute (new path w/ e_i)

(new path w/ e_1) =
(new shortest path s to c) + d_{ct}

= (e_1) + d_{ct}

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



New goal: compute (new path w/ e_i)

(new path w/ e_1) =
(new shortest path s to c) + d_{ct}

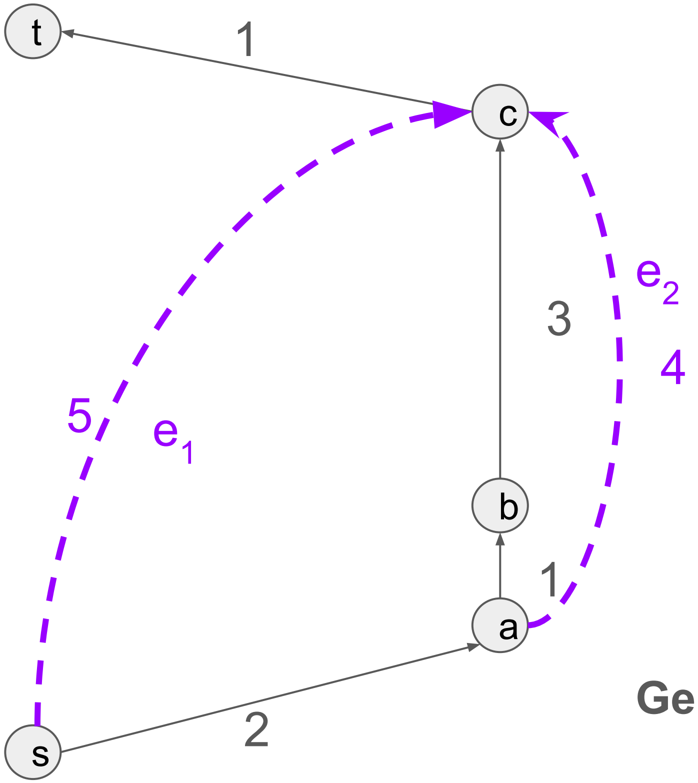
= (e_1) + d_{ct}

(new path w/ e_2) =
(new shortest path s to c) + d_{ct}

=

Observation: e_2 only creates shorter paths to c

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



New goal: compute (new path w/ e_i)

(new path w/ e_1) =
(new shortest path s to c) + d_{ct}

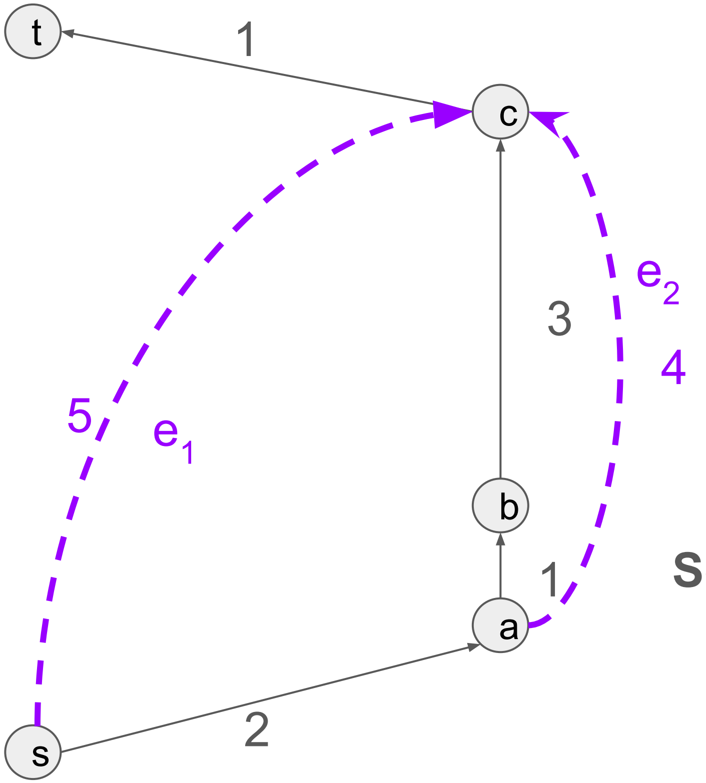
= $(e_1) + d_{ct}$

(new path w/ e_2) =
(new shortest path s to c) + d_{ct}

= $(d_{sa} + e_2) + d_{ct}$

General formula for new edge (p,q)?

Then the problem is to identify the single edge $e \in E'$ so that, when you add e to G, the distance from s to t in the augmented graph is as short as possible. Design an algorithm for this problem and and analyze its time complexity.



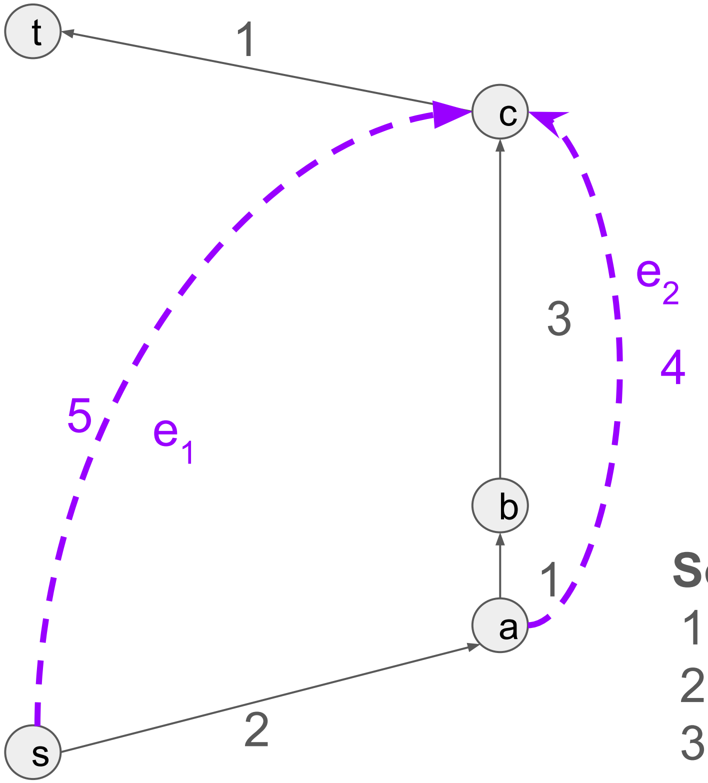
Equivalent problem: choose e_i that *maximizes* savings

- $\text{Savings}_{e1} = d_{st}$ - (new path w/ e_i)

- (new path w/ $e_i = (p,q)$) =
  (new shortest path s to q) + d_{qt}

  $= (d_{sq} + e_i) + d_{qt}$

So what do I need to calculate?



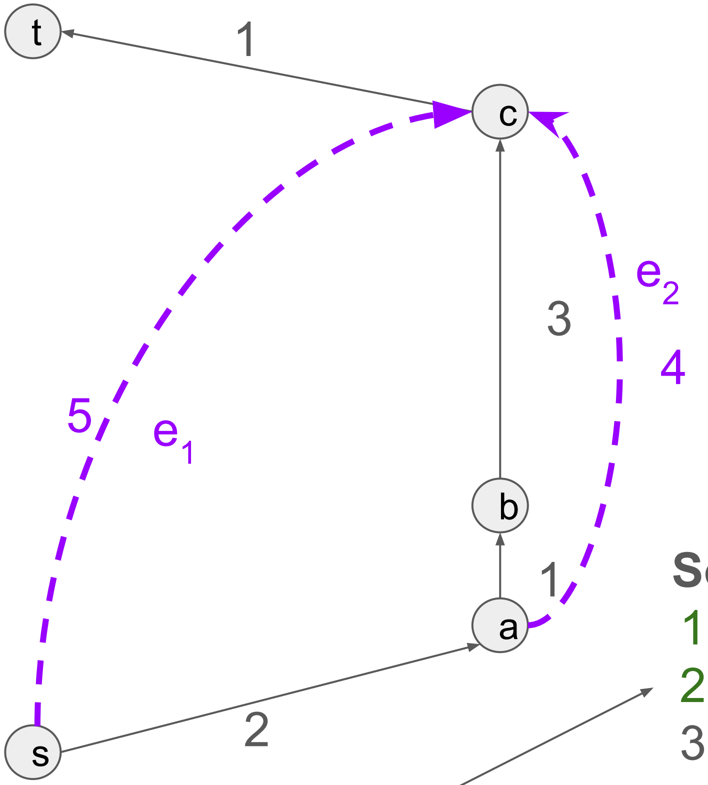
Equivalent problem: choose e_i that *maximizes* savings

- Savings$_{e1}$ = d_{st} - (new path w/ e_i)
- (new path w/ e_i = (p,q)) =
  (new shortest path s to q) + d_{qt}

  = (d_{sq} + e_i) + d_{qt}

So what do I need to calculate?

1. d_{st} : shortest path from s to t
2. d_{sq} : shortest path from s to each q
3. d_{qt} : shortest path from t to each q



Run Dijkstra(G,s) to get these

Equivalent problem: choose e_i that *maximizes* savings

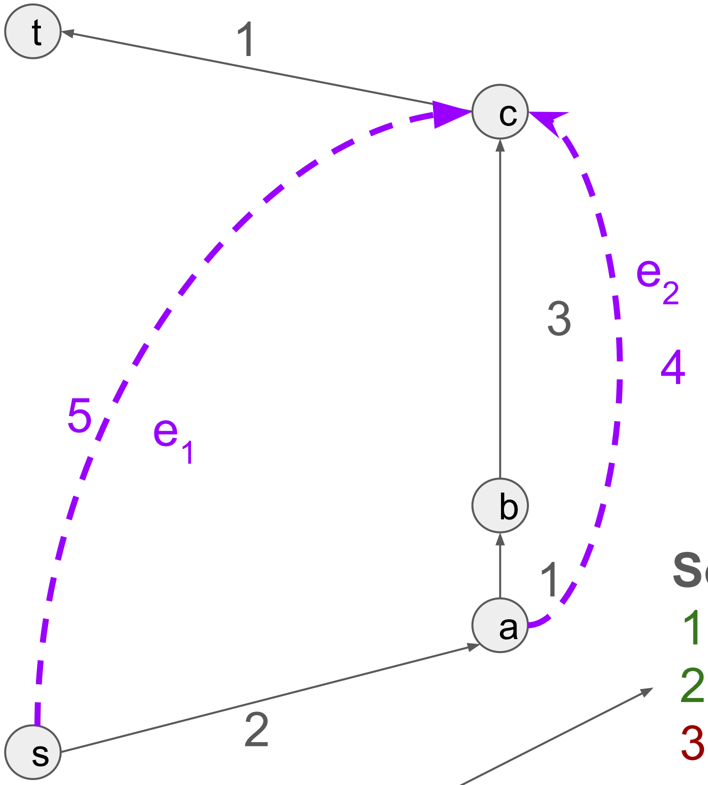
- Savings$_{e1}$ = d_{st} - (new path w/ e_i)

- (new path w/ e_i = (p,q)) =
  (new shortest path s to q) + d_{qt}

  = (d_{sq} + e_i) + d_{qt}

So what do I need to calculate?

1. d_{st} : shortest path from s to t
2. d_{sq} : shortest path from s to each q
3. d_{qt} : shortest path from t to each q



Equivalent problem: choose e_i that *maximizes* savings

- Savings$_{e1}$ = d_{st} - (new path w/ e_i)
- (new path w/ e_i = (p,q)) = (new shortest path s to q) + d_{qt}

  $= (d_{sq} + e_i) + d_{qt}$

So what do I need to calculate?

1. d_{st} : shortest path from s to t
2. d_{sq} : shortest path from s to each q
3. d_{qt} : shortest path from t to each q

Run Dijkstra(G,s) to get these

Run Dijkstra(G,q) to get this?

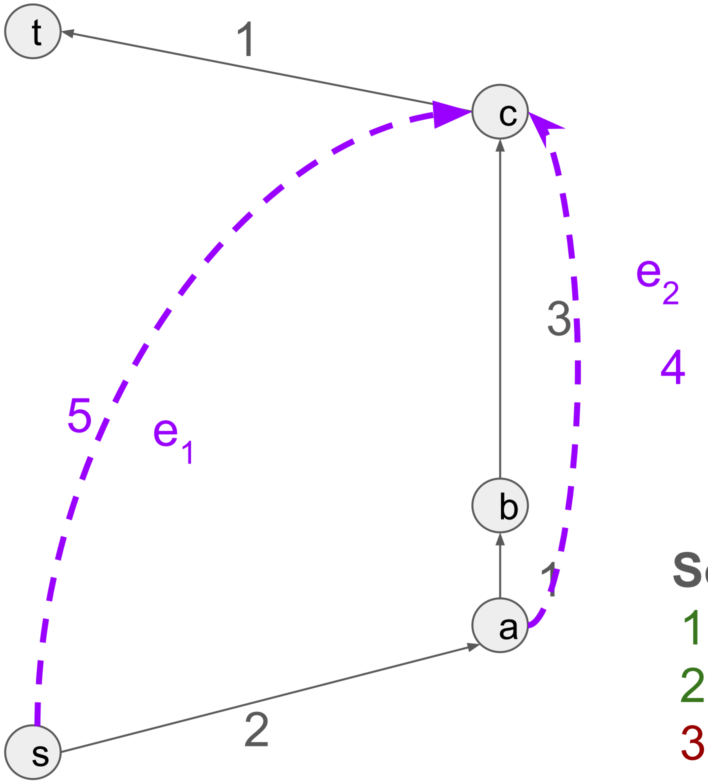
d_{qt} : shortest path from t to each q

Runing Dijkstra(G,q) for each q to get this is inefficient

- worst case running Dijstra O(n) times.

Exercise: figure out how to get all d_{qt} with only a single extra dijkstra run

(Hint: Modifying the graph may be useful



Equivalent problem: choose e_i that *maximizes* savings

- Savings$_{e1}$ = d_{st} - (new path w/ e_i)

- (new path w/ e_i = (p,q)) =
  (new shortest path s to q) + d_{qt}

  = $(d_{sq} + e_i) + d_{qt}$

So what do I need to calculate?

1. d_{st} : shortest path from s to t
2. d_{sq} : shortest path from s to each q
3. d_{qt} : shortest path from t to each q

Solution runs in O(dijkstra running time)

Question 3

(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

First, what is a 2-3 tree?

Question 3

(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

First, what is a 2-3 tree?

T is a 2-3 tree if:

- T empty
- T is a **2 node**
- T is a **3 node**

Question 3

(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

First, what is a 2-3 tree?

T is a 2-3 tree if:

- T empty
- T is a **2 node**
- T is a **3 node**

T –>

Question 3

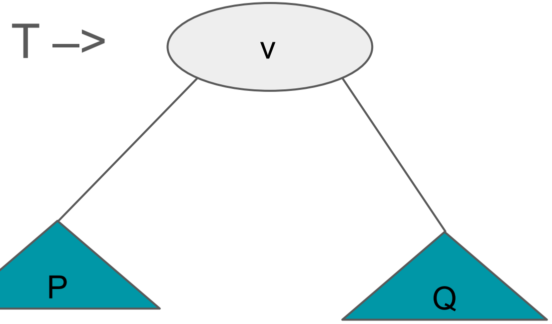
(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

First, what is a 2-3 tree?

T is a 2-3 tree if:

- T empty
- <u>T is a **2 node**</u>
- T is a **3 node**



P,Q are 2-3 subtrees where

- P,Q have the *same height*
- P < v < Q

Question 3

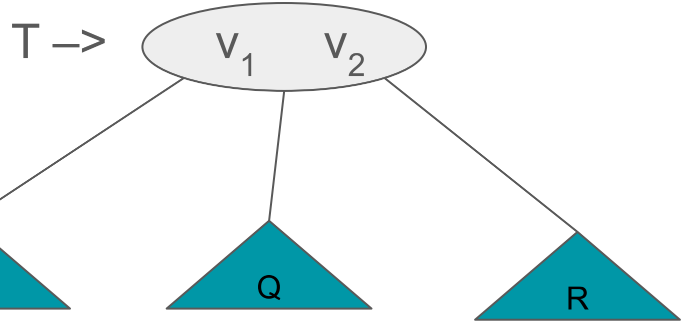
(2-3 tree)

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$?

First, what is a 2-3 tree?

T is a 2-3 tree if:

- T empty
- T is a **2 node**
- T is a **3 node**

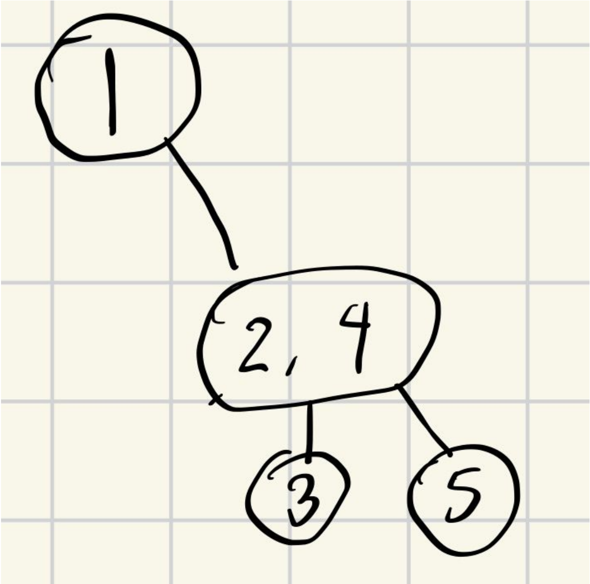


P,Q,R are 2-3 subtrees where

- P,Q have the *same height*
- $P < v_1 < Q < v_2 < R$

Let’s try to make some 2-3 trees

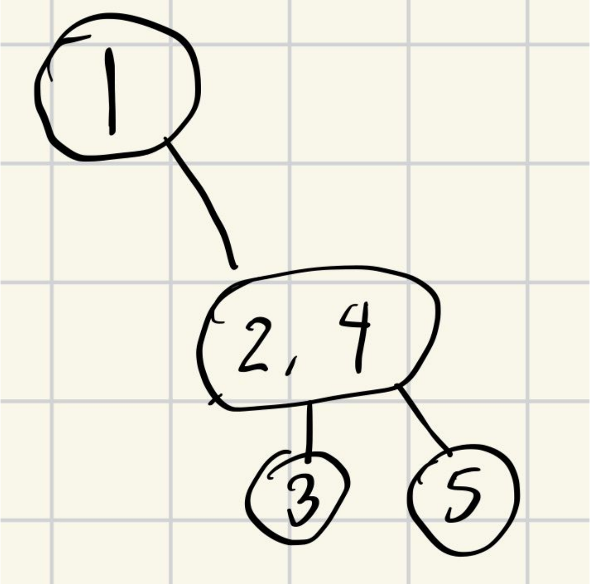
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Is this a 2-3 tree?

Let's try to make some 2-3 trees

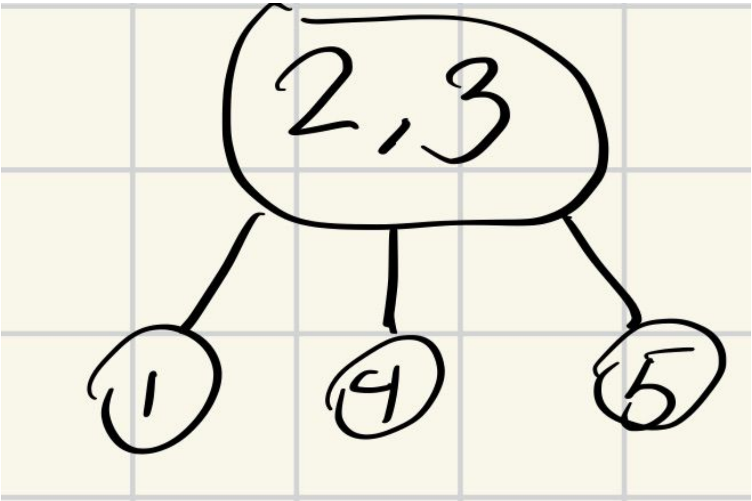
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Is this a 2-3 tree?
No the root and inner node is missing its left child

Let's try to make some 2-3 trees

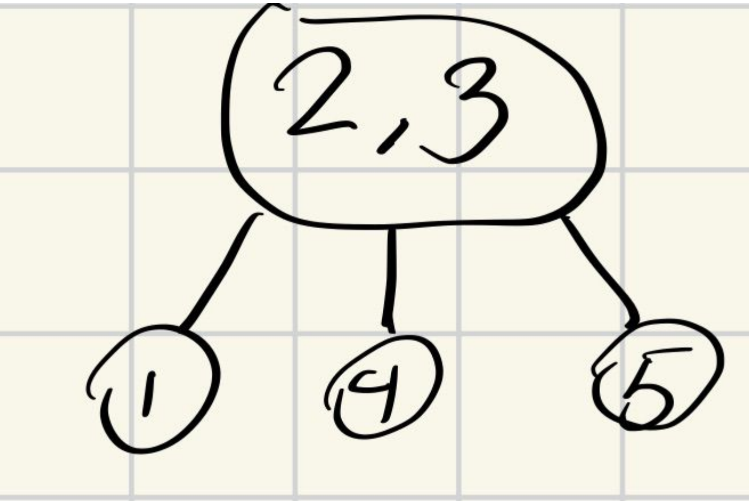
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



How about this one?

Let’s try to make some 2-3 trees

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.

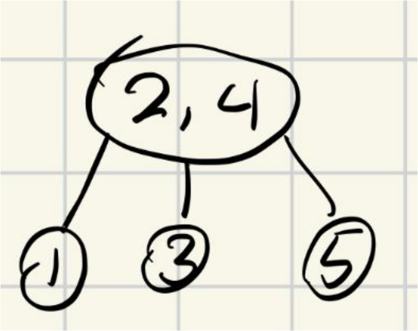


How about this one?
No,

1 < 2 ~~<~~ 4 ~~<~~ 3 < 5

Let’s try to make some 2-3 trees

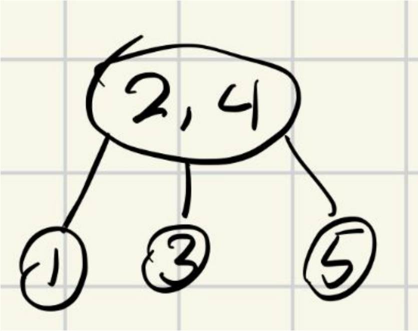
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Surely this one is bad too, right?

Let’s try to make some 2-3 trees

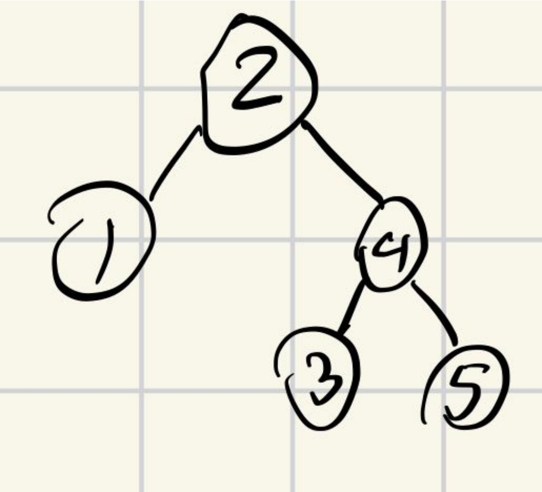
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Surely this one is bad too, right?
This one is ok

Let’s try to make some 2-3 trees

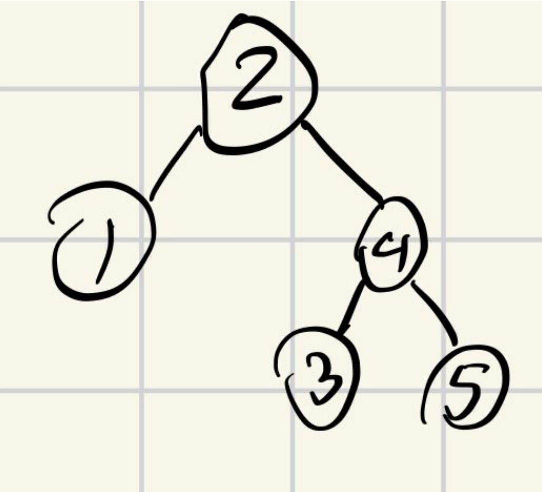
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



This one?

Let's try to make some 2-3 trees

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.

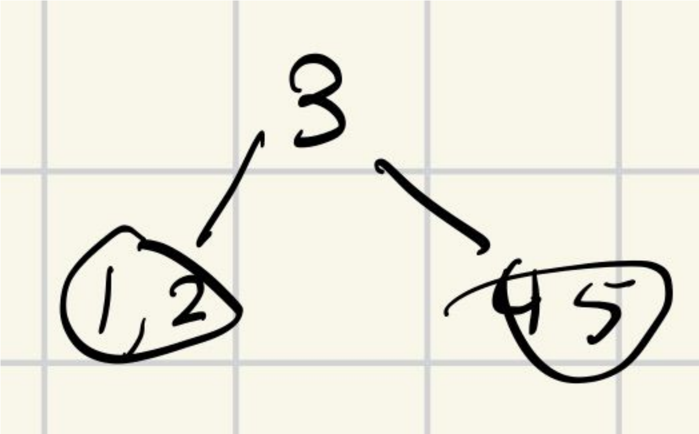


This one?
No, not all subtrees have the same height

(This *is* a BST though)

Let's try to make some 2-3 trees

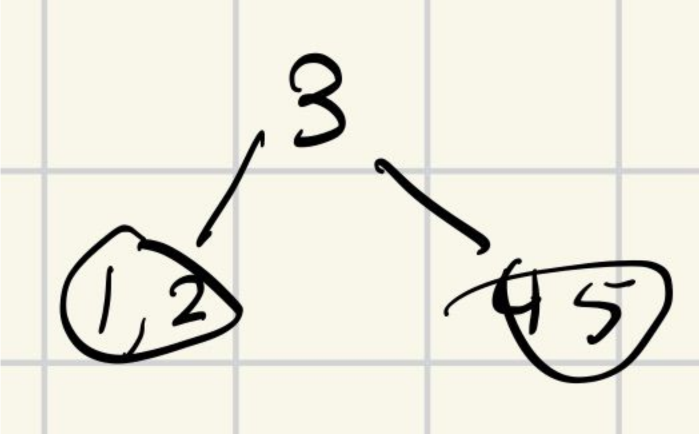
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Last one I promise. Is this a 2-3 tree?

Let's try to make some 2-3 trees

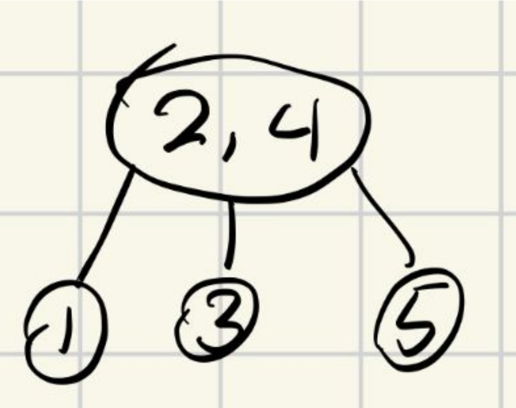
(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



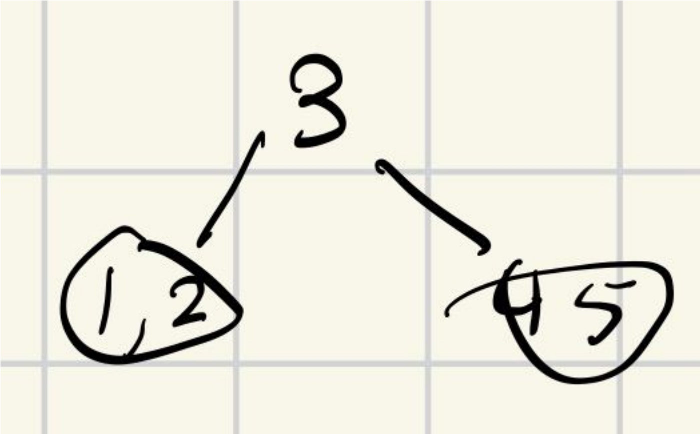
Last one I promise. Is this a 2-3 tree?
Yes nothing wrong here

The only 2-3 trees

(1) How many 2–3 trees exist storing the keys $\{1, 2, 3, 4, 5\}$? Explain your answer.



Only 2-3 tree with a 3-node root



Only 2-3 tree with a 2-node root

Can you see why?

(1) Insert $\{15, 21, 7, 24, 0, 26, 3, 28, 29\}$ (in the given order) into an initially empty 2-3 tree.

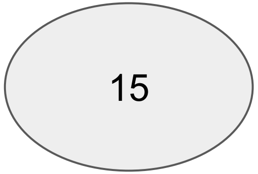
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

Insert: <u>15</u>,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

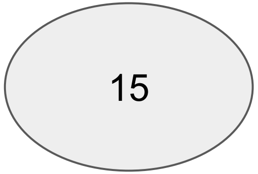
Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



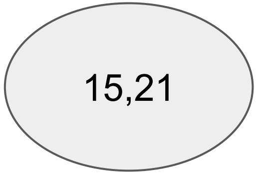
Insert: 15,<u>21</u>,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



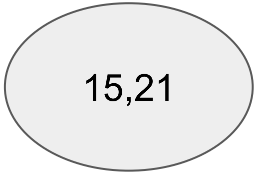
Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



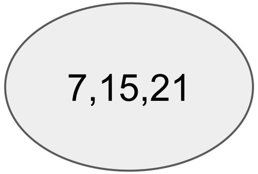
Insert: 15,21,<u>7</u>,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Insert: 15,21,7,24,0,26,3,28,29

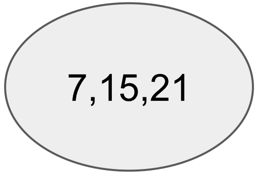
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



This is now a 4-node, need to fix.
Intuition: Pull up by the middle

Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



This is now a 4-node, need to fix.
Intuition: Pull up by the middle

Ur CPU core
fixing ur broken tree

Insert: 15,21,<u>7</u>,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

This is now a 4-node, need to fix.
Intuition: Pull up by the middle

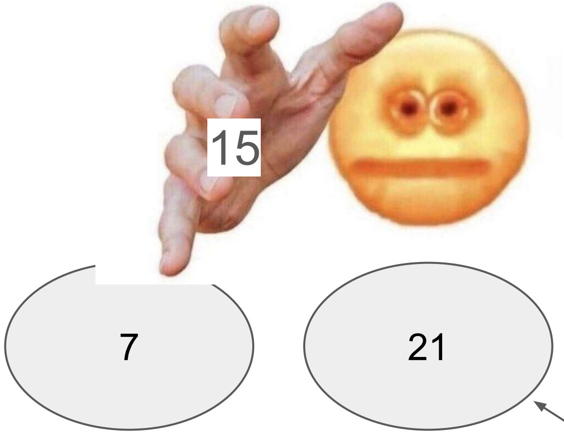
Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

This is now a 4-node, need to fix.
Intuition: Pull up by the middle

Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

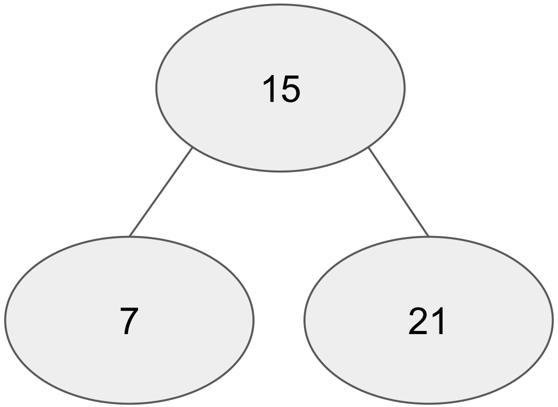


This is now a 4-node, need to fix.
Intuition: Pull up by the middle

The nodes split (out of fear)

Insert: 15,21,<u>7</u>,24,0,26,3,28,29

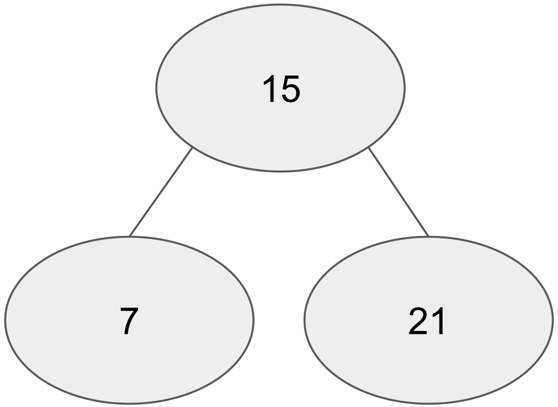
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



This is now a 4-node, need to fix.
Intuition: Pull up by the middle

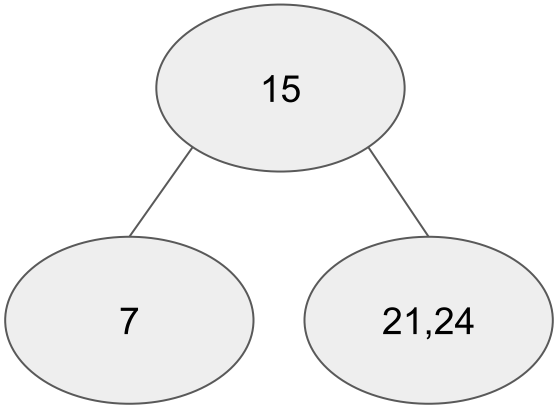
Insert: 15,21,7,<u>24</u>,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



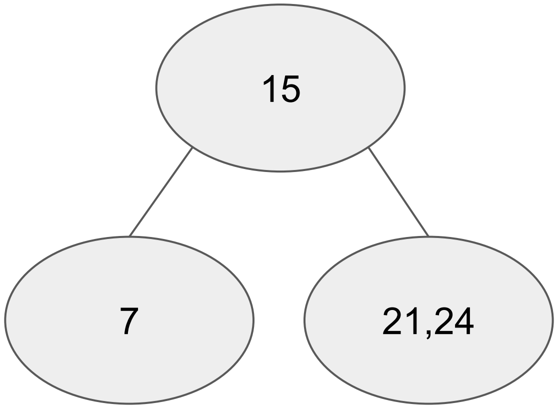
Insert: 15,21,7,<u>24</u>,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



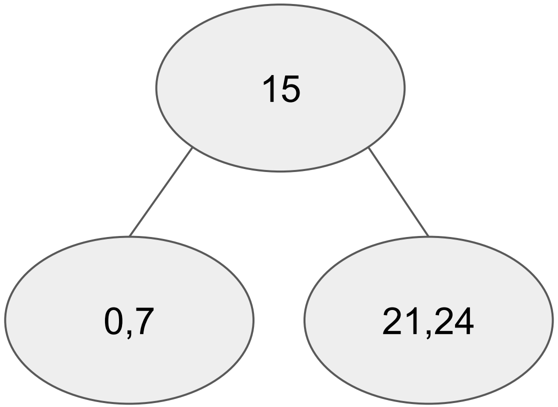
Insert: 15,21,7,24,<u>0</u>,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



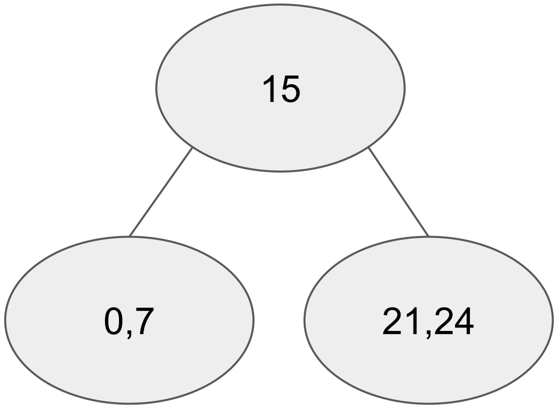
Insert: 15,21,7,24,<u>0</u>,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



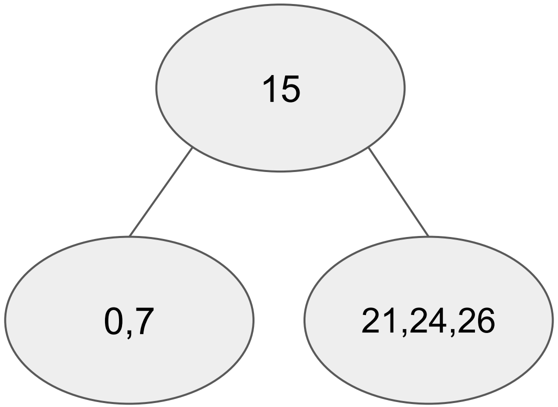
Insert: 15,21,7,24,0,<u>26</u>,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Insert: 15,21,7,24,0,<u>26</u>,3,28,29

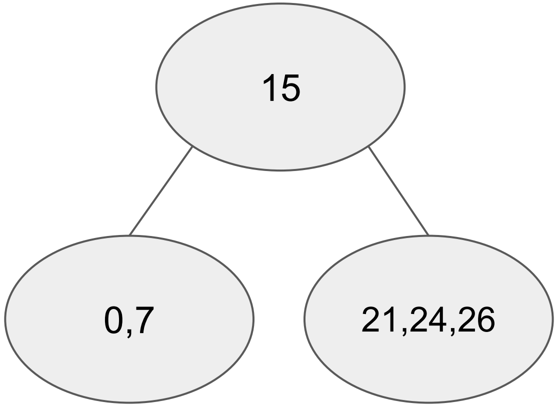
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node…

Insert: 15,21,7,24,0,26,3,28,29

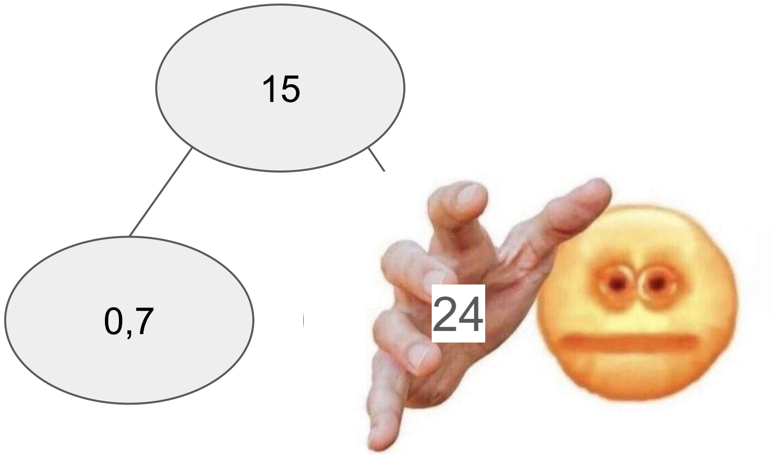
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node…

Insert: 15,21,7,24,0,<u>26</u>,3,28,29

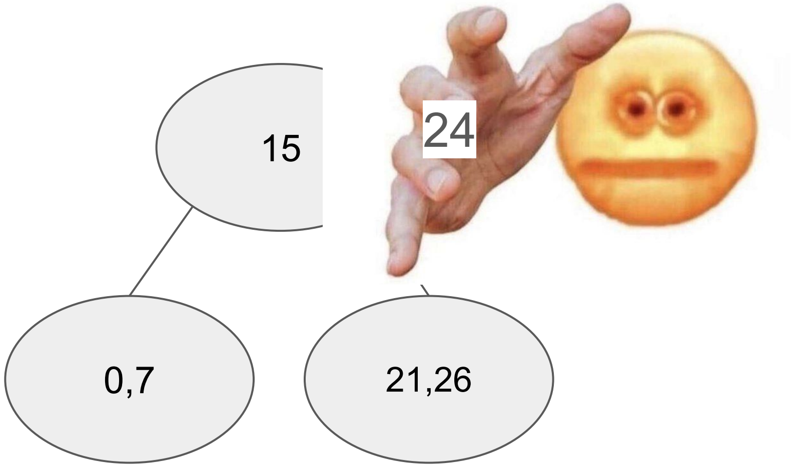
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node… “pull it up”

Insert: 15,21,7,24,0,<u>26</u>,3,28,29

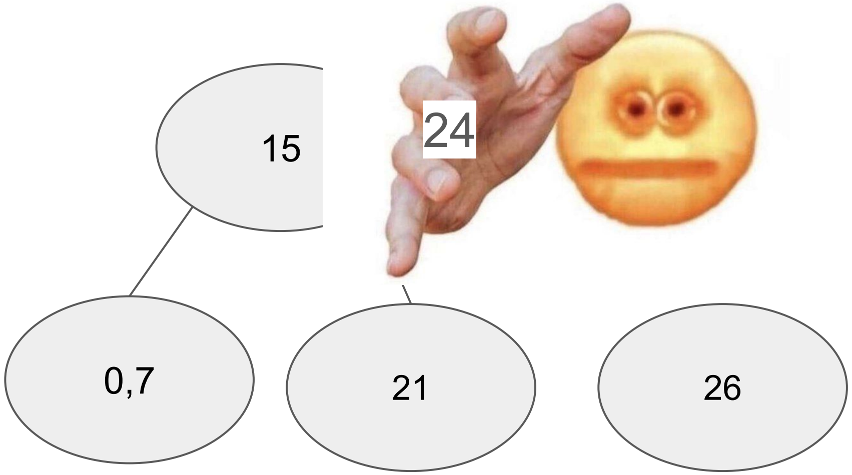
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node… “pull it up”

Insert: 15,21,7,24,0,<u>26</u>,3,28,29

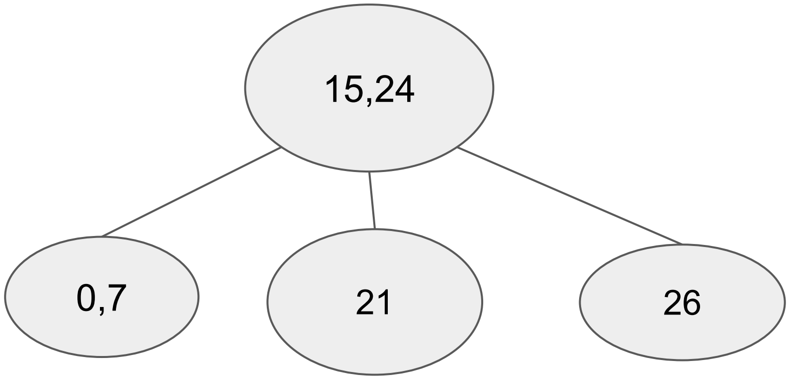
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node… "pull it up"

Insert: 15,21,7,24,0,<u>26</u>,3,28,29

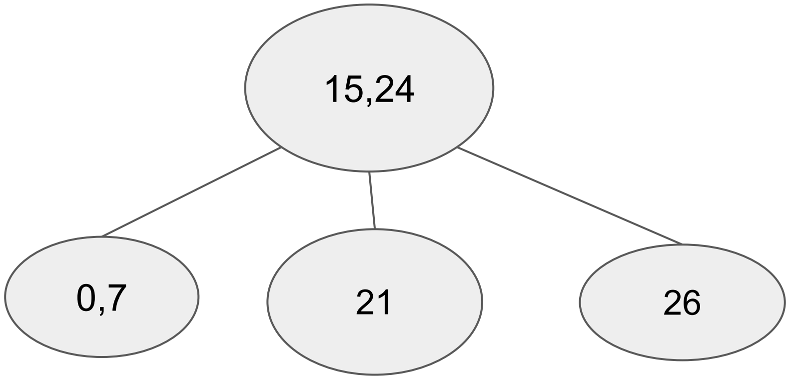
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



A 4-node… "pull it up"

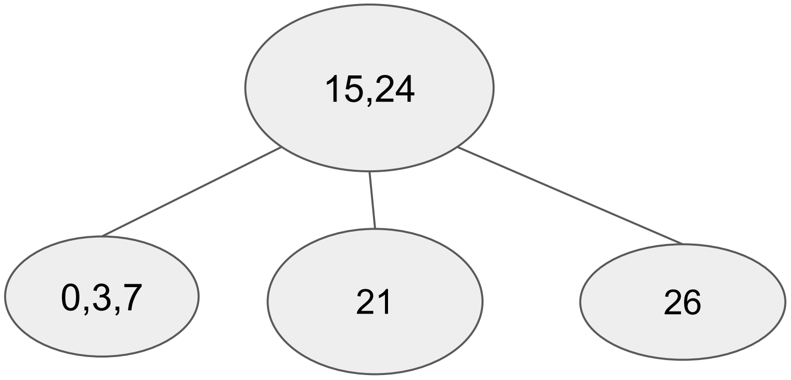
Insert: 15,21,7,24,0,26,<u>3</u>,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Insert: 15,21,7,24,0,26,3,28,29

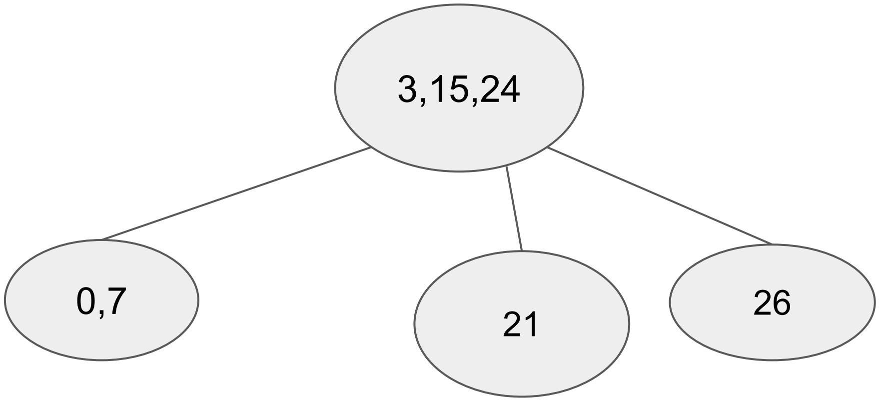
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Pull up the 3

Insert: 15,21,7,24,0,26,<u>3</u>,28,29

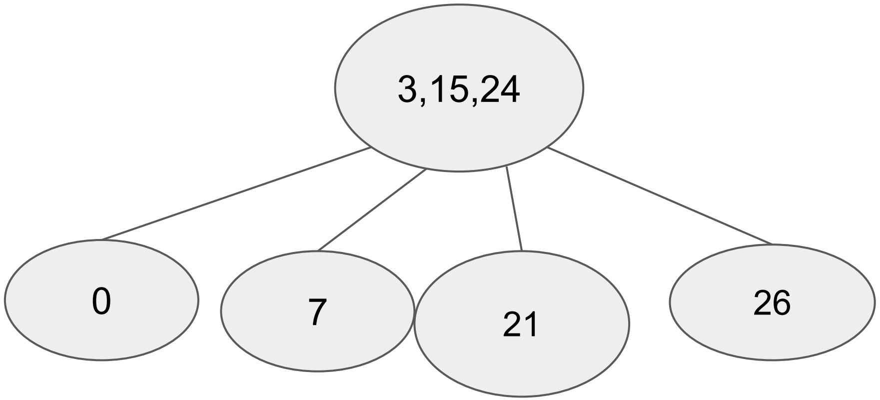
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Pull up the 3
Split the node

Insert: 15,21,7,24,0,26,<u>3</u>,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

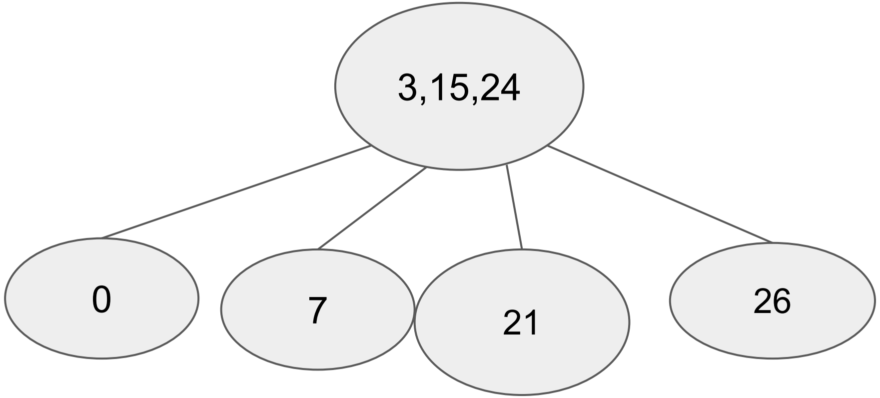


Pull up the 3
Split the node

Now the root is a 4-node, so
we need to keep pulling up

Insert: 15,21,7,24,0,26,<u>3</u>,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

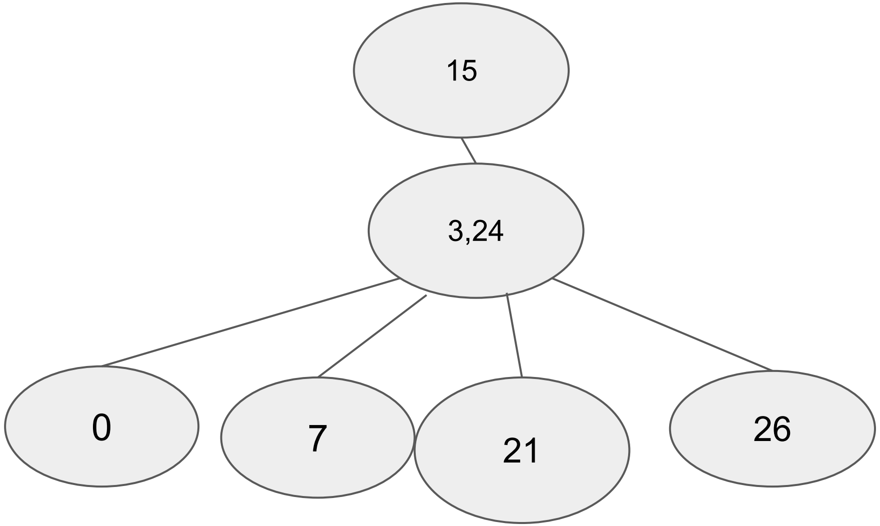


Pull up the 3
Split the node

Now the root is a 4-node, so
we need to keep pulling up

Insert: 15,21,7,24,0,26,<u>3</u>,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

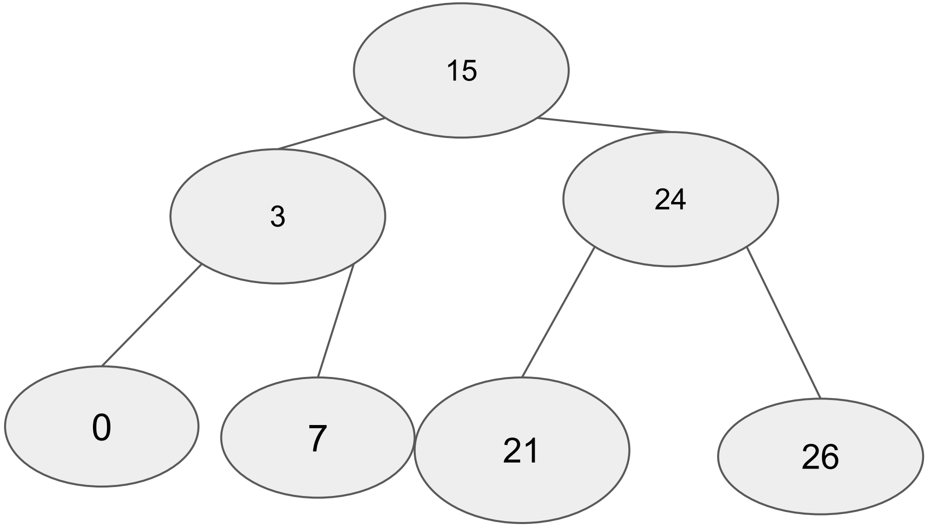


Pull up the 3
Split the node

Now the root is a 4-node, so
we need to keep pulling up

Insert: 15,21,7,24,0,26,3,28,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed

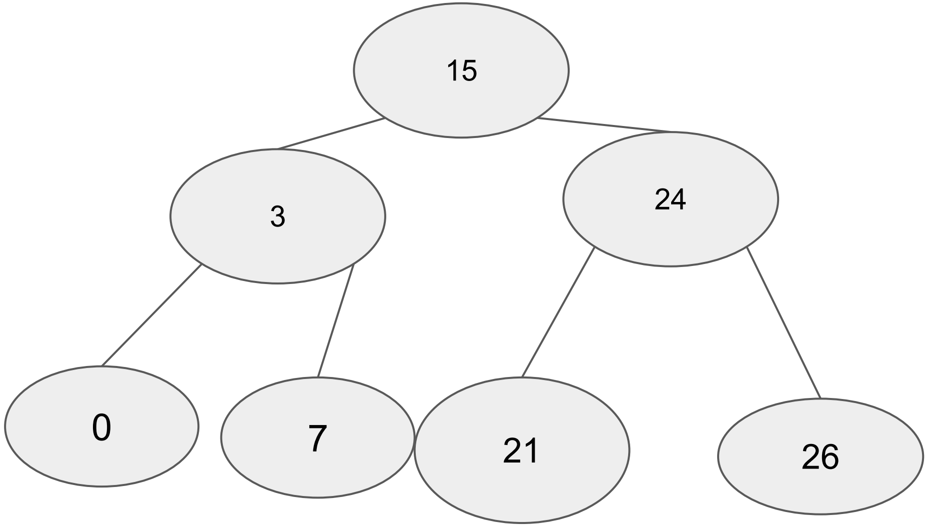


Pull up the 3
Split the node

Now the root is a 4-node, so we need to keep pulling up

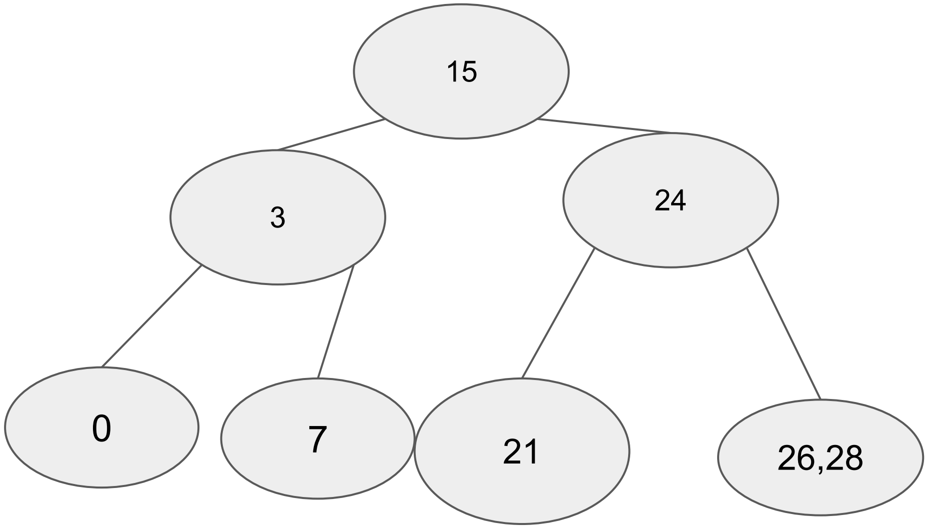
Insert: 15,21,7,24,0,26,3,<u>28</u>,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



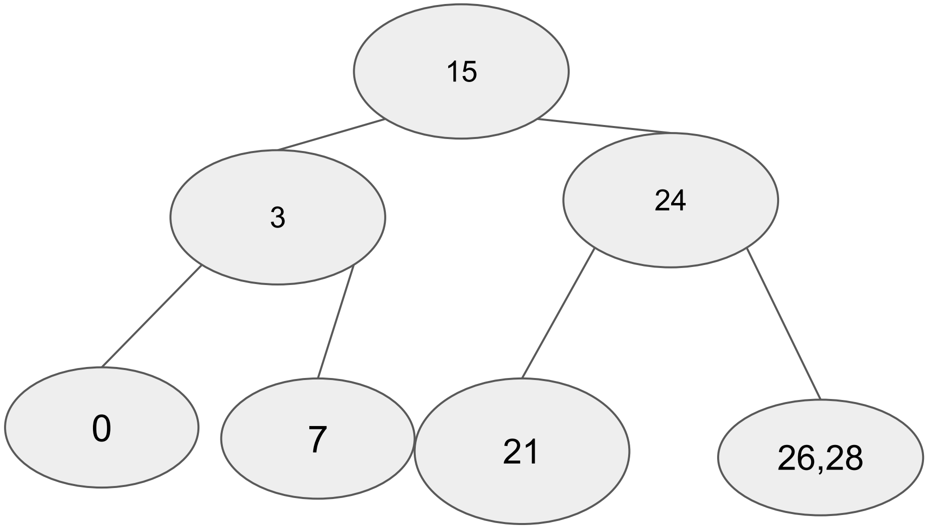
Insert: 15,21,7,24,0,26,3,<u>28</u>,29

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



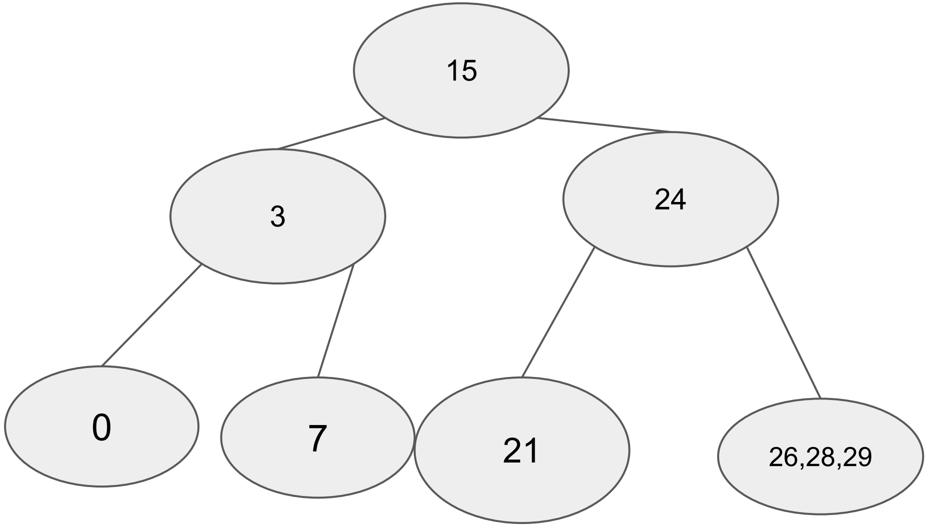
Insert: 15,21,7,24,0,26,3,28,<u>29</u>

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Insert: 15,21,7,24,0,26,3,28,<u>29</u>

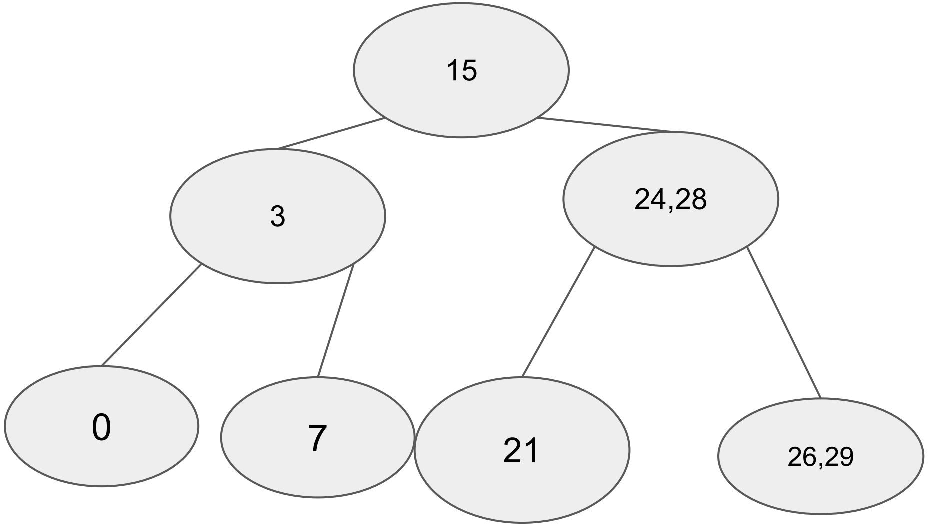
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Pull up by middle

Insert: 15,21,7,24,0,26,3,28,<u>29</u>

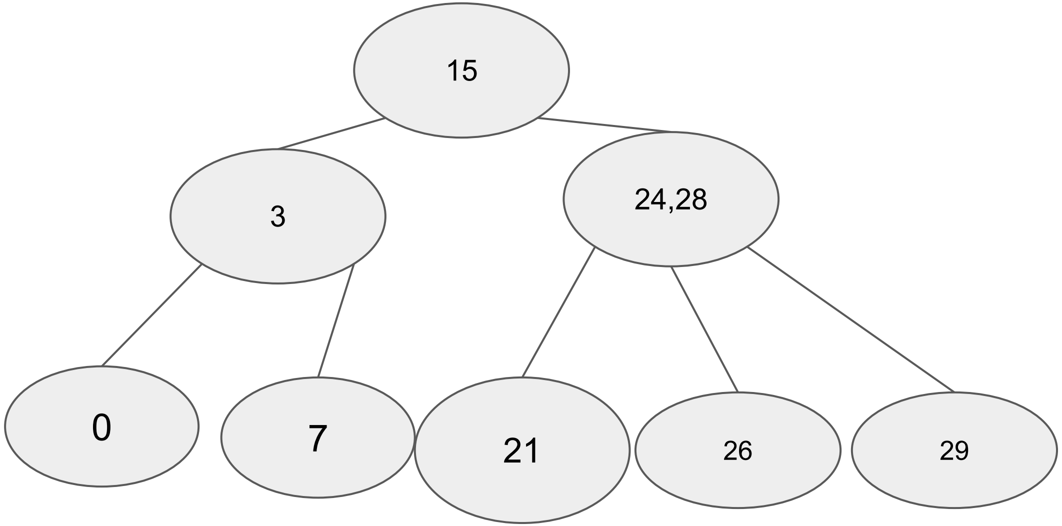
Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



Pull up by middle
Split the node

Insert: 15,21,7,24,0,26,3,28,<u>29</u>

Inserting in 2-3: Find leaf the element would be in, add it and *split* if needed



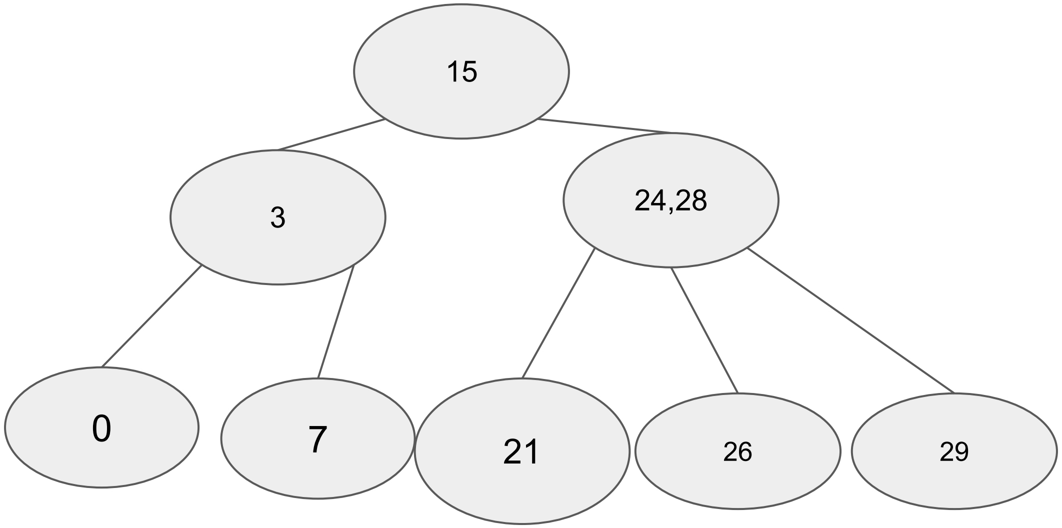
Pull up by middle
Split the node

Question 3

(Deletion) Show intermediate steps of the following questions:

$\boxed{\text{(1) How to delete 7 in the final 2-3 tree of Q1?}}$

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



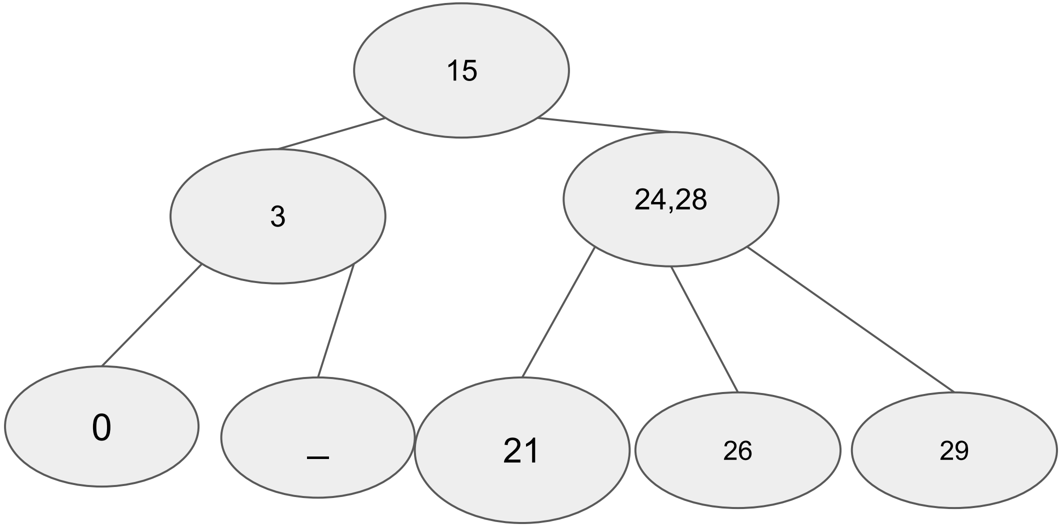
Intuition: I swap the deleted node **up to the root**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



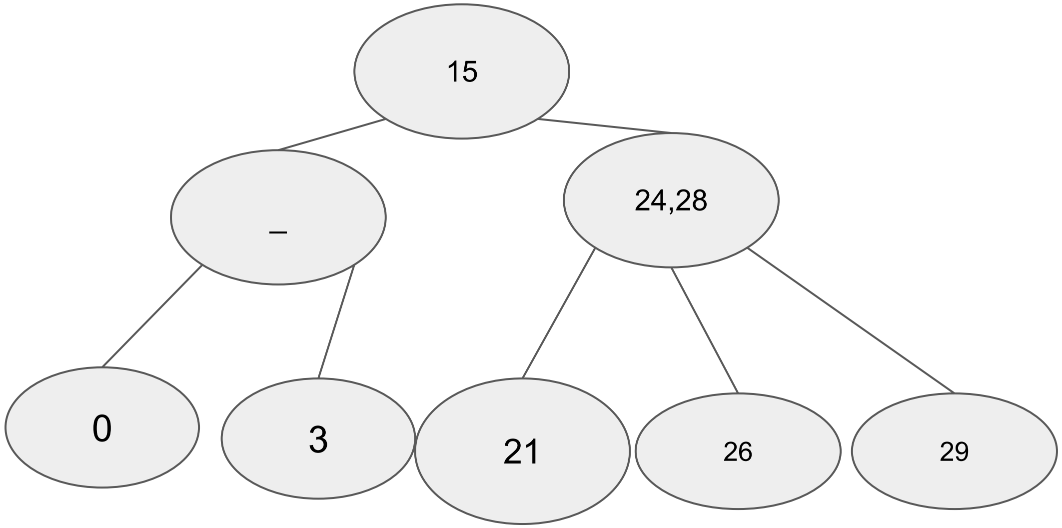
Intuition: I swap the deleted node **up to the root (let _ be deleted)**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



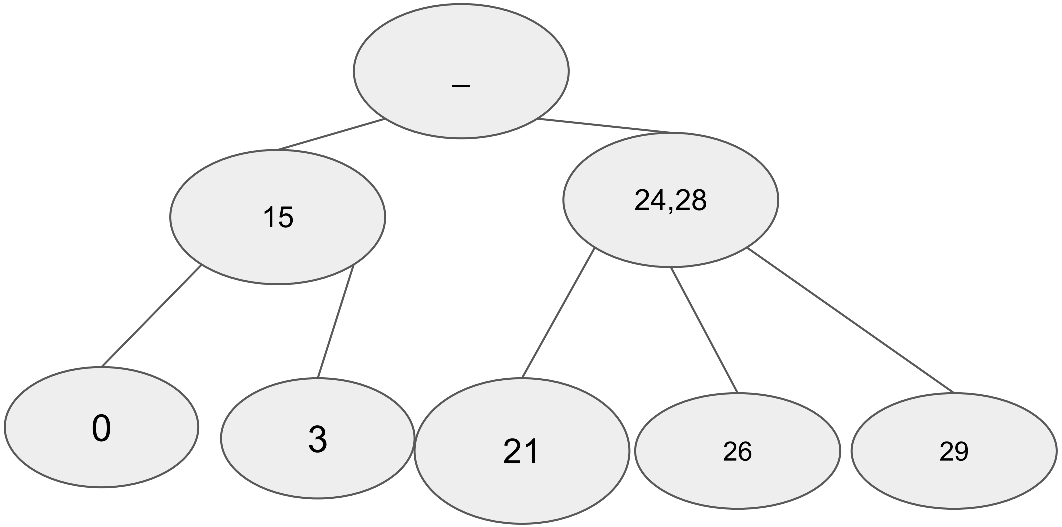
Intuition: I swap the deleted node **up to the root (let _ be deleted)**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



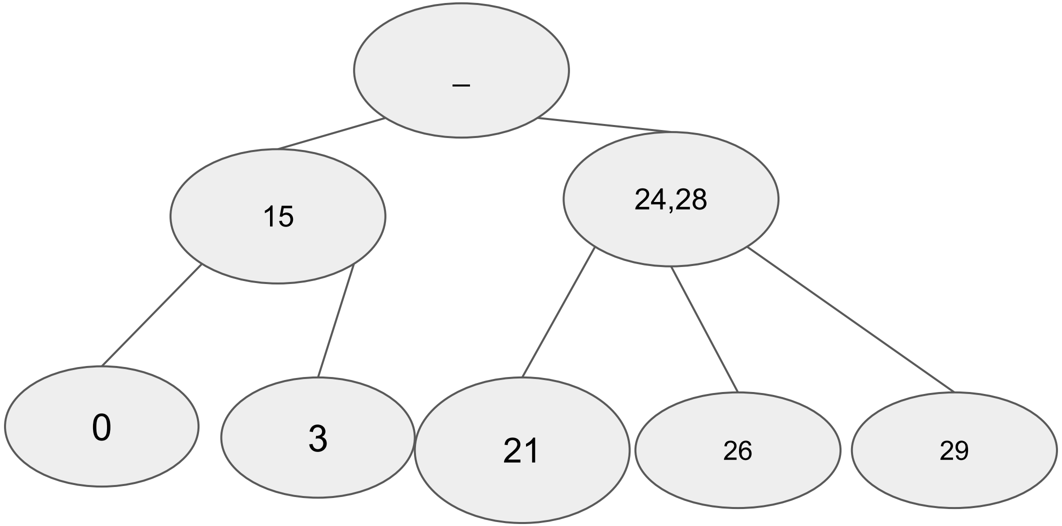
Intuition: I swap the deleted node **up to the root (let _ be deleted)**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

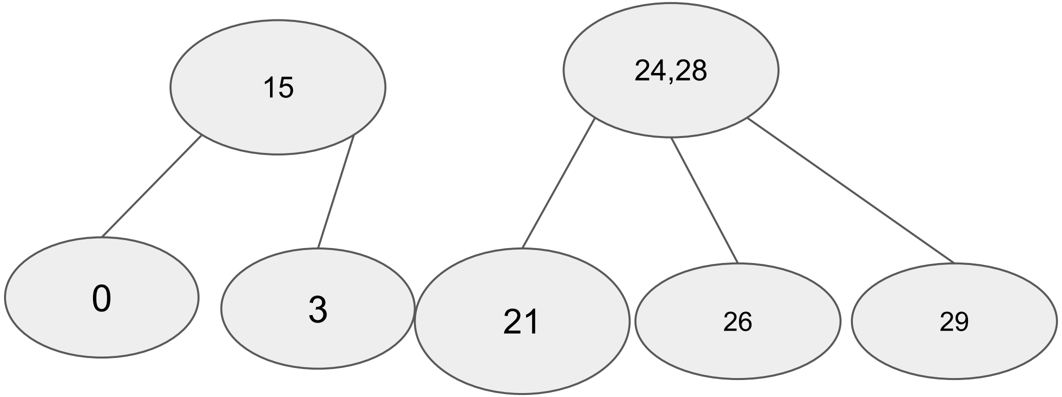
Delete root

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

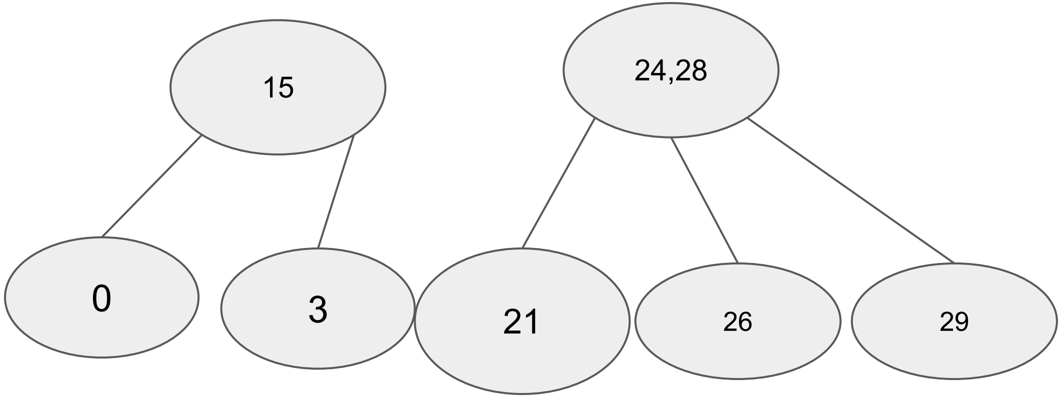
Delete root

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

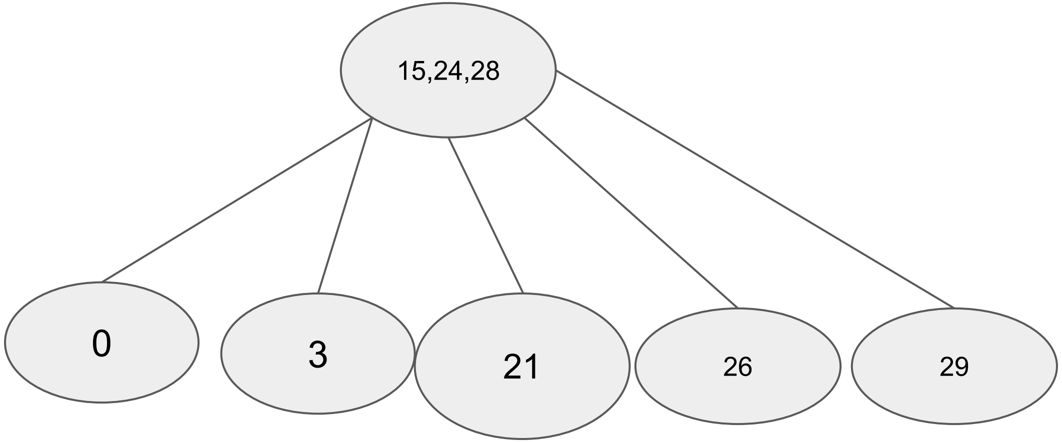
Delete root, merge children going downward

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

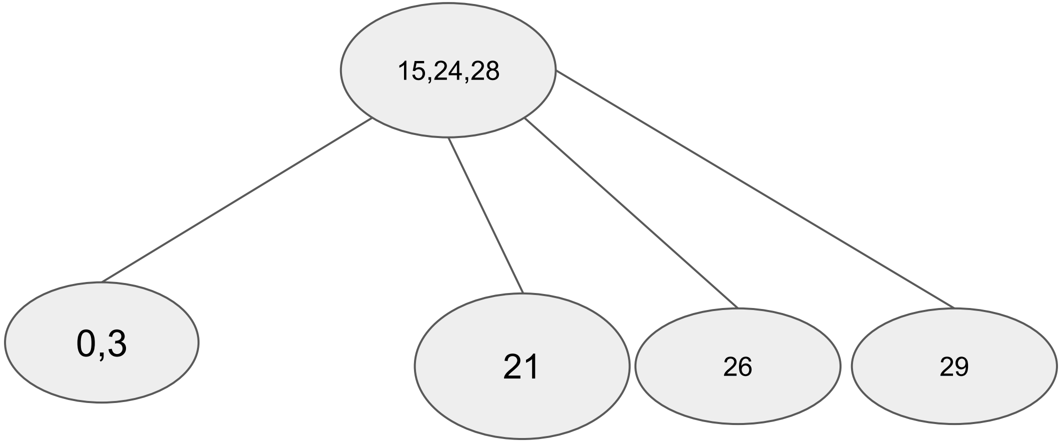
Delete root, merge children going downward

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

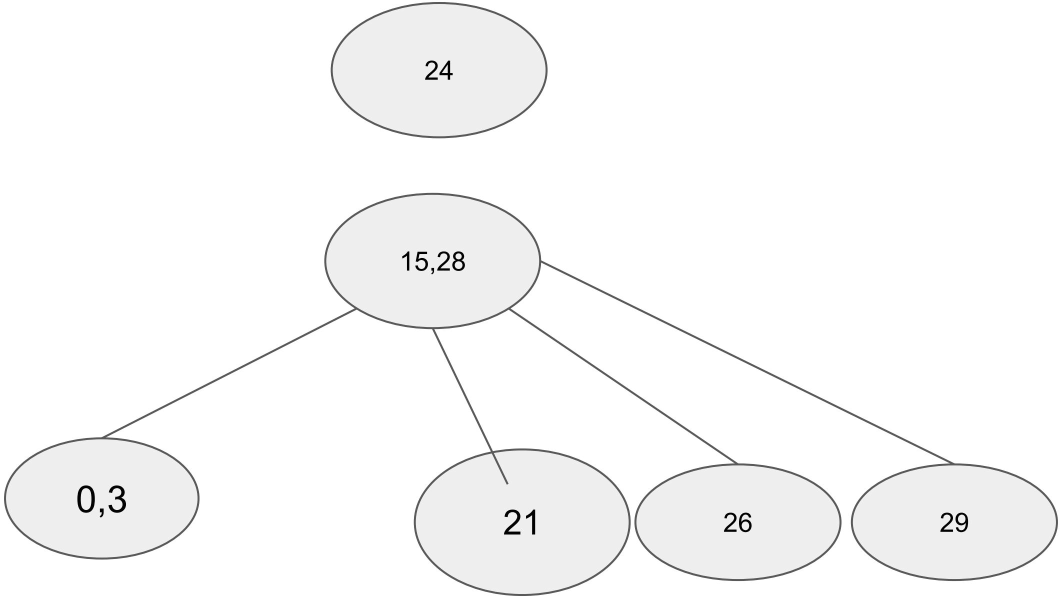
Delete root, merge children going downward, **lastly, fix any 4 node by pulling up**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

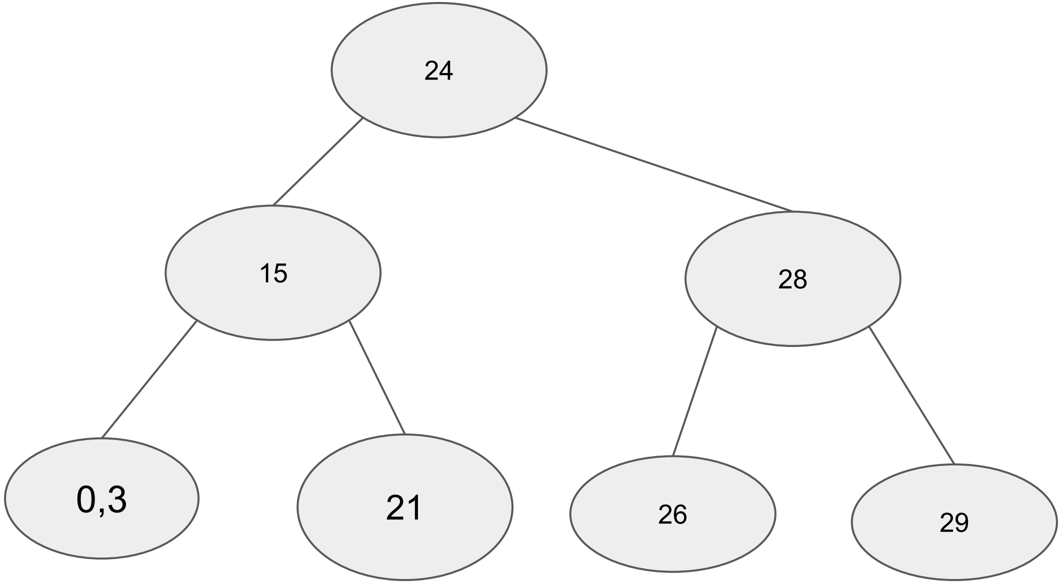
Delete root, merge children going downward, **lastly, fix any 4 node by pulling up**

Question 3

(Deletion) Show intermediate steps of the following questions:

(1) How to delete 7 in the final 2-3 tree of Q1?

(2) How to delete 7 in the final Left–Leaning Red–Black tree of Q1?



Intuition: I swap the deleted node **up to the root (let _ be deleted)**

Delete root, merge children going downward, **lastly, fix any 4 node by pulling up**