جامعة المنارة الخاصة
كلية الهندسة
هندسة الروبوت والأنظمة الذكية

مهارات الحاسوب

Computer Skills UNRC101

مدرس المقرر
أ.د. مثنى علي القبيلي

الفصل الدراسي الأول | الاثنين 14/11/2022 | العام الدراسي 2022-2023

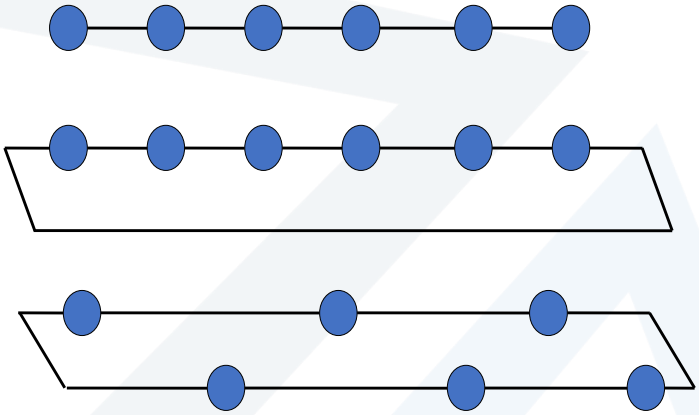
Chapter 3

Tree Construction

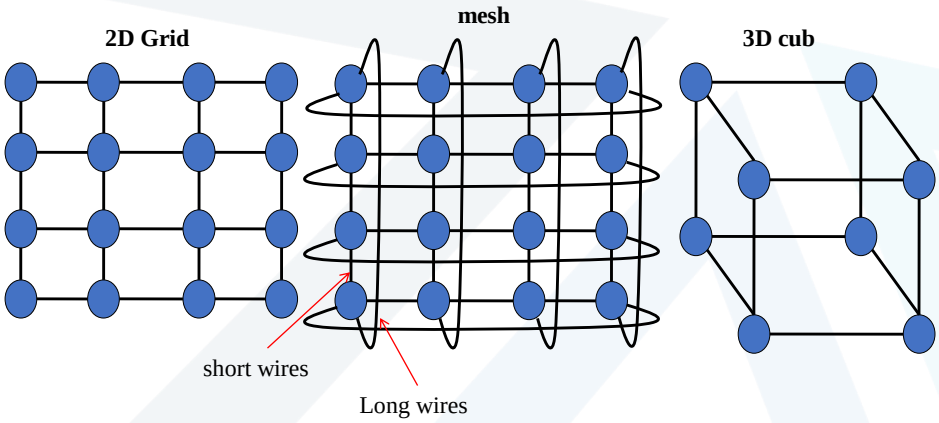
Interconnection Topologies

- Class networks scaling with N
- Logical Properties:
  - ✓ distance, degree
- Physical properties
  - ✓ length, width
- Fully connected network
  - ✓ diameter
  - ✓ outdegree
  - ✓ cost

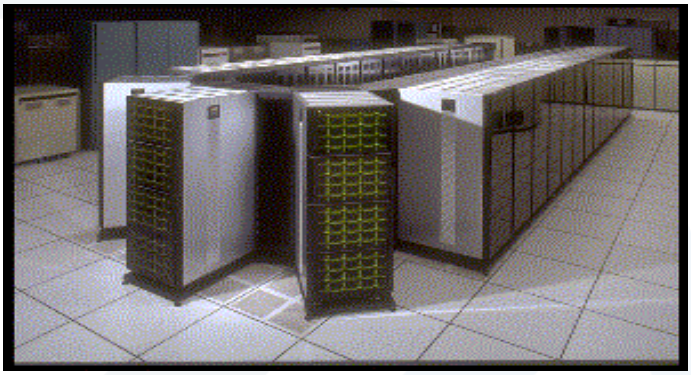
Linear Arrays and Rings



Multidimensional Meshes and Cub

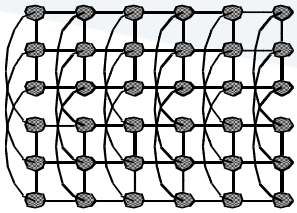


Real World 2D mesh

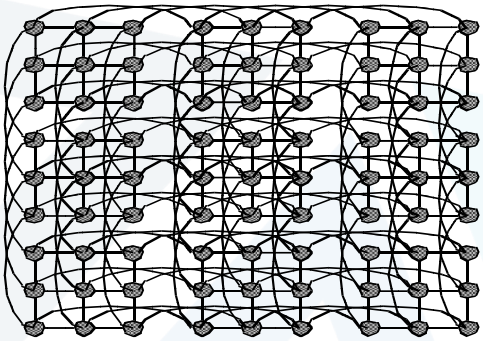


- 1824 node Paragon: 16 x 114 array

Embeddings in two dimensions



6 x 3 x 2



9x 3 x 3

- Embed multiple logical dimension in one physical dimension using long wires

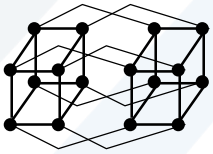
Hypercubes

- Also called binary n-cubes. # of nodes = N = 2^n
- O(logN) Hops
- Good bisection BW
- Complexity
  - ✓ Out degree is n = logN

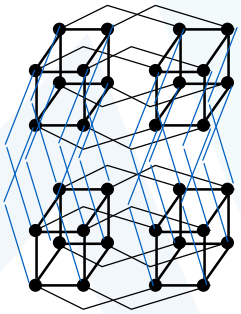
1-D

2-D

3-D

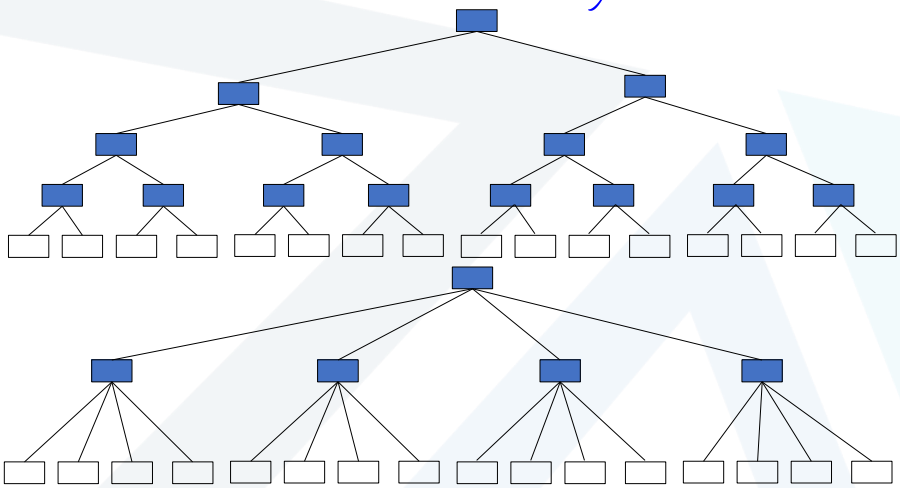


4-D

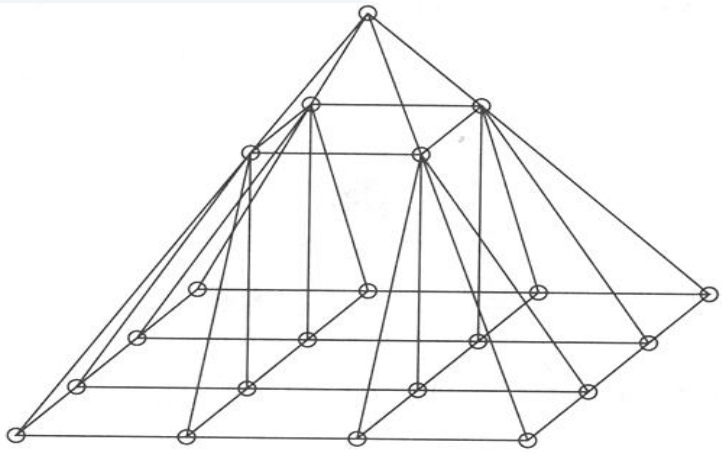


5-D !

Tree: 2-D Hierarchy

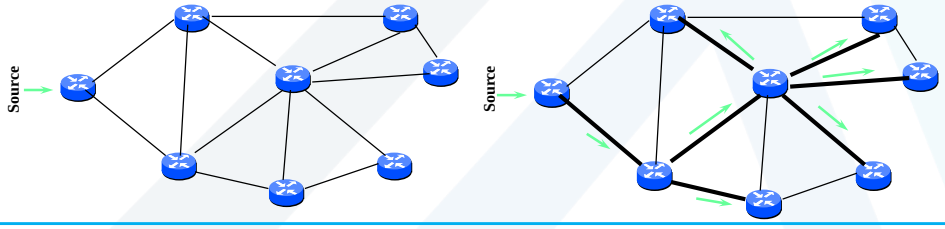


Pyramid: 3-D hierarchy

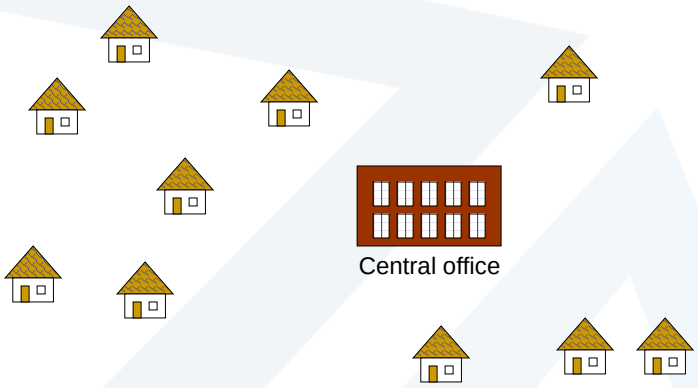


spanning tree الشجرة الممتدة

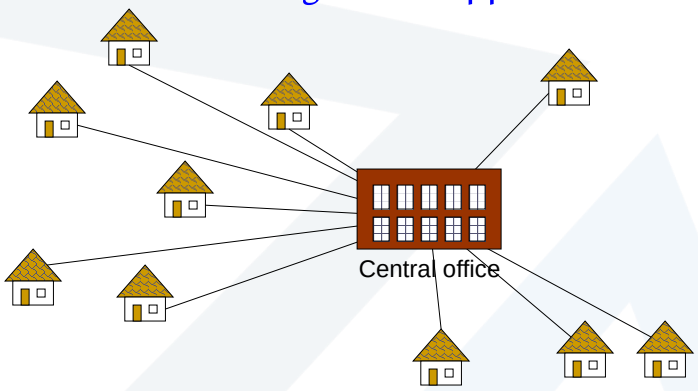
- يتم تحديد الشجرة المبنية بحيث يوجد طريق فعال واحد بين كل زوجين من الموجهات
- يبث الموجه رزم البيانات الداخلة على جزء واجهات الشجرة الممتدة فقط باستثناء الخط الذي أتى منه الطرد، بحيث يعرف كل موجه أي من خطوطه تنتمي للشجرة الممتدة
  - تعتمد الشجرة الممتدة على التكاليف المتعلقة بالعقد والأقواس التي يمكن أن تستعمل لبناء الشبكة، لذا يتم العمل على إنشاء شبكة ذات كلفة بناء منخفضة
- يدعى هذا التصميم بالتصميم منخفض الكلفة minimum-cost design أو الشجرة الممتدة الأقصر



Problem: Laying Telephone Wire

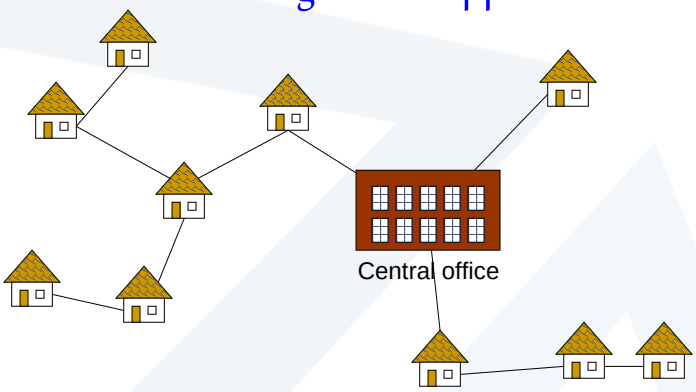


Wiring: Naïve Approach



Expensive!

Wiring: Better Approach



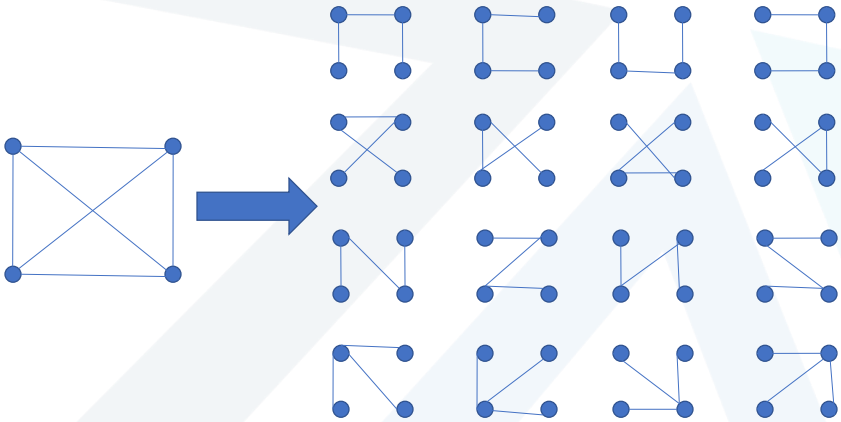
Minimize the total length of wire connecting the customers

Spanning Tree

- هي عبارة عن مخطط جزئي لمخطط غير موجه موزون G undirected weighted graph، مثل:
  - ✓ هو عبارة عن شجرة Tree
  - ✓ تغطي كل القمم/العقد v (حيث تحوي |v|-1 حافة)

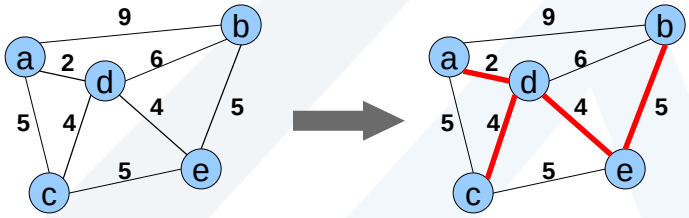
- يمكن أن يحتوي المخطط عدة أشجار ممتدة، مثلاً يحوي المخطط الكلي لأربع قمم/عقد 16 شجرة ممتدة

Spanning Tree



Minimum Spanning Tree (MST)

- بفرض أن حواف المخطط لها أوزان محددة أو أطوال/تأخير. عندها **ستكون كلفة الشجرة هي مجموع التأخيرات للوصول إلى كل العقد ويكون مقدار التأخير نهاية-إلى-نهاية الوسطي هو مجموع هذه التأخيرات مقسوماً على عدد الوصلات**
- لبناء الشجرة الممتدة الأقصر، يتم أخذ أدنى التأخيرات على وصلات المخطط للوصول إلى كل القمم حتى يتم بناء هذه الشجرة



Applications of Minimum-Cost Spanning Trees

- هنالك عدة تطبيقات تستخدم الأشجار الممتدة ذات الكلفة المنخفضة Minimum-cost spanning trees. ومنها:

  - ✓ بناء شبكات الكابلات والتي تضم n توضع وذلك اعتماداً على الكلفة الأدنى
  - ✓ بناء شبكات الطرق التي تضم n مدينة مع الكلفة الأدنى
  - ✓ الحصول على مجموعة مستقلة من معادلات الدارات من أجل شبكة كهربائية
  - ✓ بناء اتصالات الشبكات البعيدة والانترنت اعتماداً على الكلفة الأدنى

minimum spanning tree الشجرة الممتدة الأقصر

- تنبع فكرة الكلفة من ضعف الشبكة التي تعاني من وجود نقطة واحدة للفشل وكذلك من تحديد مقدار اللامركزية في الشبكة والذي يدل على أعظم درجة من العقد يمكن أن تقبلها كل عقدة في الشبكة (عدد الأبناء الأعظمي/الوصلات الأعظمية)
- تعتمد هذه الفكرة على تأخير نهاية-إلى-نهاية end-to-end delay، والهدف هو تخفيض هذا التأخير انطلاقاً من عقدة واحدة، تدعى المنبع، باتجاه العقد الأخرى في الشبكة، وهو ما يدعى بشجرة الطريق الأقصر من المنبع source Shortest Path Tree (SPT)
- يوجد عدة نماذج معروفة للشجرة الممتدة الأقصر:
  - ✓ Dijkstra's Algorithm
  - ✓ Kruskal's Algorithm
  - ✓ Prime's Algorithm
  - ✓ Hierarchical/Clusters method

Kruskal's Algorithm

- وهي خوارزمية من خوارزميات نظرية البيان Graph Theory والتي تجد الشجرة الممتدة الأقصر لمخطط متصل موزون connected weighted graph (حيث يكون لكل وصلة وزنها)
- هذا يعني بأنها ستجد مجموعة جزئية من الحواف/الأطراف والتي تشكل شجرة تشمل كل العقد، وبالتالي يتم تخفيض الوزن الكلي لكل حواف الشجرة
- إذا لم يكن المخطط متصلاً، يتم إيجاد الشجرة الممتدة الأقصر لكل عنصر متصل
- تعمل هذه الخوارزمية كما يلي:
  - ✓ يتم خلق الغابة (Forest) F (مجموعة من الأشجار)، حيث تكون كل قمة vertex في المخطط هي عبارة عن شجرة منفصلة
  - ✓ يتم خلق مجموعة S تحتوي كل الحواف في المخطط (بحيث لا تكون هذه المجموعة خالية)
  - ✓ يتم نزع كل حافة ذات أخفض وزن من المجموعة S
  - ✓ إذا اتصلت الحافة مع شجرتين مختلفتين، يتم ضمها إلى الغابة، حيث يتم توحيد الشجرتين في شجرة واحدة
  - ✓ أو يتم التخلص من الحافة

Kruskal's Algorithm

- في نهاية الخوارزمية، يكون للغابة مكون واحد فقط ويتم تشكيل الشجرة الممتدة الأقصر في المخطط
- باستعمال بنية معطيات ملائمة، يمكن أن يتم تنفيذ/إجراء هذه الخوارزمية O(m log m) مرة، حيث m هي عدد الحواف في المخطط

Kruskal's Algorithm

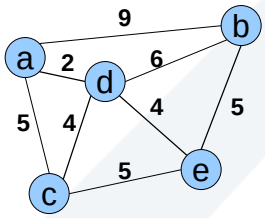
- Joseph Bernard Kruskal, Jr
- Kruskal Approach:
  - ✓ Select the minimum weight edge that does not form a cycle

Kruskal's Algorithm:

```
    sort the edges of G in increasing order by length
    keep a subgraph S of G, initially empty
    for each edge e in sorted order
        if the endpoints of e are disconnected in S
        add e to S
return S
```

Kruskal's Algorithm

- Create a forest of trees from the vertices
- Repeatedly merge trees by adding "**safe edges**" until only one tree remains
- A "safe edge" is an edge of minimum weight which does not create a cycle



forest: {a}, {b}, {c}, {d}, {e}

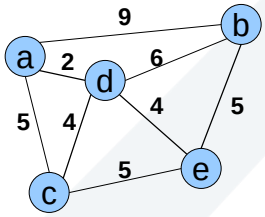
Kruskal's Algorithm

Initialization

a. Create a set for each vertex $v \in V$

b. Initialize the set of "safe edges" A comprising the MST to the empty set

c. Sort edges by increasing weight



F = {a}, {b}, {c}, {d}, {e}

$A = \varnothing$

E = {(a,d), (c,d), (d,e), (a,c), (b,e), (c,e), (b,d), (a,b)}

Kruskal's Algorithm

For each edge $(u,v) \in E$ in increasing order while more than one set remains:

If u and v, belong to different sets U and V

a. add edge (u,v) to the safe edge set

$A = A \cup \{(u,v)\}$

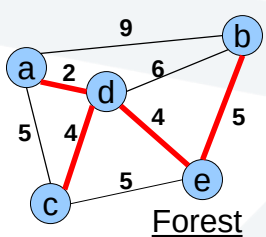
b. merge the sets U and V

$F = F - U - V + (U \cup V)$

Return A

- Running time bounded by sorting (or findMin)

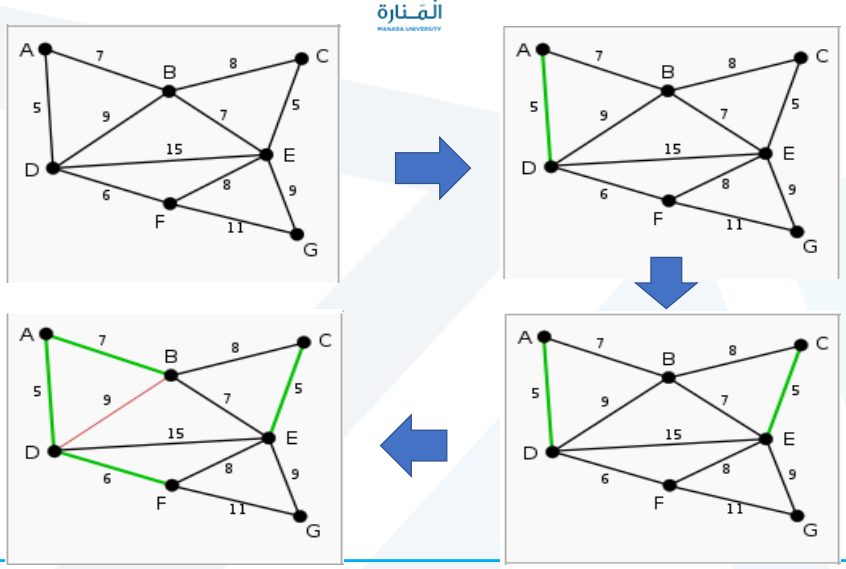
Kruskal's Algorithm



$E = \{(\cancel{a,d}), (\cancel{c,d}), (\cancel{d,e}), (a,c), (\cancel{b,e}), (c,e), (b,d), (a,b)\}$

Forest	A
{a}, {b}, {c}, {d}, {e}	$\varnothing$
{a,d}, {b}, {c}, {e}	{(a,d)}
{a,d,c}, {b}, {e}	{(a,d), (c,d)}
{a,d,c,e}, {b}	{(a,d), (c,d), (d,e)}
{a,d,c,e,b}	{(a,d), (c,d), (d,e), (b,e)}

Kruskal's Algorithm – Example 2



Kruskal's Algorithm - Example

Prime's Algorithm

Prime's Algorithm

- وهي أيضاً خوارزمية لإيجاد الشجرة الممتدة الأقصر لبيان متصل موزون
- في حال لم يكن البيان متصلاً، سنحاول فقط إيجاد الشجرة الممتدة الأقصر لعنصر متصل واحد من هذه العناصر المتصلة
- تعمل هذه الخوارزمية كما يلي:
  - ✓ يتم خلق شجرة تحوي قمة vertex واحدة، يتم اختيارها بشكل شجري من المخطط
  - ✓ يتم خلق مجموعة تحوي كل الحواف في المخطط
  - ✓ إجراء حلقة loop حتى تتصل كل حافة بقيمتين في الشجرة (قيمة باتجاه عقدة ما من الشجرة)
  - ✓ يتم الحذف/النزع للحافة ذات الوزن الأدنى والتي تصل قمة من الشجرة مع قمة غير متصلة من الشجرة
  - ✓ يتم إضافة هذه الحافة إلى الشجرة
- يمكن أن يتم تنفيذ/إجراء هذه الخوارزمية $O(m+n \log m)$ مرة، حيث m هي عدد الحواف في المخطط و n هي عدد القمم
- يتم في هذه الخوارزمية الحصول على SPT مثل خوارزمية Kruskal

Prime's Algorithm

- Robert Clay Prim
- Prim Approach:
  - ✓ Choose an arbitrary start node v
  - ✓ At any point in time, we have connected component N containing v and other nodes V-N
  - ✓ Choose the minimum weight edge from N to V-N

Prim's Algorithm:

```
let T be a single vertex x
while (T has fewer than n vertices)
{
   find the smallest edge connecting T to G-T
   add it to T
}
```


Prime's Algorithm

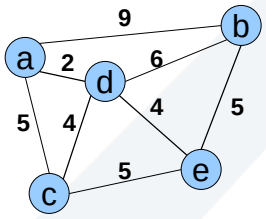
Initialization

a. Pick a vertex ***r*** to be the root

b. Set $D(r) = 0$, $parent(r) = null$

c. For all vertices $v \in V$, $v \neq r$, set $D(v) = \infty$

d. Insert all vertices into priority queue ***P***, using distances as the keys



e	a	b	c	d
0	∞	∞	∞	∞

Vertex	Parent
e	-

Prime's Algorithm

While *P* is not empty:

1. Select the next vertex ***u*** to add to the tree

   u = ***P.deleteMin***()

2. Update the weight of each vertex ***w*** adjacent to ***u*** which is **not** in the tree (i.e., ***w*** ∈ ***P***)

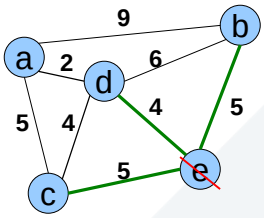
   If ***weight***(***u,w***) < ***D***(***w***),

   a. ***parent***(***w***) = ***u***

   b. ***D***(***w***) = ***weight***(***u,w***)

   c. Update the priority queue to reflect new distance for ***w***

Prime's Algorithm



e	d	b	c	a
0	∞	∞	∞	∞

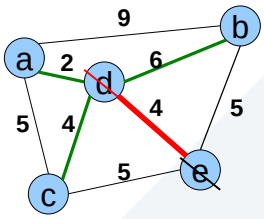
Vertex	Parent
e	-
b	-
c	-
d	-

d	b	c	a
4	**5**	**5**	∞

Vertex	Parent
e	-
b	e
c	e
d	e

The MST initially consists of the vertex ***e***, and we update the distances and parent for its adjacent vertices

Prime's Algorithm



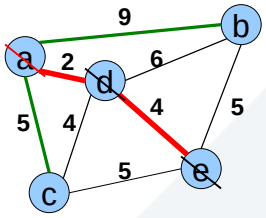
d	b	c	a
4	**5**	**5**	∞

Vertex	Parent
e	-
b	e
c	e
d	e

a	c	b
2	**4**	**5**

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

Prime's Algorithm



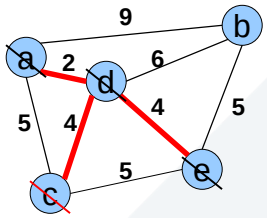
a	c	b
2	**4**	**5**

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

c	b
4	**5**

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

Prime's Algorithm



c	b
4	**5**

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

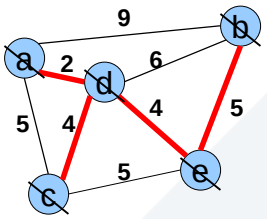
b
5

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

Prime's Algorithm

b
5

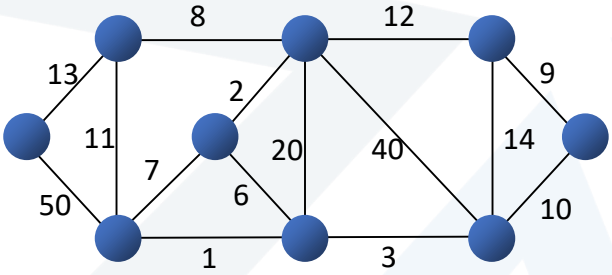
Vertex	Parent
e	-
b	e
c	d
d	e
a	d



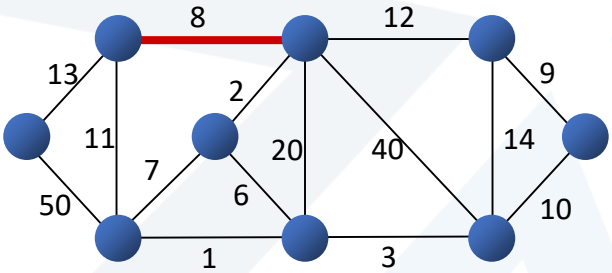
The final minimum spanning tree

Vertex	Parent
e	-
b	e
c	d
d	e
a	d

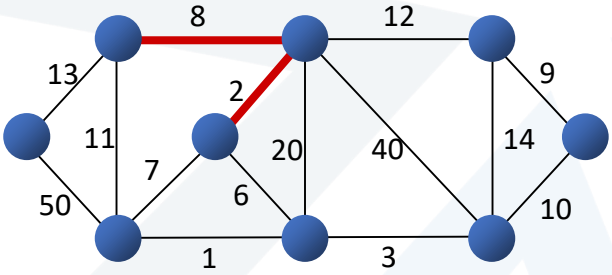
Prim's Algorithm – Example 2



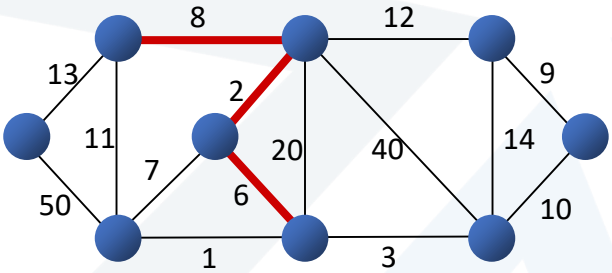
Prim's Algorithm – Example 2



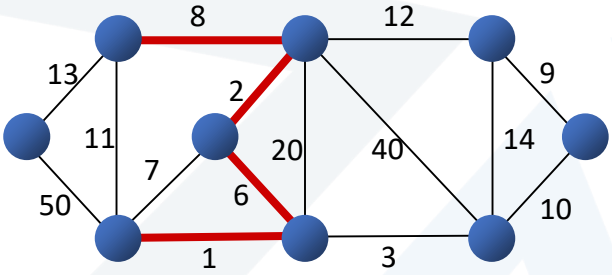
Prim's Algorithm – Example 2



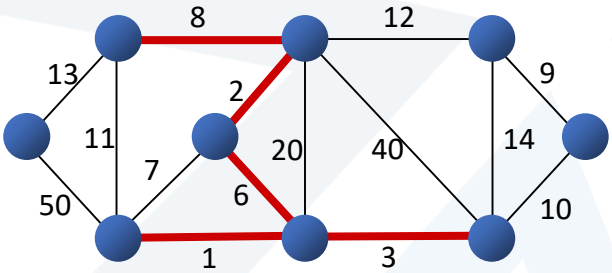
Prim's Algorithm – Example 2



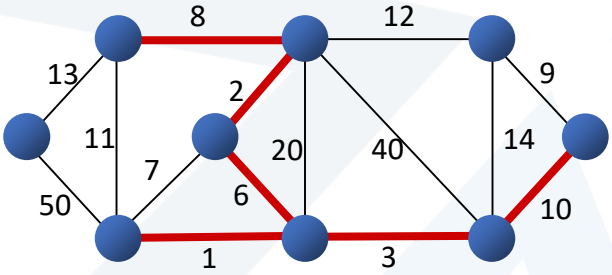
Prim's Algorithm – Example 2



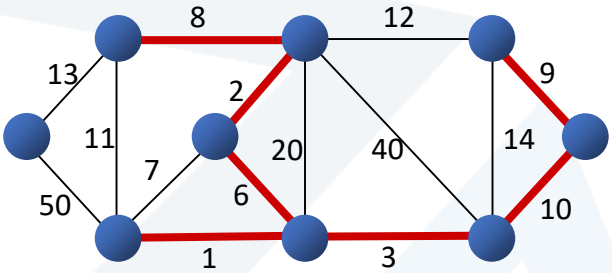
Prim's Algorithm – Example 2



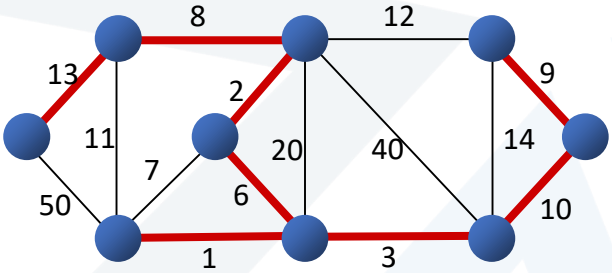
Prim's Algorithm – Example 2



Prim's Algorithm – Example 2



Prim's Algorithm – Example 2



مثال

Nodes	1	2	3	4	5	6	7	8	9	10
1	-	20	15	10	10	20	35	20	30	20
2	20	-	40	20	15	15	30	10	15	50
3	15	40	-	20	10	10	15	12	17	30
4	10	20	20	-	16	30	10	25	30	15
5	10	15	10	16	-	30	20	12	30	10
6	20	15	10	30	30	-	20	30	15	30
7	35	30	15	10	20	20	-	12	11	20
8	20	10	12	25	12	30	12	-	18	30
9	30	15	17	30	30	15	11	18	-	30
10	20	50	30	15	10	30	20	30	30	-

RTT table

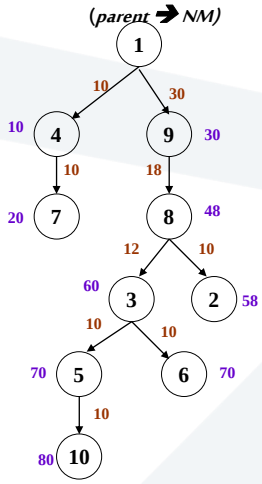
Join/ Leave events

Max-fanout=2

مثال

Kruskal's Algorithm: best end-to-end delay

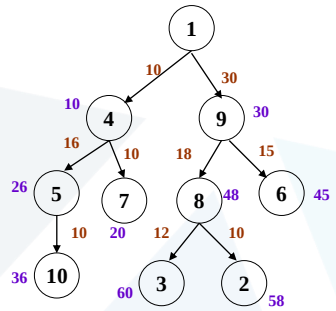
(parent → NM)



Tree cost=446 sec
mean end-to-end delay =49.56 sec
Max-diameter=80sec

Kruskal's Algorithm: best end-to-end delay

(parent → NM) + max Diameter<=60

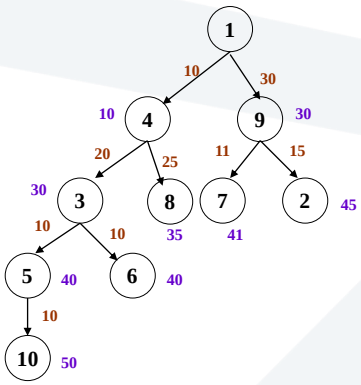


Tree cost=333 sec
mean end-to-end delay =37 sec
Max-diameter=60sec

مثال

Prim's Algorithm: best end-to-end delay

(RP→ *parent* → *NM)*



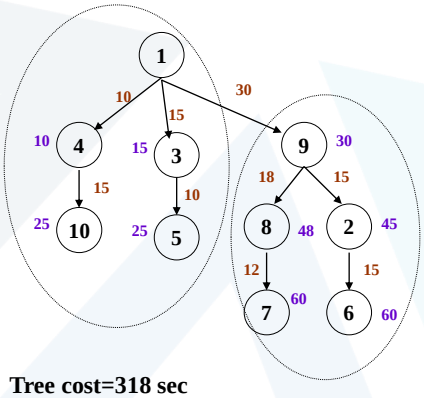
Tree cost=321 sec
mean end-to-end delay =35.67 sec
Max-diameter=50sec

Algorithm3: Clusters (Prime in each cluster)

In our example: 2 clusters

1 and 9 are leaders in each cluster

1 is principal leader



Tree cost=318 sec
mean end-to-end delay =35.33 sec
Max-diameter=60sec

Thanks