

BASIC GRAPH CONCEPTS

DEFINITION

$$G = (V, E)$$

V = set of vertices

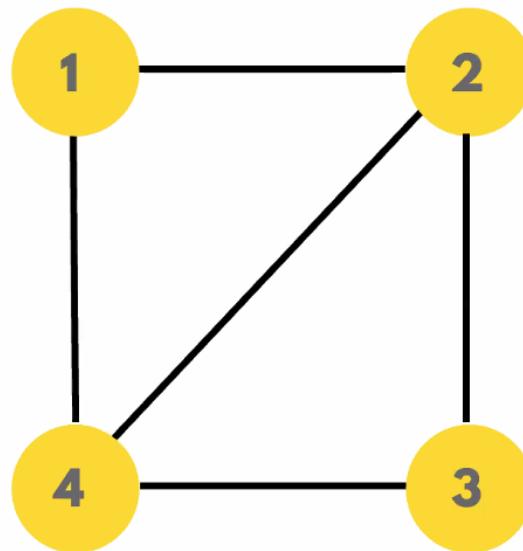
E = set of edges

$$|E| \leq |V^2|$$



BASIC GRAPH CONCEPTS

DEFINITION



$$G = (V, E)$$

$$V = \{1, 2, 3, 4\}$$

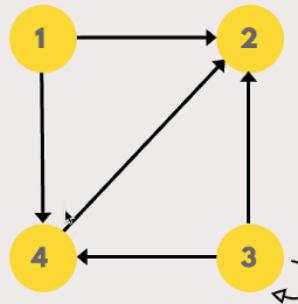
$$E = \{(1,2), (2,3), (2,4), (3,4), (4,1)\}$$

$$|E| \leq |V^2|$$



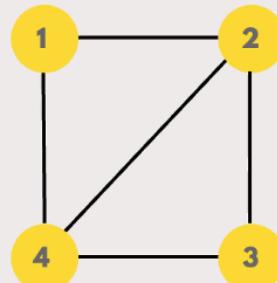
BASIC GRAPH CONCEPTS

DIRECTED VS UNDIRECTED



DIRECTED GRAPH

Self-loop edges are possible only in directed graphs



UNDIRECTED GRAPH

Ashley Hurrelbrink	Lacramioara #
Calin Ovidiu-Raul	Laurentiu Taran
Dobai Lorena	Dobai Lorena
Denis	Alexandra Cretu
Andrei Tatov	Andrei Tatov
Sebi Cozma	Dodenciu Rares
El Kharoubi Iosif	DabuRaul
Vivienne Dumit...	razvan
Vivienne Dumitrescu	razvan



Unmute

Stop Video

Participants
18

Chat

Share Screen

Record

Show Captions

Reactions

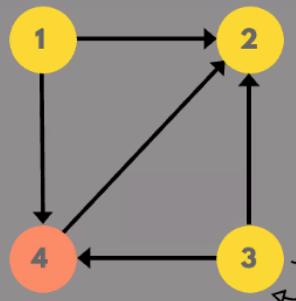
Apps

Leave

BASIC GRAPH CONCEPTS

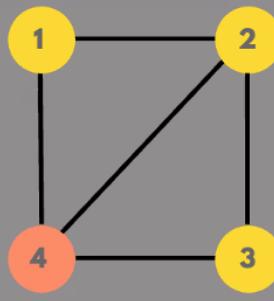
DEGREE OF A VERTEX v

The number of edges adjacent to v



DIRECTED GRAPH

IN-DEGREE = 2
OUT-DEGREE = 1



UNDIRECTED GRAPH

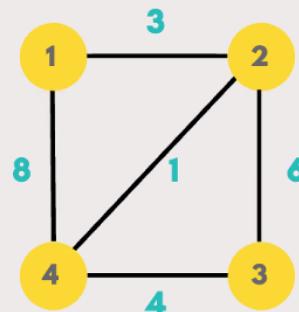
DEGREE = 3

Ashley Hurrelbrink, Lacramioara #, Calin Ovidiu-Raul, Laurentiu Taran, Dobai Lorena, Denis, Dobai Lorena, Alexandra Cretu, Andrei Tatov, Sebi Cozma, Dodenciu Rares, El Kharoubi Iosif, Boldea Patricia-Maria, DabuRaul, Vivienne Dumitrescu, Vivienne Dumitrescu.



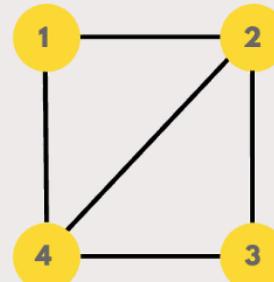
BASIC GRAPH CONCEPTS

WEIGHTED VS UNWEIGHTED



WEIGHTED GRAPH

EACH EDGE HAS AN ASSOCIATED WEIGHT (NUMERICAL VALUE)



UNWEIGHTED GRAPH

Ashley Hurrelbrink	Lacramioara #
Calin Ovidiu-Raul	Laurentiu Taran
Dobai Lorena	Denis
Alexandra Cretu	Andrei Tatov
Sebi Cozma	Dodenciu Rares
El Kharoubi Iosif	Boldea Patricia-Maria
DabuRaul	Vivienne Dumitrescu
Vivienne Dumit...	



BASIC GRAPH CONCEPTS

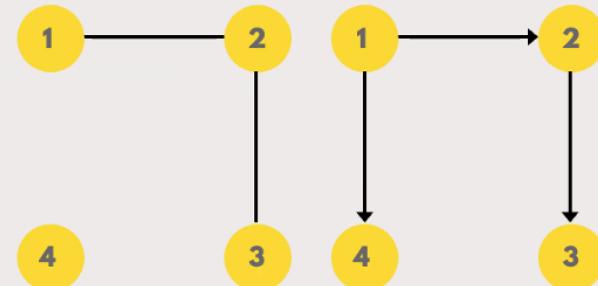
CONNECTED VS DISCONNECTED

CONNECTED

A directed graph is **STRONGLY CONNECTED** if there is a directed path between any two vertices.

An undirected graph is **CONNECTED** if there is a path between any two vertices.

DISCONNECTED



Ashley Hurrelbrink	Lacramioara #
Calin Ovidiu-Raul	Laurentiu Taran
Dobai Lorena	Dobai Lorena
Denis	Alexandra Cretu
Andrei Tatov	Andrei Tatov
Sebi Cozma	Dodenciu Rares
El Kharoubi Iosif	Boldea Patricia-Maria
DabuRaul	Vivienne Dumitrescu

BASIC GRAPH CONCEPTS

DENSE VS SPARSE

DENSE

The number of edges is close to the maximum possible (V^2)

SPARSE

The number of edges is small.





Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

GRAPH REPRESENTATION

1. ADJACENCY MATRIX

2. ADJACENCY LIST



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options



1 GRAPH REPRESENTATION ADJACENCY MATRIX



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



GRAPH REPRESENTATION

ADJACENCY MATRIX

WEIGHTED GRAPH

N*N MATRIX A

A[P, Q] = VALUE
IF THERE IS AN EDGE FROM P TO Q

A[P, Q] = 0
IF THERE IS NOT AN EDGE FROM P TO Q

UNWEIGHTED GRAPH

N*N MATRIX A

A[P, Q] = 1
IF THERE IS AN EDGE FROM P TO Q

A[P, Q] = 0
IF THERE IS NOT AN EDGE FROM P TO Q



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

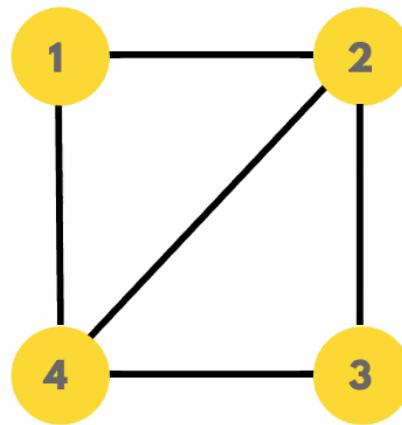
Apps

Leave

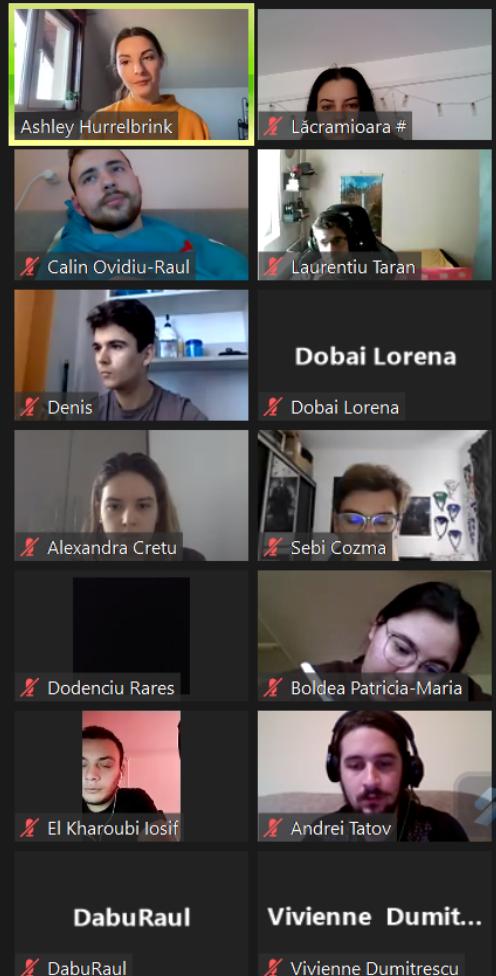
GRAPH REPRESENTATION

ADJACENCY MATRIX

UNDIRECTED GRAPH EXAMPLE



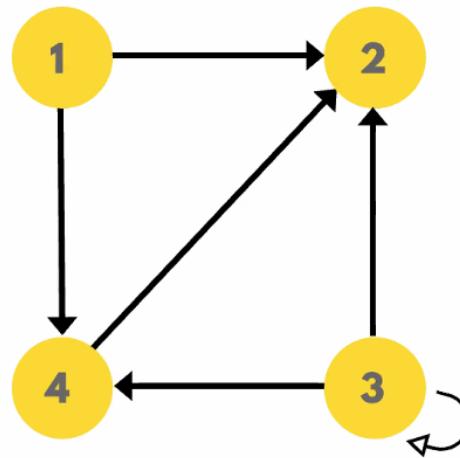
		column			
		1	2	3	4
row	1	0	1	0	1
	2	1	0	1	1
	3	0	1	0	1
	4	1	1	1	0



GRAPH REPRESENTATION

ADJACENCY MATRIX

DIRECTED GRAPH EXAMPLE



column

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



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options



2 GRAPH REPRESENTATION ADJACENCY LIST



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

Apps



Leave

GRAPH REPRESENTATION

ADJACENCY LIST

WEIGHTED GRAPH

FOR EACH VERTEX STORE A
LIST OF VERTICES ADJACENT
TO V AND THE WEIGHT

UNWEIGHTED GRAPH

FOR EACH VERTEX STORE A
LIST OF VERTICES ADJACENT
TO V



Unmute
Stop Video

Participants
19

Chat
^

Share Screen
^

Record
^

Show Captions
^

Reactions
^

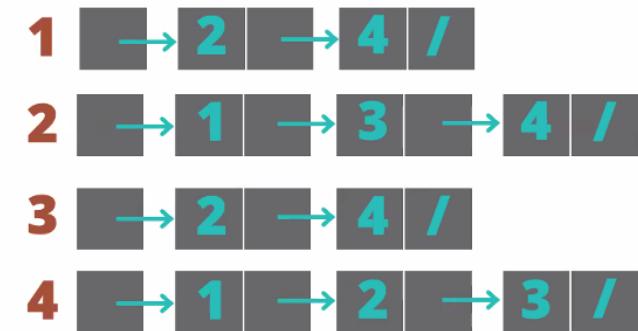
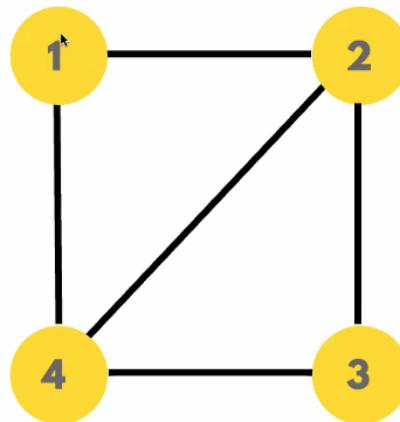
Apps
^

Leave

GRAPH REPRESENTATION

ADJACENCY LIST

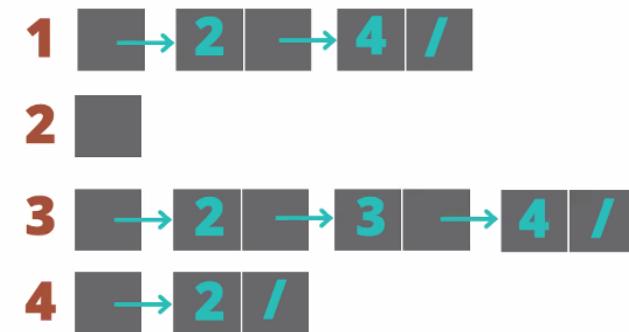
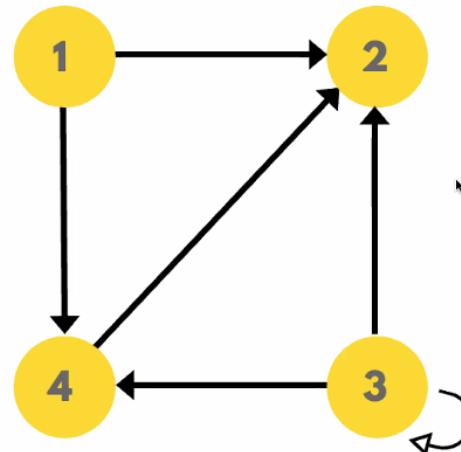
UNDIRECTED GRAPH EXAMPLE



GRAPH REPRESENTATION

ADJACENCY LIST

DIRECTED GRAPH EXAMPLE



Ashley Hurrelbrink	Lacramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul
Laurentiu Taran	Dobai Lorena
Denis	Dobai Lorena
Alexandra Cretu	Sebi Cozma
Dodenciu Rares	Boldea Patricia-Maria
El Kharoubi Iosif	Andrei Tatov
DabuRaul	Vivienne Dumitrescu
	Vivienne Dumitrescu



Recording

You are viewing Ashley Hurrelbrink's screen

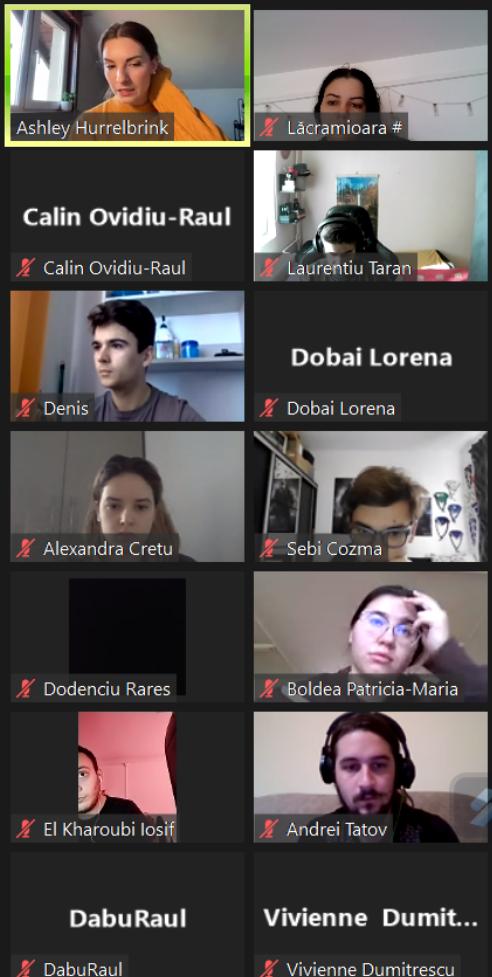
View Options

View

GRAPH REPRESENTATION

1. ADJACENCY MATRIX

2. ADJACENCY LIST



Unmute
Stop Video

Participants
19

Chat

Share Screen

Record

Show Captions

Reactions

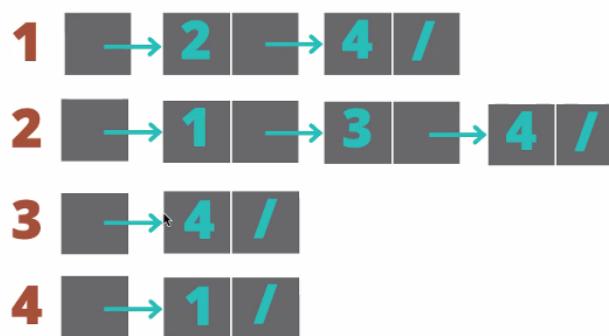
Apps

Leave

GRAPH REPRESENTATION

EXERCISE

Transform the adjacency list to a adjacency matrix



column

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

A grid of 16 video feeds showing participants in a meeting. Each feed includes a name and a small video thumbnail. The names are: Dascalu Alin, Alin Costut, DabuRaul, Calin Ovidiu-Raul, Lăcramioara #, Alexandra Cretu, Dobai Lorena, Sebi Cozma, Andrei Tatov, Boldea Patricia-Maria, Ashley Hurrelbrink, Laurentiu Taran, Denis, and El Kharoubi Iosif.

Dascalu Alin	Alin Costut
DabuRaul	Calin Ovidiu-Raul
Lăcramioara #	Alexandra Cretu
Dobai Lorena	Sebi Cozma
Andrei Tatov	Boldea Patricia-Maria
Ashley Hurrelbrink	Laurentiu Taran
Denis	El Kharoubi Iosif



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View



✋ DabuRaul ✋ Calin Ovidiu-Raul

✖ DabuRaul ✖ Calin Ovidiu-Raul

✖ Lăcramioara # ✖ Alexandra Cretu

✖ Dobai Lorena ✖ Dobai Lorena

✖ Sebi Cozma ✖ Andrei Tatov

✖ Andreea Tatov ✖ Boldea Patricia-Maria

Ashley Hurrelbrink

✖ Laurentiu Taran

✖ Denis ✖ El Kharoubi Iosif

Vivienne Dumit... ✖ Vivienne Dumitrescu

✖ Urs David Krin



Unmute

Stop Video

Participants 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

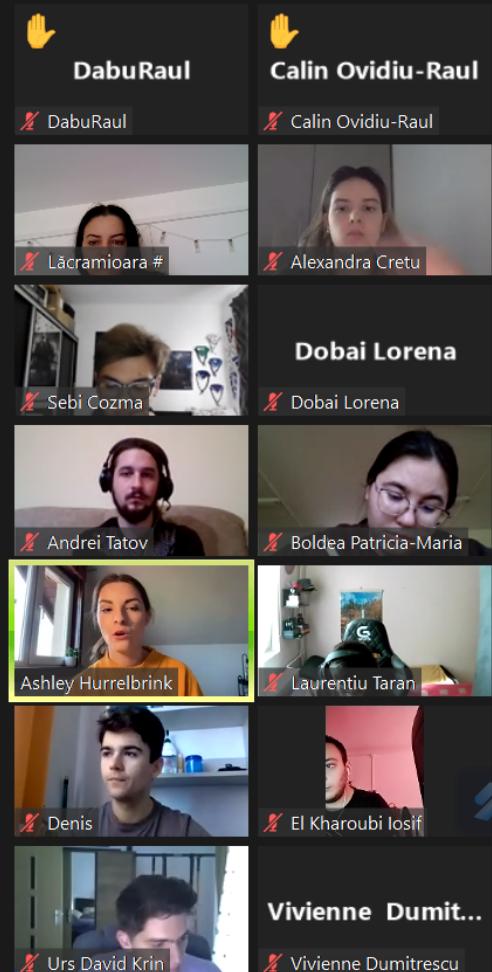


SPANNING TREE

DEFINITION

A **spanning tree** is a **sub-graph** of an **undirected connected** graph, which includes all the **vertices** of the graph with a minimum possible number of **edges**.

THE EDGES MAY OR MAY NOT HAVE WEIGHTS ASSIGNED TO THEM.



Unmute
Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



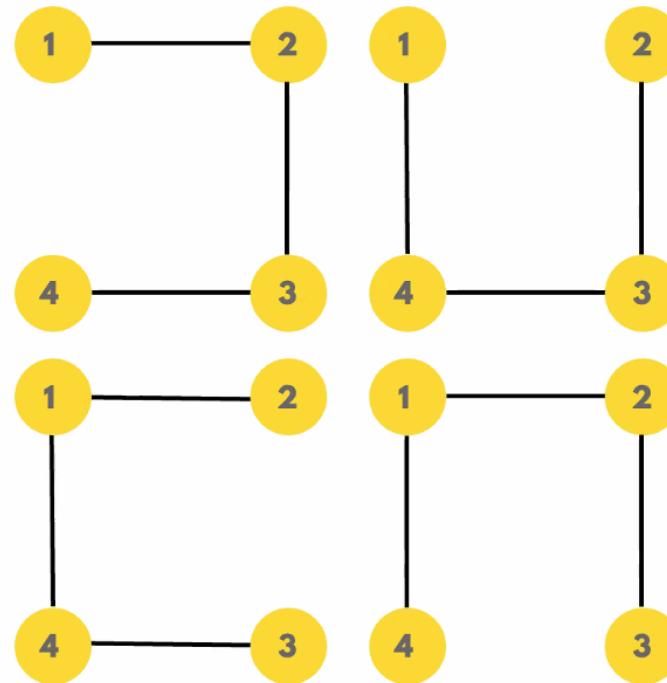
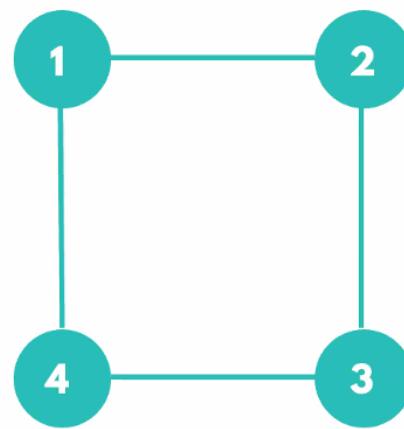
Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

SPANNING TREE EXAMPLE



DabuRaul	Calin Ovidiu-Raul
Lăcramioara #	Alexandra Cretu
Sebi Cozma	Dobai Lorena
Andrei Tatov	Boldea Patricia-Maria
Ashley Hurrelbrink	Laurentiu Taran
Denis	El Kharoubi Iosif
Urs David Krin	Vivienne Dumitrescu



Unmute

Stop Video

Participants 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

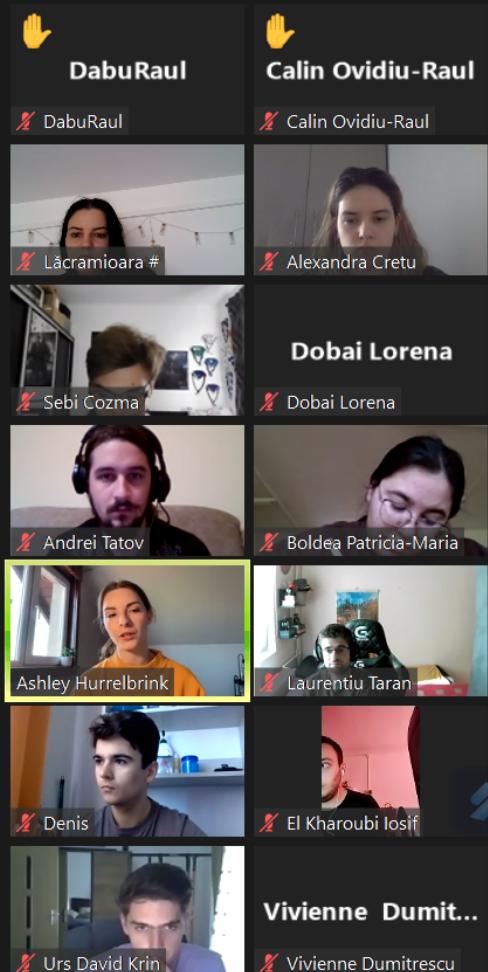
Leave

MINIMUM SPANNING TREE

DEFINITION

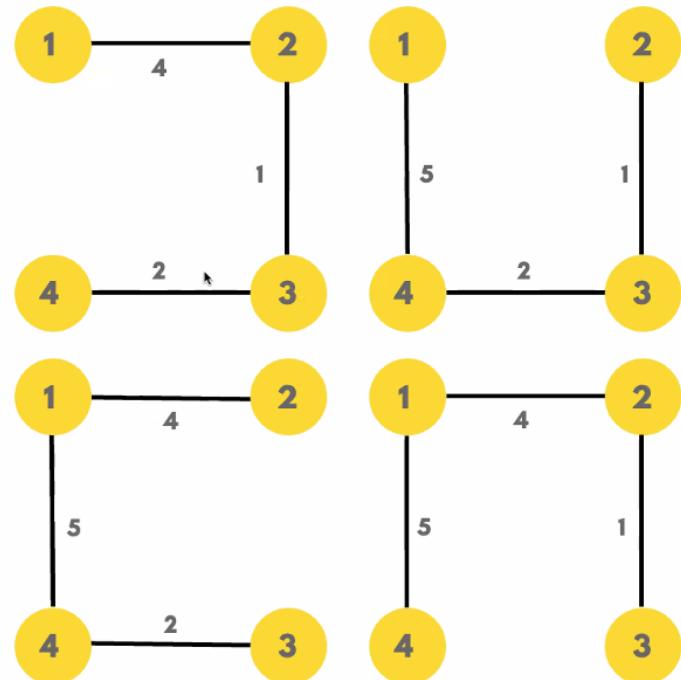
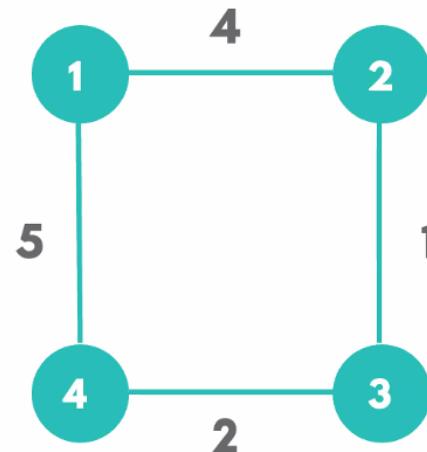
A **minimum spanning tree** is a spanning tree in which the **sum of the weight of the edges (COST)** is as minimum as possible.

THE EDGES MUST HAVE WEIGHTS ASSIGNED TO THEM.



MINIMUM SPANNING TREE

EXAMPLE



You are viewing Ashley Hurrelbrink's screen

Participants: 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

Unmute

Stop Video

DabuRaul

Lăcramioara #

Alexandra Cretu

Dobai Lorena

Sebi Cozma

Dobai Lorena

Andrei Tatov

Boldea Patricia-Maria

Ashley Hurrelbrink

Laurentiu Taran

Denis

El Kharoubi Iosif

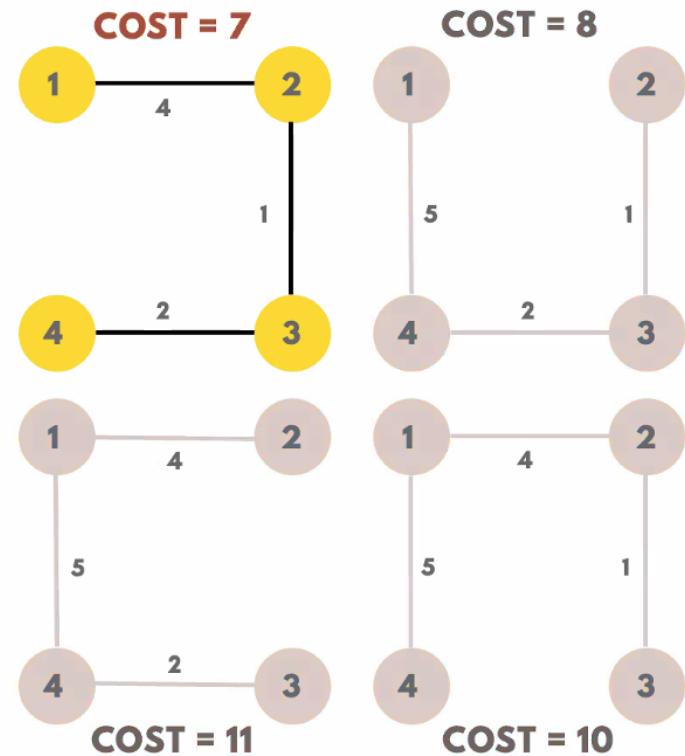
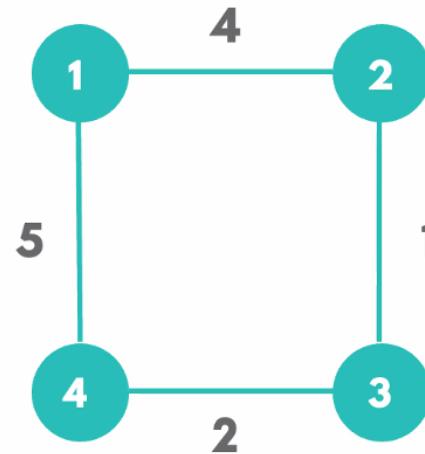
Vivienne Dumit...

Urs David Krin

Vivienne Dumitrescu

MINIMUM SPANNING TREE

EXAMPLE



DabuRaul	Calin Ovidiu-Raul
Lăcramioara #	Alexandra Cretu
Sebi Cozma	Dobai Lorena
Andrei Tatov	Dobai Lorena
Ashley Hurrelbrink	Boldea Patricia-Maria
Denis	Laurentiu Taran
El Kharoubi Iosif	Vivienne Dumitrescu
Urs David Krin	Vivienne Dumitrescu



Unmute

Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

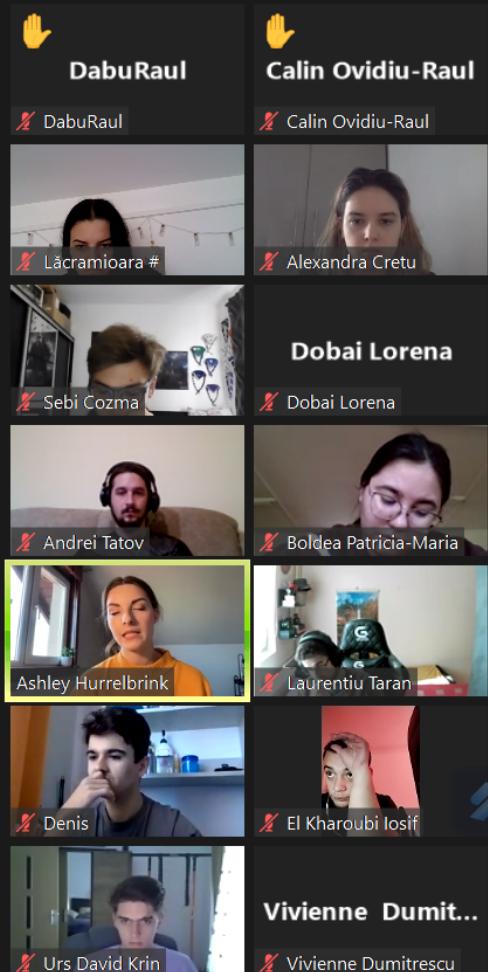
Apps

Leave

MINIMUM SPANNING TREE PROPERTIES

THE MST OF A GRAPH WITH N NODES MUST:

- contain all the n nodes
- contain $N-1$ edges
- provide exactly one path between any two nodes of the graph





Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View



Unmute

Stop Video

Participants

Chat

Share Screen

Record

Show Captions

Reactions

Apps

▼

Leave

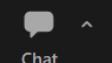
PRIM'S ALGORITHM

STEPS

1 INITIALIZE THE MINIMUM SPANNING TREE WITH A VERTEX CHOSEN AT RANDOM.

2 FIND ALL THE EDGES THAT CONNECT THE TREE TO NEW VERTICES, FIND THE MINIMUM AND ADD IT TO THE TREE.

3 KEEP REPEATING STEP 2 UNTIL WE GET A MINIMUM SPANNING TREE.





Recording

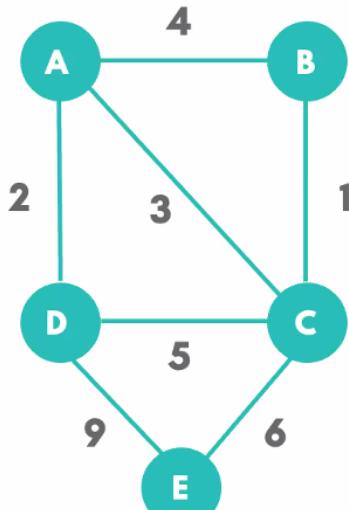
You are viewing Ashley Hurrelbrink's screen

View Options

View

PRIM'S ALGORITHM

STEPS



Participants 20



Chat



Share Screen



Record



Show Captions



Reactions



Apps



Stop Video

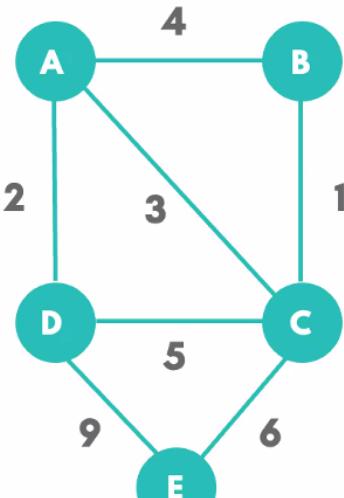


Leave

DabuRaul	DabuRaul	Lacramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul	Alexandra Cretu
Dobai Lorena	Dobai Lorena	Andrei Tatov
Boldea Patricia-Maria	Ashley Hurrelbrink	
Laurentiu Taran	Denis	
El Kharoubi Iosif	Urs David Krin	
Sebi Cozma	Vivienne Dumitrescu	

PRIM'S ALGORITHM

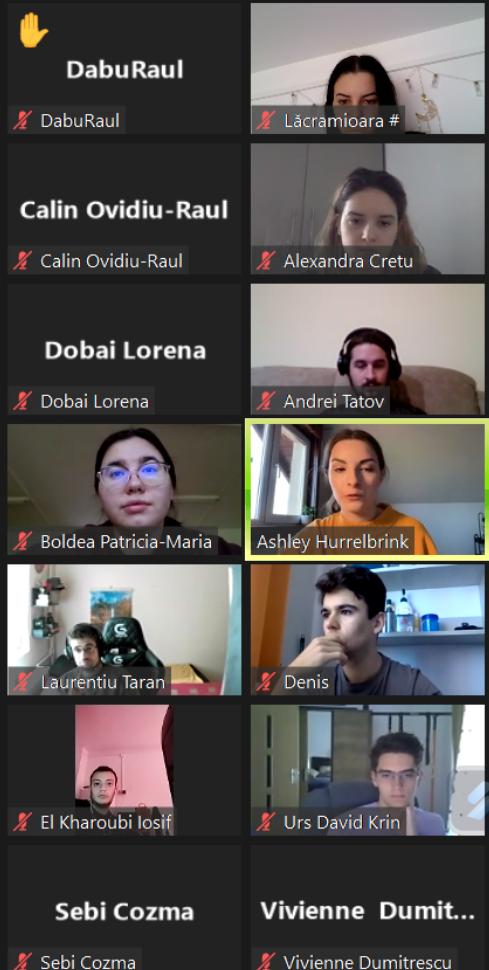
STEPS



MST

ORDER

1 INITIALIZE THE MINIMUM SPANNING TREE WITH A VERTEX CHOSEN AT RANDOM.



Unmute

Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

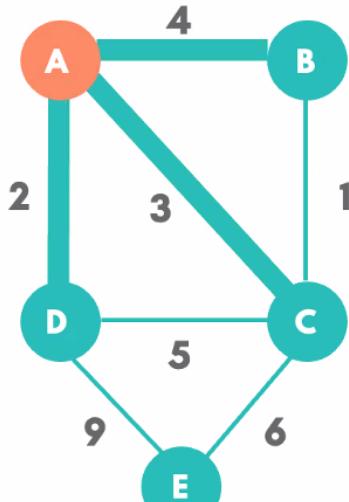
Reactions

Apps

Leave

PRIM'S ALGORITHM

STEPS



MST

{ A }

ORDER

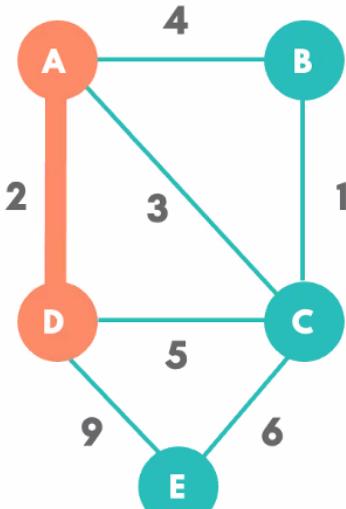
2 FIND ALL THE EDGES THAT CONNECT THE TREE TO NEW VERTICES, FIND THE MINIMUM AND ADD IT TO THE TREE.

DabuRaul	Lăcramioara #
DabuRaul	Alexandra Cretu
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	Andrei Tatov
Dobai Lorena	
Dobai Lorena	Boldea Patricia-Maria
	Ashley Hurrelbrink
	Laurentiu Taran
	Denis
	El Kharoubi Iosif
	Urs David Krin
Sebi Cozma	Vivienne Dumit...
Sebi Cozma	Vivienne Dumitrescu



PRIM'S ALGORITHM

STEPS



MST

{ A, D }

ORDER

(A - D)

2 FIND ALL THE EDGES THAT CONNECT THE TREE TO NEW VERTICES, FIND THE MINIMUM AND ADD IT TO THE TREE.

DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Alexandra Cretu
Dobai Lorena	Andrei Tatov
Boldea Patricia-Maria	Ashley Hurrelbrink
Laurentiu Taran	Denis
Urs David Krin	El Kharoubi Iosif
Sebi Cozma	Vivienne Dumitrescu



Unmute

Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

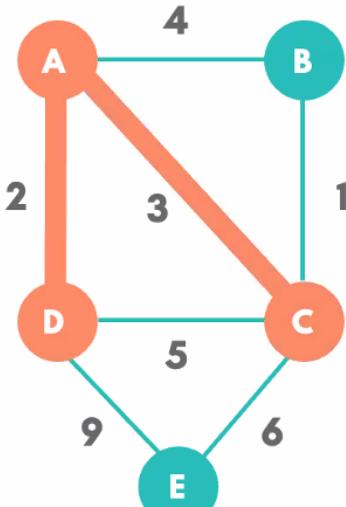
Reactions

Apps

Leave

PRIM'S ALGORITHM

STEPS



MST

{ A, D, C }

ORDER

(A - D)
(A - C)

2 FIND ALL THE EDGES THAT CONNECT THE TREE TO NEW VERTICES, FIND THE MINIMUM AND ADD IT TO THE TREE.



Unmute

Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

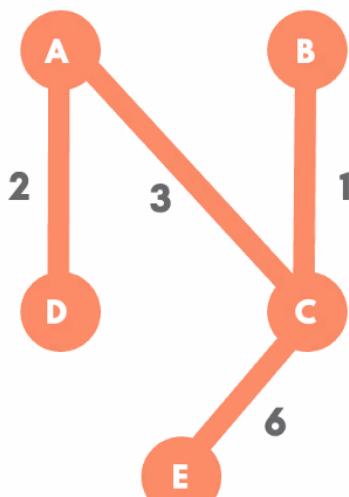
Reactions

Apps

Leave

PRIM'S ALGORITHM

STEPS



MST

{ A, D, C, B, E }

ORDER

(A - D)
(A - C)
(C - B)
(C - E)

3 KEEP REPEATING STEP 2 UNTIL WE GET A MINIMUM SPANNING TREE.

DabuRaul	
DabuRaul	
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	
Dobai Lorena	
Dobai Lorena	
Denis	
Laurentiu Taran	
Boldea Patricia-Maria	
Andrei Tatov	
Urs David Krin	
El Kharoubi Iosif	
Sebi Cozma	
Vivienne Dumitrescu	



Unmute

Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

PRIM'S ALGORITHM IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

Randomly select a vertex v

```
VISIT = {v};  
MST = {};
```

While ($\text{VISIT} \neq V$)

Find a vertex $u \in V - \text{VISIT}$ that connects to a vertex $v \in \text{VISIT}$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - \text{VISIT}$ and $y \in \text{VISIT}$

```
VISIT = VISIT U {u};  
MST = MST U (u, v)
```

EndWhile

Return A

$O(V^2)$

$O(X)$

A screenshot of a video conference interface showing a list of participants and their video feeds. The participants are: DabuRaul, Lacramioara, Calin Ovidiu-Raul, Calin Ovidiu-Raul, Dobai Lorena, Dobai Lorena, Denis, Denis, Laurentiu Taran, Boldea Patricia-Maria, Andrei Tatov, Urs David Krin, El Kharoubi Iosif, El Kharoubi Iosif, Sebi Cozma, Vivienne Dumitrescu, and Sebi Cozma. Ashley Hurrelbrink is highlighted with a yellow box around her video feed.



Unmute
Stop Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

PRIM'S ALGORITHM

BRUTE FORCE

Test all edges

Find a vertex $u \in V - VISIT$ that connects to a vertex $v \in VISIT$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - VISIT$ and $y \in VISIT$

$O(X)$

```
min_weight = infinity  
For each edge (x, y) in E  
    if  $x \in V - VISIT$  &&  $y \in VISIT$  &&  $w(x, y) < min\_weight$   
        u = x;  
        v = y;  
        min_weight = w(x, y);
```

$O(E)$



Stop Video (Alt+V)



Stop Video



20



Chat



Share Screen



Record



Show Captions



Reactions



Apps

A video grid showing 12 participants in a Zoom meeting. The participants are listed on the left with their names and a small video thumbnail on the right.

DabuRaul	Lăcramioara #
DabuRaul	Ashley Hurrelbrink
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	
Dobai Lorena	Alexandra Cretu
Dobai Lorena	
Denis	Laurentiu Taran
Denis	
Boldea Patricia-Maria	Andrei Tatov
Boldea Patricia-Maria	
Urs David Krin	El Kharoubi Iosif
Urs David Krin	
Sebi Cozma	Vivienne Dumitrescu
Sebi Cozma	



Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

PRIM'S ALGORITHM

BRUTE FORCE

Test all edges

Find a vertex $u \in V - VISIT$ that connects to a vertex $v \in VISIT$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - VISIT$ and $y \in VISIT$

$O(X)$

```
min_weight = infinity  
For each edge (x, y) in E  
    if  $x \in V - VISIT$  &&  $y \in VISIT$  &&  $w(x, y) < min\_weight$   
        u = x;  
        v = y;  
        min_weight = w(x, y);
```

$O(V^2)$



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

A video grid showing 12 participants. The participants are listed on the left with their names and a small video thumbnail on the right. The names are: DabuRaul, Lăcramioara #, Calin Ovidiu-Raul, Ashley Hurrelbrink, Dobai Lorena, Alexandra Cretu, Denis, Laurentiu Taran, Boldea Patricia-Maria, Andrei Tatov, Urs David Krin, El Kharoubi Iosif, Sebi Cozma, Vivienne Dumitrescu.

PRIM'S ALGORITHM

BRUTE FORCE IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

Randomly select a vertex v

```
VISIT = {v};  
MST = {};
```

While ($\text{VISIT} \neq V$)

Find a vertex $u \in V - \text{VISIT}$ that connects to a vertex $v \in \text{VISIT}$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - \text{VISIT}$ and $y \in \text{VISIT}$

```
VISIT = VISIT U {u};  
MST = MST U (u, v)
```

EndWhile

Return A

$O(V^3)$

$O(V^2)$

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul
Dobai Lorena	Alexandra Cretu
Denis	Laurentiu Taran
Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	El Kharoubi Iosif
Sebi Cozma	Vivienne Dumitrescu
Ashley Hurrelbrink	



PRIM'S ALGORITHM IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

Find a vertex $u \in V - VISIT$ that connects to a vertex $v \in VISIT$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - VISIT$ and $y \in VISIT$

$O(X)$

BRUTE FORCE

Test all edges

IMPROVED

Keep a list of vertex candidates in an array

BETTER

Keep vertex candidates in a priority queue



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

A video grid showing 12 participants in a Zoom meeting. The participants are listed on the left with their names and a small red icon indicating they are muted. The names are: DabuRaul, Lăcramioara #, Calin Ovidiu-Raul, Calin Ovidiu-Raul, Dobai Lorena, Dobai Lorena, Denis, Denis, Boldea Patricia-Maria, Andrei Tatov, Urs David Krin, El Kharoubi Iosif, Sebi Cozma, and Vivienne Dumitrescu. The participant Ashley Hurrelbrink is highlighted with a yellow border around her video thumbnail.



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

PRIM'S ALGORITHM DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
    d[v] = infinity
    p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
    d[u] = weight(edge(v,u))
    p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
    for loop to search d and select minimum u with d[u]>0

    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0

    //update distance and parent array
    for all edges from u to w
        if(weight(edge(u,w)) < d[w])
            d[w] = weight(edge(u, w))
            p[w] = u;
```

O(?)



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Dobai Lorena
Calin Ovidiu-Raul	Alexandra Cretu
Denis	Laurentiu Taran
Denis	Andrei Tatov
Boldea Patricia-Maria	El Kharoubi Iosif
Urs David Krin	Vivienne Dumitrescu
Sebi Cozma	
	Vivienne Dumitrescu

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
    d[v] = infinity
    p[v] = null
d[v] = 0 //v is randomly chosen
//update distance and parent array
For all edges from v to u
    d[u] = weight(edge(v,u))
    p[u] = v
VISIT = {v}
MST = {}
While (VISIT != V).
    for loop to search d and select minimum u with d[u]>0 → O(V^2)
        VISIT = VISIT U {u}
        MST = MST U {edge(p[u], u)}
        d[u] = 0
//update distance and parent array
for all edges from u to w
    if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```

O(?)



Unmute

Start Video

Participants 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

PRIM'S ALGORITHM DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
    d[v] = infinity
    p[v] = null

d[v] = 0 //v is randomly chosen

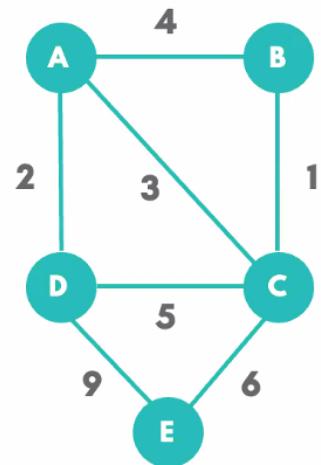
//update distance and parent array
For all edges from v to u
    d[u] = weight(edge(v,u))
    p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
    for loop to search d and select minimum u with d[u]>0
        VISIT = VISIT U {u}
        MST = MST U {edge(p[u], u)}

        d[u] = 0;

//update distance and parent array
for all edges from u to w
    if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



	VISIT	MST	A	B	C	D	E
d			inf	inf	inf	inf	inf
p			null	null	null	null	null

Participant	Video Status
DabuRaul	Recording
Lăcramioara #	Recording
Calin Ovidiu-Raul	Recording
Dobai Lorena	Recording
Denis	Recording
Laurentiu Taran	Recording
Boldea Patricia-Maria	Recording
Andrei Tatov	Recording
Urs David Krin	Recording
El Kharoubi Iosif	Recording
Sebi Cozma	Recording
Vivienne Dumitrescu	Recording

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
d[v] = infinity
p[v] = null

d[v] = 0 //v is randomly chosen

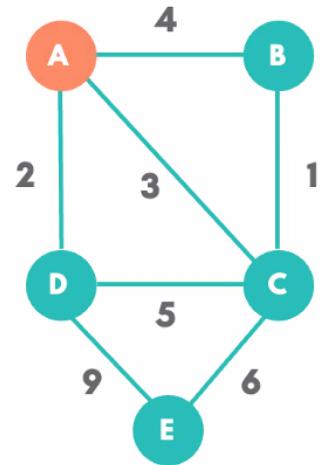
//update distance and parent array
For all edges from v to u
d[u] = weight(edge(v,u))
p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
    for loop to search d and select minimum u with d[u]>0
        VISIT = VISIT U {u}
        MST = MST U {edge(p[u], u)}

        d[u] = 0;

//update distance and parent array
for all edges from u to w
if(weight(edge(u,w)) < d[w])
    d[w] = weight(edge(u, w))
    p[w] = u;
```



	A	B	C	D	E
d	0	inf	inf	inf	inf
p	null	null	null	null	null

DabuRaul	#	Lăcramioara #
DabuRaul		Lăcramioara #
Calin Ovidiu-Raul		Ashley Hurrelbrink
Calin Ovidiu-Raul		
Dobai Lorena		
Dobai Lorena		Alexandra Cretu
Denis		
Denis		Laurentiu Taran
Boldea Patricia-Maria		
Boldea Patricia-Maria		Andrei Tatov
El Kharoubi Iosif		
Urs David Krin		El Kharoubi Iosif
Sebi Cozma		
Sebi Cozma		Vivienne Dumitrescu
Vivienne Dumitrescu		



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

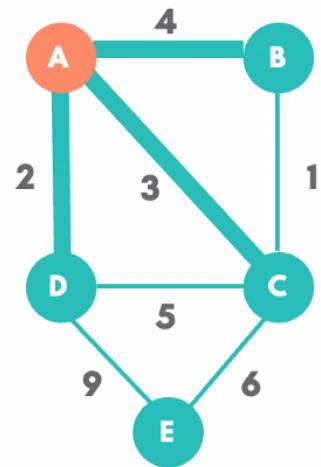
VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0

    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT

MST

	A	B	C	D	E
d	0	4	3	2	inf
p	null	A	A	A	null

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

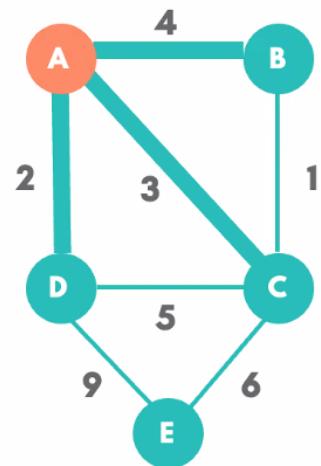
VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0

    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A}

MST { }

	A	B	C	D	E
d	0	4	3	2	inf
p	null	A	A	A	null

Hand icon: DabuRaul

Lăcramioara #

DabuRaul

Lăcramioara #

Calin Ovidiu-Raul

Calin Ovidiu-Raul

Dobai Lorena

Dobai Lorena

Denis

Denis

Boldea Patricia-Maria

Laurentiu Taran

El Kharoubi Iosif

El Kharoubi Iosif

Sebi Cozma

Vivienne Dumitrescu

Sebi Cozma

Vivienne Dumitrescu

Urs David Krin

Andrei Tatov

El Kharoubi Iosif

El Kharoubi Iosif

Sebi Cozma

Vivienne Dumitrescu

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

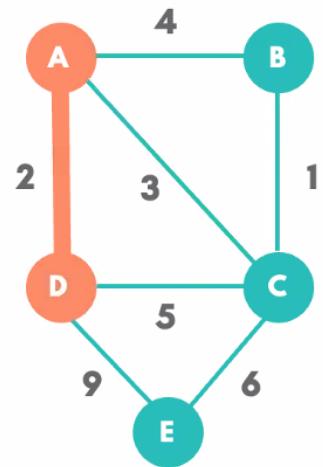
//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0
    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D}

MST {A-D}

	A	B	C	D	E
d	0	4	3	2	inf
p	null	A	A	A	null



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

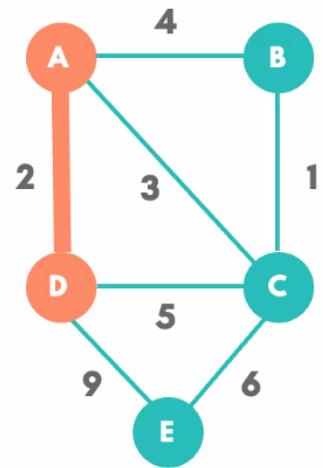
VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0

    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D}

MST {A-D}

	A	B	C	D	E
d	0	4	3	0	inf
p	null	A	A	A	null

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	Alexandra Cretu
Denis	Laurentiu Taran
Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	El Kharoubi Iosif
Sebi Cozma	Vivienne Dumitrescu



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

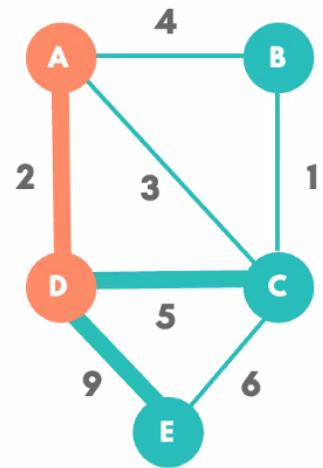
//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0
    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D}

MST {A-D}

	A	B	C	D	E
d	0	4	3	0	9
p	null	A	A	A	E

DabuRaul	DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	Dobai Lorena	Alexandra Cretu
Denis	Denis	Laurentiu Taran
Boldea Patricia-Maria	Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	Urs David Krin	El Kharoubi Iosif
Sebi Cozma	Sebi Cozma	Vivienne Dumitrescu



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

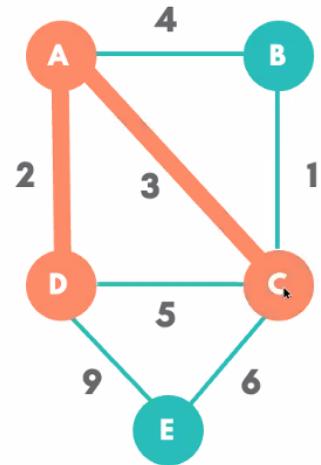
//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0
    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

  d[u] = 0;

//update distance and parent array
for all edges from u to w
  if(weight(edge(u,w)) < d[w])
    d[w] = weight(edge(u, w))
    p[w] = u;
```



VISIT {A, D, C}

MST {A-D, A-C}

	A	B	C	D	E
d	0	4	3	0	9
p	null	A	A	A	E

DabuRaul	Lacramioara #
DabuRaul	Lacramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Calin Ovidiu-Raul	
Dobai Lorena	Alexandra Cretu
Dobai Lorena	
Denis	Laurentiu Taran
Denis	
Boldea Patricia-Maria	Andrei Tatov
Boldea Patricia-Maria	
Urs David Krin	El Kharoubi Iosif
Urs David Krin	
Sebi Cozma	Vivienne Dumitrescu
Sebi Cozma	



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

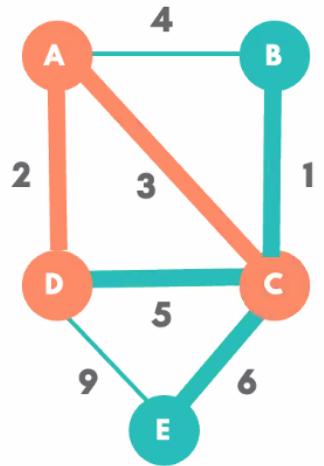
VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0

    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D, C}

MST {A-D, A-C}

	A	B	C	D	E
d	0	1	0	0	6
p	null	C	A	A	C

DabuRaul	DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	Dobai Lorena	Alexandra Cretu
Denis	Denis	Laurentiu Taran
Boldea Patricia-Maria	Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	Urs David Krin	El Kharoubi Iosif
Sebi Cozma	Sebi Cozma	Vivienne Dumitrescu

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
    d[v] = infinity
    p[v] = null

d[v] = 0 //v is randomly chosen

//update distance and parent array
For all edges from v to u
    d[u] = weight(edge(v,u))
    p[u] = v

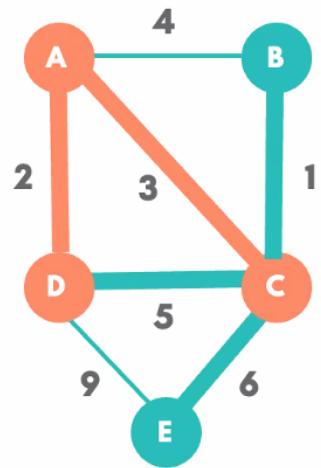
VISIT = {v}
MST = {}

While (VISIT != V).
    for loop to search d and select minimum u with d[u]>0

        VISIT = VISIT U {u}
        MST = MST U {edge(p[u], u)}

        d[u] = 0;

//update distance and parent array
for all edges from u to w
    if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D, C}

MST {A-D, A-C}

	A	B	C	D	E
d	0	1	0	0	6
p	null	C	A	A	C

DabuRaul	# DabuRaul	# Lăcrămioara #
Calin Ovidiu-Raul	# Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	# Dobai Lorena	Alexandra Cretu
Denis	# Denis	Laurentiu Taran
Boldea Patricia-Maria	# Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	# Urs David Krin	El Kharoubi Iosif
Sebi Cozma	# Sebi Cozma	Vivienne Dumitrescu



PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

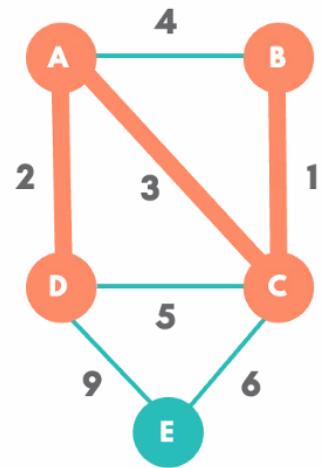
//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0
    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT { A , D , C , B }

MST { A-D , A-C , C-B }

	A	B	C	D	E
d	0	0	0	0	6
p	null	C	A	A	C

Unmute My Audio (Alt+A). Or you can simply press and hold the space bar to temporarily unmute.

Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

PRIM'S ALGORITHM

DISTANCE ARRAY

Keep a list of vertex candidates in an array

```
//set distance and parent array
For each v in V
  d[v] = infinity
  p[v] = null

d[v] = 0 //v is randomly chosen

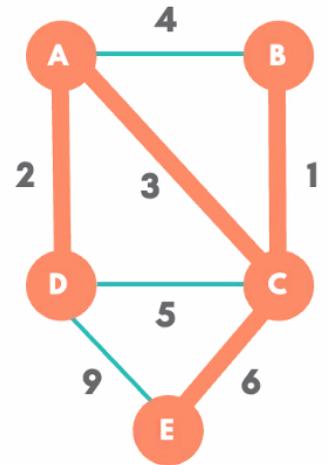
//update distance and parent array
For all edges from v to u
  d[u] = weight(edge(v,u))
  p[u] = v

VISIT = {v}
MST = {}

While (VISIT != V).
  for loop to search d and select minimum u with d[u]>0
    VISIT = VISIT U {u}
    MST = MST U {edge(p[u], u)}

    d[u] = 0;

    //update distance and parent array
    for all edges from u to w
      if(weight(edge(u,w)) < d[w])
        d[w] = weight(edge(u, w))
        p[w] = u;
```



VISIT {A, D, C, B, E}

MST {A-D, A-C, C-B, C-E}

	A	B	C	D	E
d	0	0	0	0	0
p	null	C	A	A	C

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	
Dobai Lorena	Alexandra Cretu
Denis	
Denis	Laurentiu Taran
Boldea Patricia-Maria	
Boldea Patricia-Maria	Andrei Tatov
El Kharoubi Iosif	
El Kharoubi Iosif	Urs David Krin
Sebi Cozma	Vivienne Dumit...
Sebi Cozma	Vivienne Dumitrescu



PRIM'S ALGORITHM IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

Find a vertex $u \in V - VISIT$ that connects to a vertex $v \in VISIT$ such that $w(u, v) \leq w(x, y)$, for any $x \in V - VISIT$ and $y \in VISIT$

$O(X)$

BRUTE FORCE

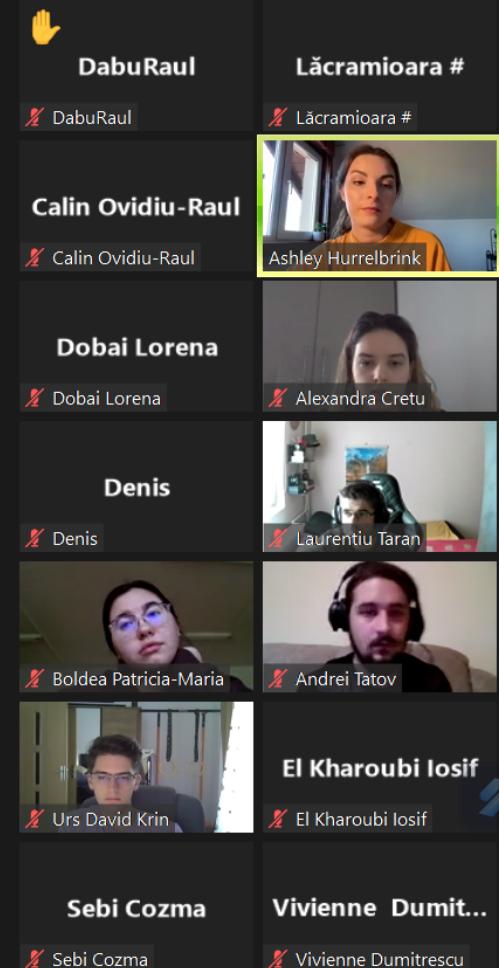
Test all edges

IMPROVED

Keep a list of vertex candidates in an array

BETTER

Keep vertex candidates in a priority queue



PRIM'S ALGORITHM

PRIORITY QUEUE

Keep a list of vertex candidates in an array

```
//set priority queue, key and parent array
For each u in Q → O(V)
    Q.insert(u)
    key[u] = infinity
    p[u] = null

DECREASE_KEY(v, 0) //v is randomly chosen

VISIT = { v }
MST = { }

While (Q not Empty). → O(V)
    u = Q.extractMin(); → O(log V)
    //update key and parent array
    for all edges from u to w → O(V)
        if(Q.contains(w) && weight(edge(u,w)) < key[w])
            DECREASE_KEY(w, weight(edge(u, w))) → O(log V)
            p[w] = u;
```

$O(V^2 \log V)$



Unmute

Start Video

Participants 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

PRIM'S ALGORITHM IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

DISTANCE ARRAY

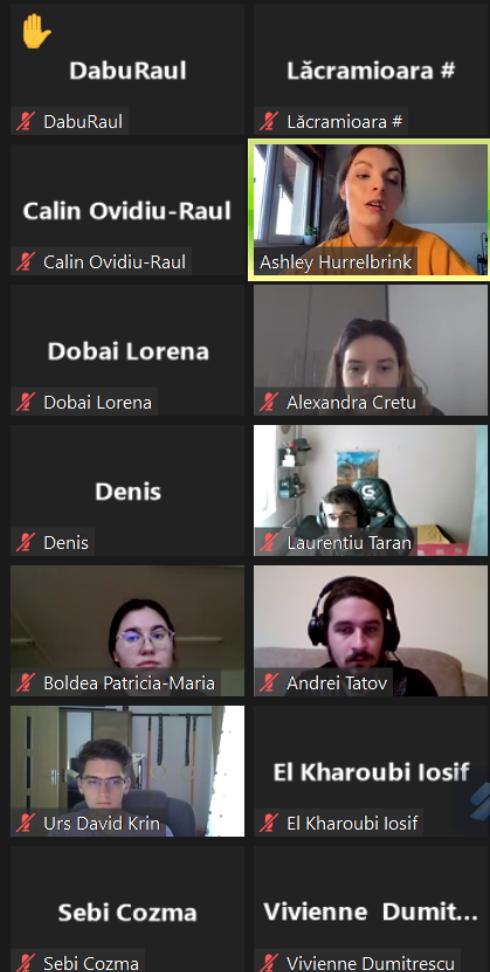
$O(V^2)$

Good for dense graphs

PRIORITY QUEUE

$O(E \log V)$

Good for sparse graphs



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

3

REVIEW KRUSKAL'S ALGORITHM



Unmute



Start Video



Participants



Chat



Share Screen



Record



Show Captions



Reactions



Apps



Leave

DabuRaul	DabuRaul	Lăcramioara #	Lăcramioara #
Calin Ovidiu-Raul	Calin Ovidiu-Raul	Ashley Hurrelbrink	Ashley Hurrelbrink
Dobai Lorena	Dobai Lorena	Alexandra Cretu	Alexandra Cretu
Denis	Denis	Laurentiu Taran	Laurentiu Taran
Boldea Patricia-Maria	Boldea Patricia-Maria	Andrei Tatov	Andrei Tatov
Urs David Krin	Urs David Krin	El Kharoubi Iosif	El Kharoubi Iosif
Sebi Cozma	Sebi Cozma	Vivienne Dumitrescu	Vivienne Dumitrescu



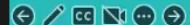
Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

1 KRUSKAL'S ALGORITHM STEPS



Unmute
Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

Hand icon: DabuRaul

Lăcramioara #

DabuRaul

Lăcramioara #

Calin Ovidiu-Raul

Calin Ovidiu-Raul

Dobai Lorena

Dobai Lorena

Denis

Denis

Boldea Patricia-Maria

Andrei Tatov

El Kharoubi Iosif

Urs David Krin

El Kharoubi Iosif

Sebi Cozma

Vivienne Dumitrescu

Sebi Cozma

Vivienne Dumitrescu

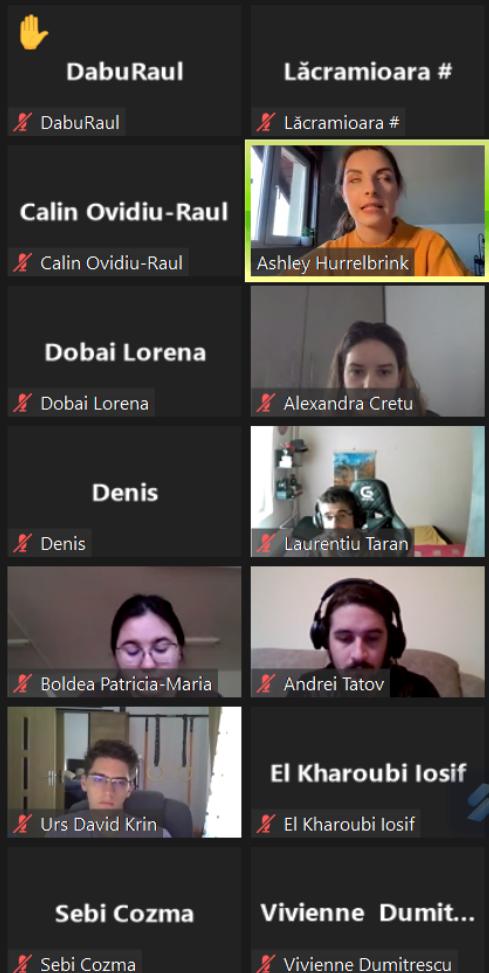
KRUSKAL'S ALGORITHM

STEPS

- 1** Sort all the edges in non-decreasing order of their weight.

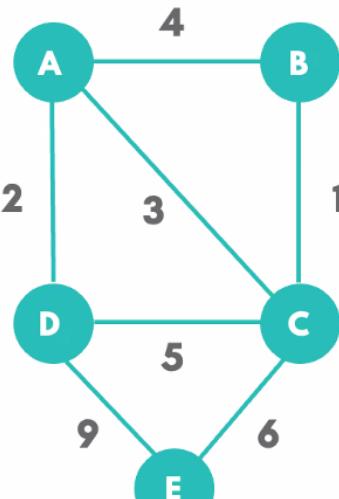
- 2** PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

- 3** REPEAT STEP#2 UNTIL THERE ARE $(V-1)$ EDGES IN THE SPANNING TREE.



KRUSKAL'S ALGORITHM

STEPS



MST

{ A }, { B }, { C }, { D }, { E }

ORDER

- Sort all the edges in non-decreasing order of their weight.

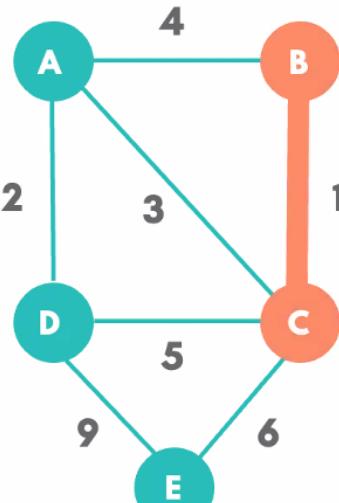
(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
(A-B) --> 4
(D-C) --> 5
(C-E) --> 6
(D-E) --> 9

A list of participants in a video conference:

- DabuRaul
- Lăcramioara #
- Calin Ovidiu-Raul
- Dobai Lorena
- Denis
- Boldea Patricia-Maria
- Urs David Krin
- Sebi Cozma
- El Kharoubi Iosif
- Vivienne Dumitrescu

KRUSKAL'S ALGORITHM

STEPS



MST

{ A }, { B, C }, { D }, { E }

ORDER

(B - C)

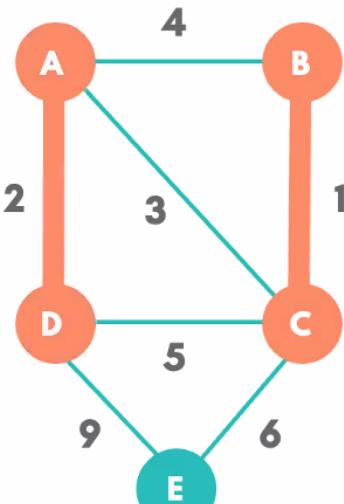
2 PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
(A-B) --> 4
(D-C) --> 5
(C-E) --> 6
(D-E) --> 9

DabuRaul
Lăcramioara #
Calin Ovidiu-Raul
Dobai Lorena
Denis
Boldea Patricia-Maria
Urs David Krin
Sebi Cozma
El Kharoubi Iosif
Vivienne Dumitrescu
Ashley Hurrelbrink

KRUSKAL'S ALGORITHM

STEPS



MST

{ A , D } , { B , C } , { E }

ORDER

(B - C)
(A - D)

2 PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
(A-B) --> 4
(D-C) --> 5
(C-E) --> 6
(D-E) --> 9

DabuRaul
Lăcramioara #
Calin Ovidiu-Raul
Dobai Lorena
Denis
Boldea Patricia-Maria
Urs David Krin
El Kharoubi Iosif
Vivienne Dumitrescu

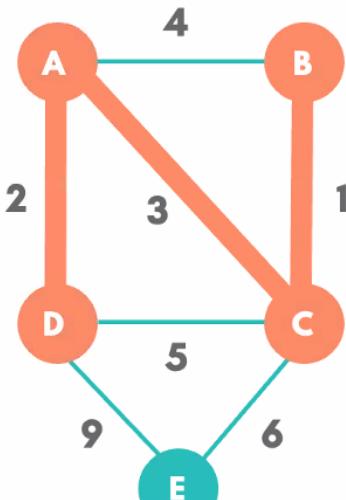
DabuRaul
Lăcramioara #
Calin Ovidiu-Raul
Dobai Lorena
Denis
Boldea Patricia-Maria
Urs David Krin
El Kharoubi Iosif

Ashley Hurrelbrink

Alexandra Cretu
Laurentiu Taran
Andrei Tatov
Sebi Cozma
Vivienne Dumitrescu

KRUSKAL'S ALGORITHM

STEPS



MST

{ A , D , B , C } , { E }

ORDER

(B - C)
(A - D)
(A - C)

2 PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
(A-B) --> 4
(D-C) --> 5
(C-E) --> 6
(D-E) --> 9

Hand	Participant Name	Video Preview
Up	DabuRaul	
Up	DabuRaul	
Up	Lăcramioara #	
Up	Lăcramioara #	
Up	Calin Ovidiu-Raul	
Up	Calin Ovidiu-Raul	
Up	Dobai Lorena	
Up	Dobai Lorena	
Up	Alexandra Cretu	
Up	Denis	
Up	Denis	
Up	Laurentiu Taran	
Up	Boldea Patricia-Maria	
Up	Andrei Tatov	
Up	Sebi Cozma	
Up	Sebi Cozma	
Up	El Kharoubi Iosif	
Up	Vivienne Dumitrescu	



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

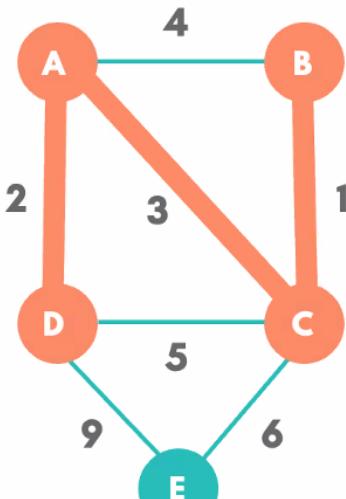
Reactions

Apps

Leave

KRUSKAL'S ALGORITHM

STEPS



MST

{ A , D , B , C } , { E }

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
~~(A-B) --> 4~~
(D-C) --> 5
(C-E) --> 6
(D-E) --> 9

ORDER

(B - C)
(A - D)
(A - C)

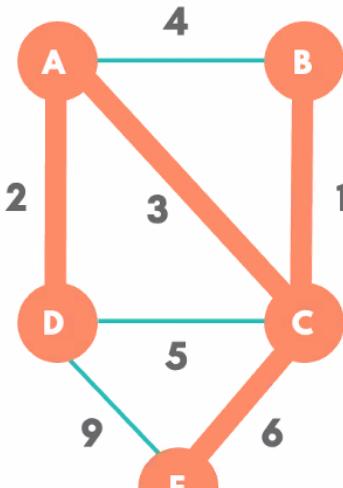
2 PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Calin Ovidiu-Raul	
Dobai Lorena	Alexandra Cretu
Dobai Lorena	
Denis	Laurentiu Taran
Denis	
Boldea Patricia-Maria	Andrei Tatov
Boldea Patricia-Maria	
Urs David Krin	Sebi Cozma
Urs David Krin	
El Kharoubi Iosif	Vivienne Dumitrescu
El Kharoubi Iosif	



KRUSKAL'S ALGORITHM

STEPS



MST

{ A , D , B , C , E }

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
~~(A-B)~~ --> 4
~~(D-C)~~ --> 5
(C-E) --> 6
(D-E) --> 9

ORDER

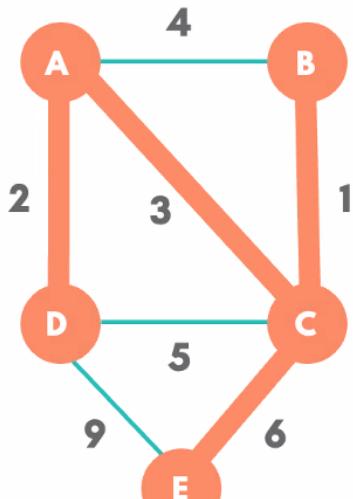
(B - C)
(A - D)
(A - C)
(C - E)

2 PICK THE SMALLEST EDGE. CHECK IF IT FORMS A CYCLE WITH THE SPANNING TREE FORMED SO FAR. IF CYCLE IS NOT FORMED, INCLUDE THIS EDGE. ELSE, DISCARD IT.

DabuRaul
Lăcramioara #
Calin Ovidiu-Raul
Dobai Lorena
Denis
Boldea Patricia-Maria
Urs David Krin
Sebi Cozma
El Kharoubi Iosif
Vivienne Dumitrescu

KRUSKAL'S ALGORITHM

STEPS



MST

{ A , D , B , C , E }

(B-C) --> 1
(A-D) --> 2
(A-C) --> 3
~~(A-B) --> 4~~
~~(D-C) --> 5~~
(C-E) --> 6
(D-E) --> 9

ORDER

(B - C)
(A - D)
(A - C)
(C - E)

3 REPEAT STEP#2 UNTIL THERE ARE (V-1) EDGES IN THE SPANNING TREE.

	DabuRaul	Lăcramioara #
	DabuRaul	
	Calin Ovidiu-Raul	Ashley Hurrelbrink
	Calin Ovidiu-Raul	
	Dobai Lorena	Alexandra Cretu
	Dobai Lorena	
	Denis	Laurentiu Taran
	Denis	
	Boldea Patricia-Maria	Andrei Tatov
	Boldea Patricia-Maria	
	Urs David Krin	Sebi Cozma
	Urs David Krin	
	El Kharoubi Iosif	Vivienne Dumitrescu
	El Kharoubi Iosif	



Unmute

Start Video

Participants

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options



2 KRUSKAL'S ALGORITHM IMPLEMENTATION



Unmute
Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

DabuRaul	
DabuRaul	
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	
Dobai Lorena	Alexandra Cretu
Denis	
Denis	Laurentiu Taran
Boldea Patricia-Maria	
El Kharoubi Iosif	
El Kharoubi Iosif	Sebi Cozma
Vivienne Dumitrescu	
Vivienne Dumitrescu	Vivienne Dumitrescu



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

KRUSKAL'S ALGORITHM

IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

```
MST = {}

//create tree and add it to the forest MST
For each v in V
    MAKE_SET(v)

SORT(e) //sort all the edges into increasing order by weight

//find edge with lowest cost and unite the trees
For all edges(u, v) in increasing order
    if ( FIND_SET(u) != FIND_SET(v) )
        UNION(u, v)
        MST = MST U {(u,v)}
```

The screenshot shows a video conference interface with a list of participants on the right. Each participant has a name, a small profile picture, and a video feed. A yellow hand icon is next to the name of the person currently speaking. The participants listed are:

- DabuRaul
- Lăcramioara #
- DabuRaul
- Lăcramioara #
- Calin Ovidiu-Raul
- Calin Ovidiu-Raul
- Ashley Hurrelbrink
- Dobai Lorena
- Dobai Lorena
- Alexandra Cretu
- Denis
- Denis
- Laurentiu Taran
- Boldea Patricia-Maria
- Andrei Tatov
- Urs David Krin
- Sebi Cozma
- Sebi Cozma
- El Kharoubi Iosif
- Vivienne Dumit...
- El Kharoubi Iosif
- Vivienne Dumitrescu



Unmute



Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options



KRUSKAL'S ALGORITHM IMPLEMENTATION

Given $G = (V, E)$. Output a MST.

MAKE_SET()
FIND_SET()
UNION()

LINKED LIST

Each set is a singly linked list

AS A FOREST

Forests of up-trees are used



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave





Recording

You are viewing Ashley Hurrelbrink's screen

View Options



KRUSKAL'S ALGORITHM

LINKED LIST

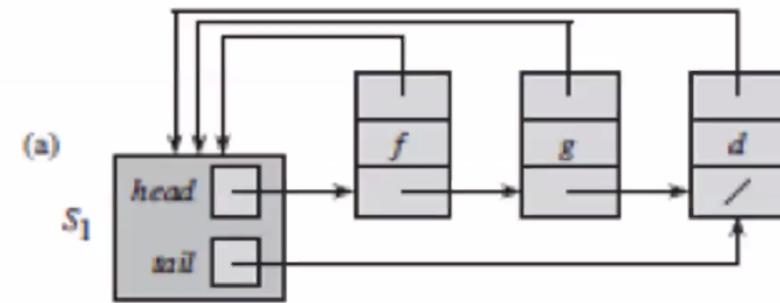
Given $G = (V, E)$. Output a MST.

SET ATTRIBUTES

- Head
- Tail

OBJECT ATTRIBUTES

- the set member
- pointer to the set object
- next



DabuRaul

Lăcramioara #

DabuRaul

Lăcramioara #

Calin Ovidiu-Raul

Calin Ovidiu-Raul

Ashley Hurrelbrink

Dobai Lorena

Dobai Lorena

Alexandra Cretu

Denis

Denis

Laurentiu Taran

Boldea Patricia-Maria

Andrei Tatov

Sebi Cozma

Sebi Cozma

El Kharoubi Iosif

Vivienne Dumitrescu

El Kharoubi Iosif

Vivienne Dumitrescu



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

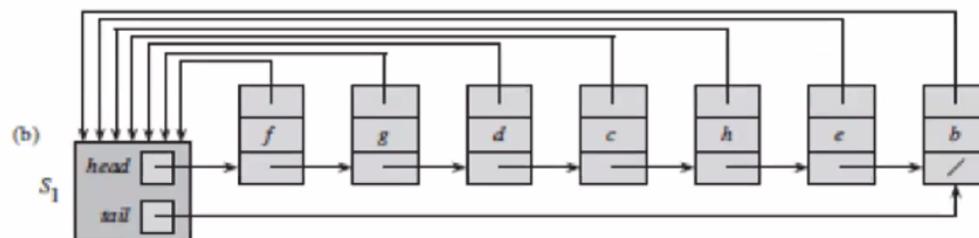
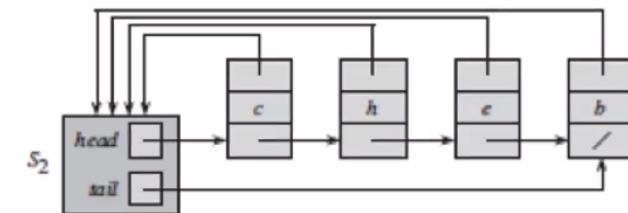
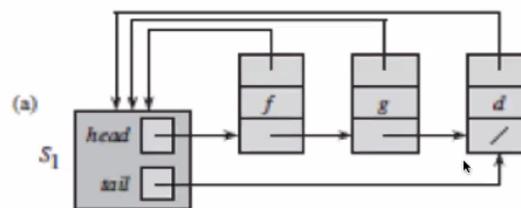
Apps

Leave

KRUSKAL'S ALGORITHM

LINKED LIST

Given $G = (V, E)$. Output a MST.



DabuRaul

Lăcramioara #

DabuRaul

Lăcramioara #

Calin Ovidiu-Raul

Calin Ovidiu-Raul

Ashley Hurrelbrink

Dobai Lorena

Dobai Lorena

Alexandra Cretu

Denis

Denis

Laurentiu Taran

Boldea Patricia-Maria

Andrei Tatov

Sebi Cozma

Sebi Cozma

El Kharoubi Iosif

Vivienne Dumitrescu

El Kharoubi Iosif

Vivienne Dumitrescu



KRUSKAL'S ALGORITHM

LINKED LIST

Given $G = (V, E)$. Output a MST.

MAKE_SET(X)

Create a new **linked list** whose only object is x .

FIND_SET(X)

Follow the pointer from x back to its set object and then return the member in the object that **head points to**.

UNION(X, Y)

Append y 's list onto the end of x 's list.

The representative of x 's list becomes the new **set representative**.

Use the **tail pointer** for x 's list to quickly find where to append y 's list.



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

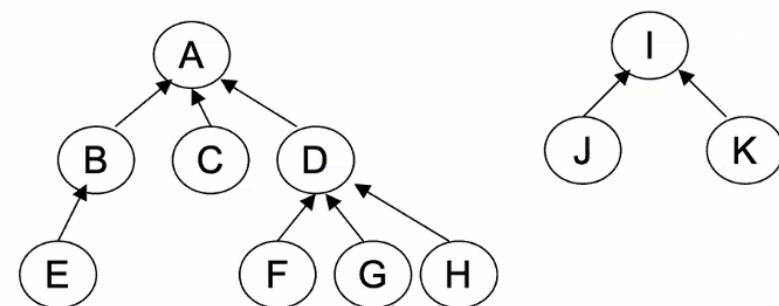
KRUSKAL'S ALGORITHM AS A FOREST

Given $G = (V, E)$. Output a MST.

IMPLEMENTED AS A FOREST OF UP-TREES

UP-TREE

- Set of nodes
- Every node has a parent except for the root
- The root is the set representative



DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Calin Ovidiu-Raul	
Dobai Lorena	Alexandra Cretu
Dobai Lorena	
Denis	Laurentiu Taran
Denis	Laurentiu Taran
Boldea Patricia-Maria	Andrei Tatov
Boldea Patricia-Maria	
Urs David Krin	Sebi Cozma
Urs David Krin	Sebi Cozma
El Kharoubi Iosif	Vivienne Dumit...
El Kharoubi Iosif	Vivienne Dumitrescu



Unmute

Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View

KRUSKAL'S ALGORITHM AS A FOREST

Given $G = (V, E)$. Output a MST.

MAKE_SET(X)

Initialize node as root

FIND_SET(X)

walk upwards in the tree, starting from x , following parent links, until arriving at the root.

UNION(X, Y)

One tree becomes a subtree of the other



Unmute
Start Video

Participants
20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Calin Ovidiu-Raul	Alexandra Cretu
Dobai Lorena	Denis
Dobai Lorena	Laurentiu Taran
Denis	Boldea Patricia-Maria
Denis	Andrei Tatov
Urs David Krin	Sebi Cozma
Urs David Krin	Vivienne Dumitrescu
El Kharoubi Iosif	El Kharoubi Iosif
El Kharoubi Iosif	Vivienne Dumitrescu



Leave



Recording

You are viewing Ashley Hurrelbrink's screen

View Options

View



APPLICATION PRIM & KRUSKAL ALGORITHM



Unmute

Start Video

Participants 20

Chat

Share Screen

Record

Show Captions

Reactions

Apps

Leave

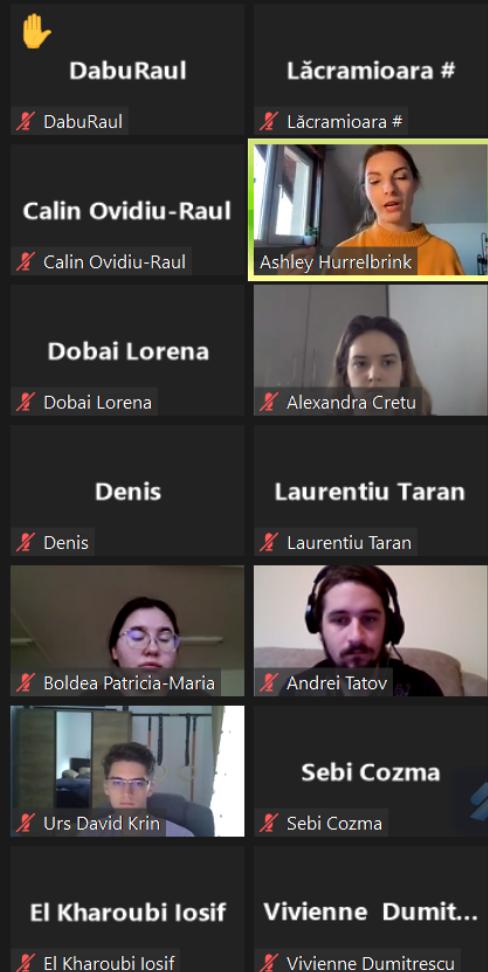
DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	
Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	
Dobai Lorena	Alexandra Cretu
Denis	
Denis	Laurentiu Taran
Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	Sebi Cozma
El Kharoubi Iosif	
El Kharoubi Iosif	Vivienne Dumitrescu

▼

OVERVIEW

PRIM VS KRUSKAL

Prim's Algorithm	Kruskal's Algorithm
The tree that we are making or growing always remains connected.	The tree that we are making or growing usually remains disconnected.
Prim's Algorithm grows a solution from a random vertex by adding the next cheapest vertex to the existing tree.	Kruskal's Algorithm grows a solution from the cheapest edge by adding the next cheapest edge to the existing tree / forest.
Prim's Algorithm is faster for dense graphs.	Kruskal's Algorithm is faster for sparse graphs.



Unmute



Start Video

20

^

Share Screen

^

Show Captions

^

Apps

Leave

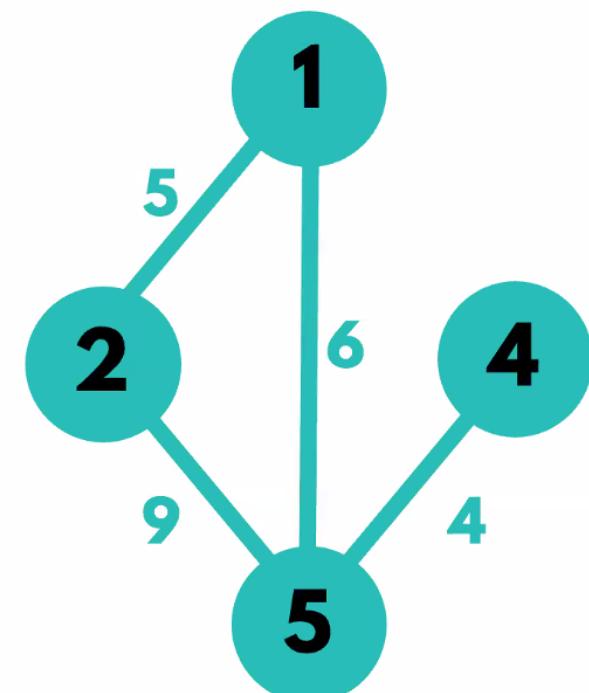
LAB 10 Application

PRIM & KRUSKAL

- 1** IMPLEMENT PRIM'S ALGORITHM FOR MST USING DISTANCE ARRAYS.

- 2** GIVEN THE WEIGHTED UNDIRECTED GRAPH IN THE FIGURE, ILLUSTRATE THE STEPS OF PRIM'S ALGORITHM. APPLY THE ALGORITHM ILLUSTRATING STEP BY STEP THE CONTENTS OF THE DATA STRUCTURES THAT ARE USED.

- 3** GIVEN THE WEIGHTED UNDIRECTED GRAPH, ILLUSTRATE THE STEPS OF KRUSKAL'S ALGORITHM. EXPLAIN EACH STEP OF THE ALGORITHM AND THE CONTENTS OF THE ADDITIONAL DATA STRUCTURES THAT ARE USED.



DabuRaul	Lăcramioara #
DabuRaul	Lăcramioara #
Calin Ovidiu-Raul	Ashley Hurrelbrink
Dobai Lorena	Alexandra Cretu
Denis	Laurentiu Taran
Boldea Patricia-Maria	Andrei Tatov
Urs David Krin	Sebi Cozma
El Kharoubi losif	Vivienne Dumitrescu