Минимално покриващо дърво

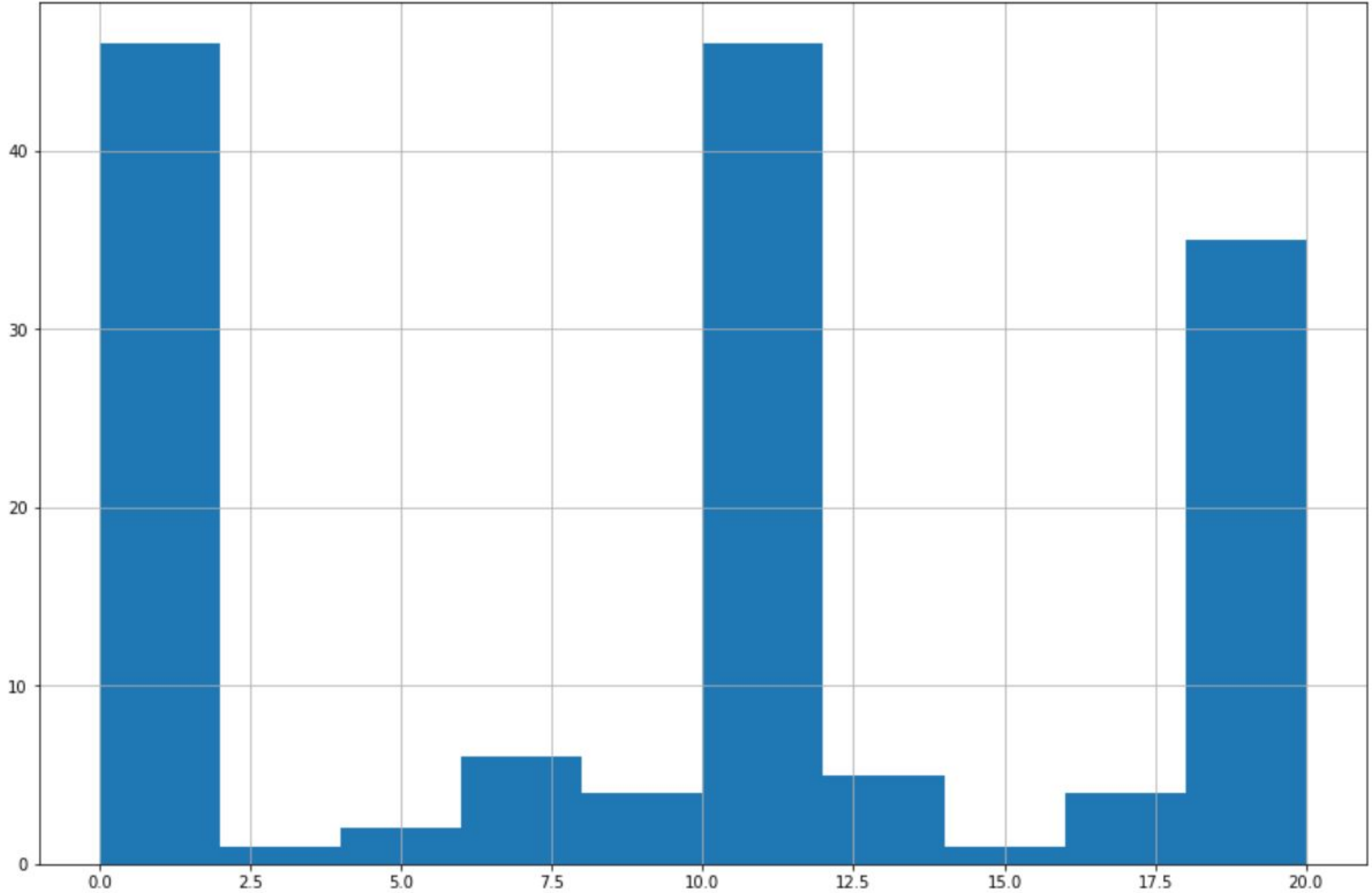
Лекция 12 по СДА, Софтуерно Инженерство
Зимен семестър 2018-2019г
д-р Милен Чечев

План за днес

- Организационни въпроси
  - Резултати от контролно 5
  - Обобщени резултати
  - Изпит / Оценяване
- Трета лекция за графи
  - Проблема за минимално покриващо дърво и мотивация
  - Алгоритъм на Прим
  - Алгоритъм на Крускал

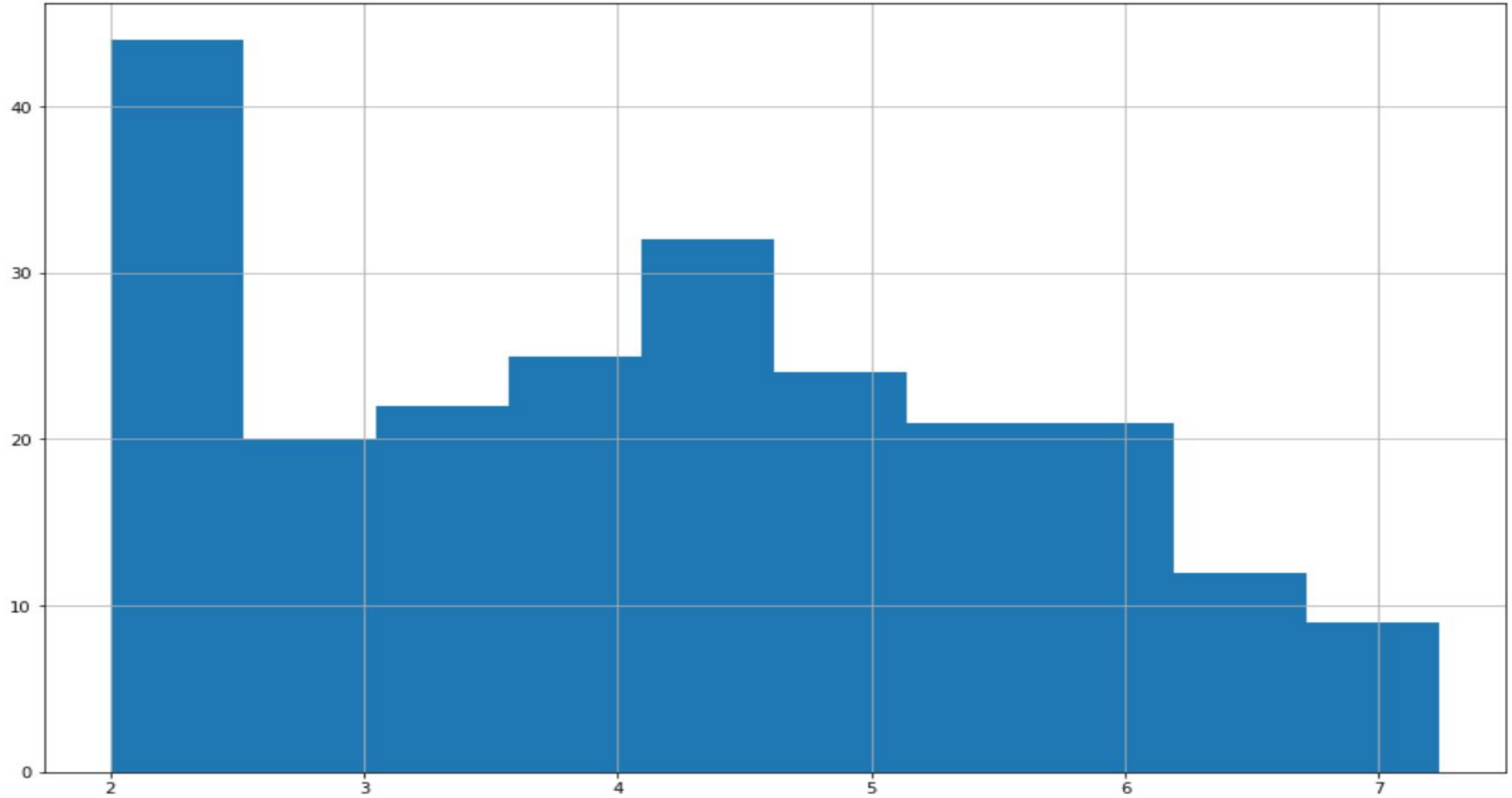
Контролно 5

- 20 контролни са анулирани заради неспазване на правилата за присъствие в залата!
- Средна оценка - 9.4/20



0
10
20
30
40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Текущи резултати контролно 1-5



Обобщение - оценки до момента

- 187 студента със текуща оценка от контролни >= 2.5
- 47 студента с текуща оценка от контролни >= 5.5 (53 точки или повече)

Текущи оценки от домашни:

- Голям процент от домашните са подобни, което налага за съмнителните домашни да се проведе защита след която да бъдат въведени точките от тях
- Точният формат на защитата ще бъде уточнен следващата седмица.

Какво следва?

- 10.01.2018 - Контролно 6 - минимално покриващо дърво и търсене на най-кратък път в граф
- 17.01.2018 - Контролно 7 - преговорни задачи
- 27.01.2018 - изпит
  - Тест с предимно затворени въпроси с множество верни отговори - 75 мин
  - Задачи - 75 мин

Домашно 11 със срок до 12.01

Домашно 12 със срок до 19.01

Минимално покриващо дърво на граф

MINIMUM SPANNING TREE

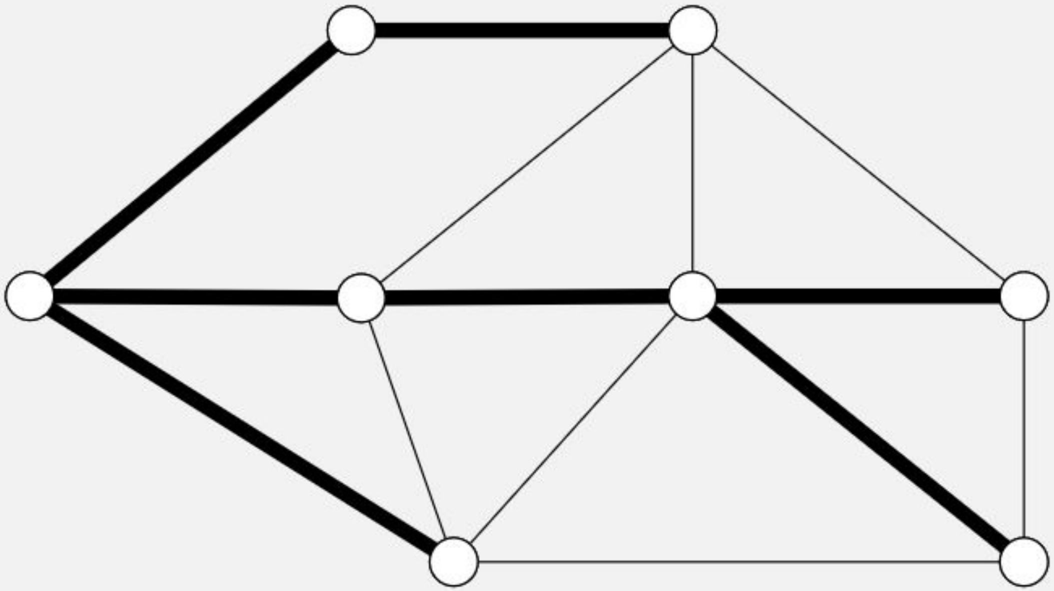
Проблем:

Даден граф да се трансформира към дърво(да няма цикли в графа) оставяйки в графа такива ребра, че сумата им да е минимална.

*разглеждаме проблема за ненасочен граф с тегла по ребрата

Def. A spanning tree of G is a subgraph T that is:

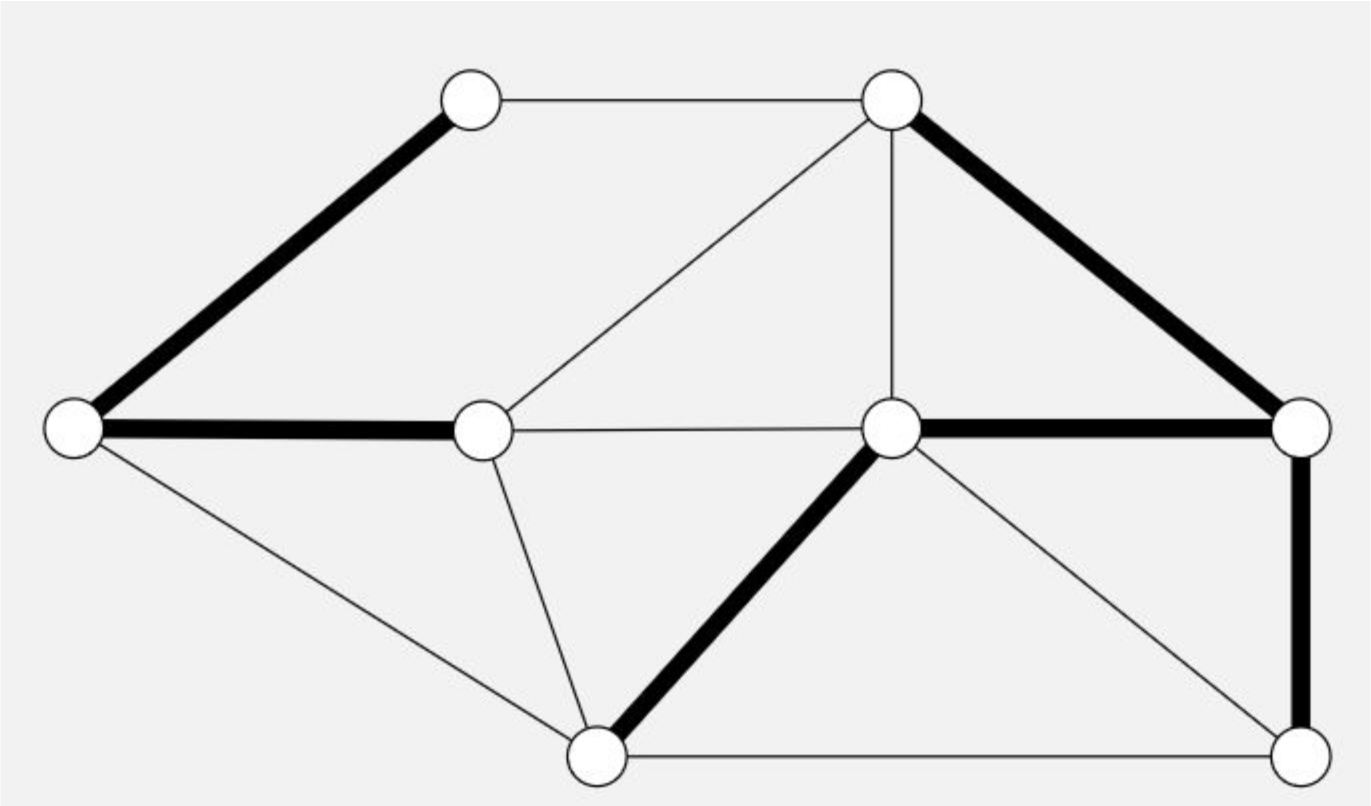
- A tree: connected and acyclic.
- Spanning: includes all of the vertices.



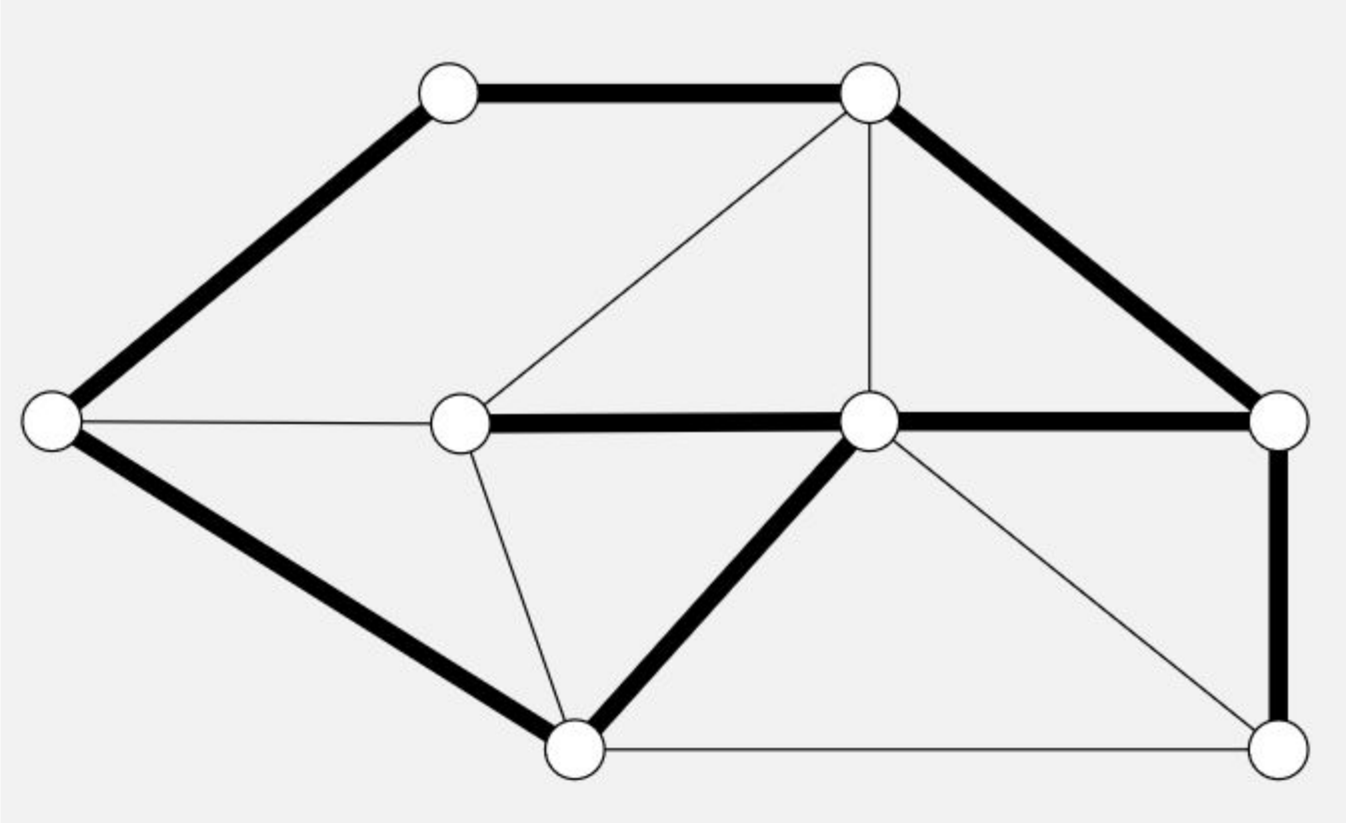
graph G

spanning tree T

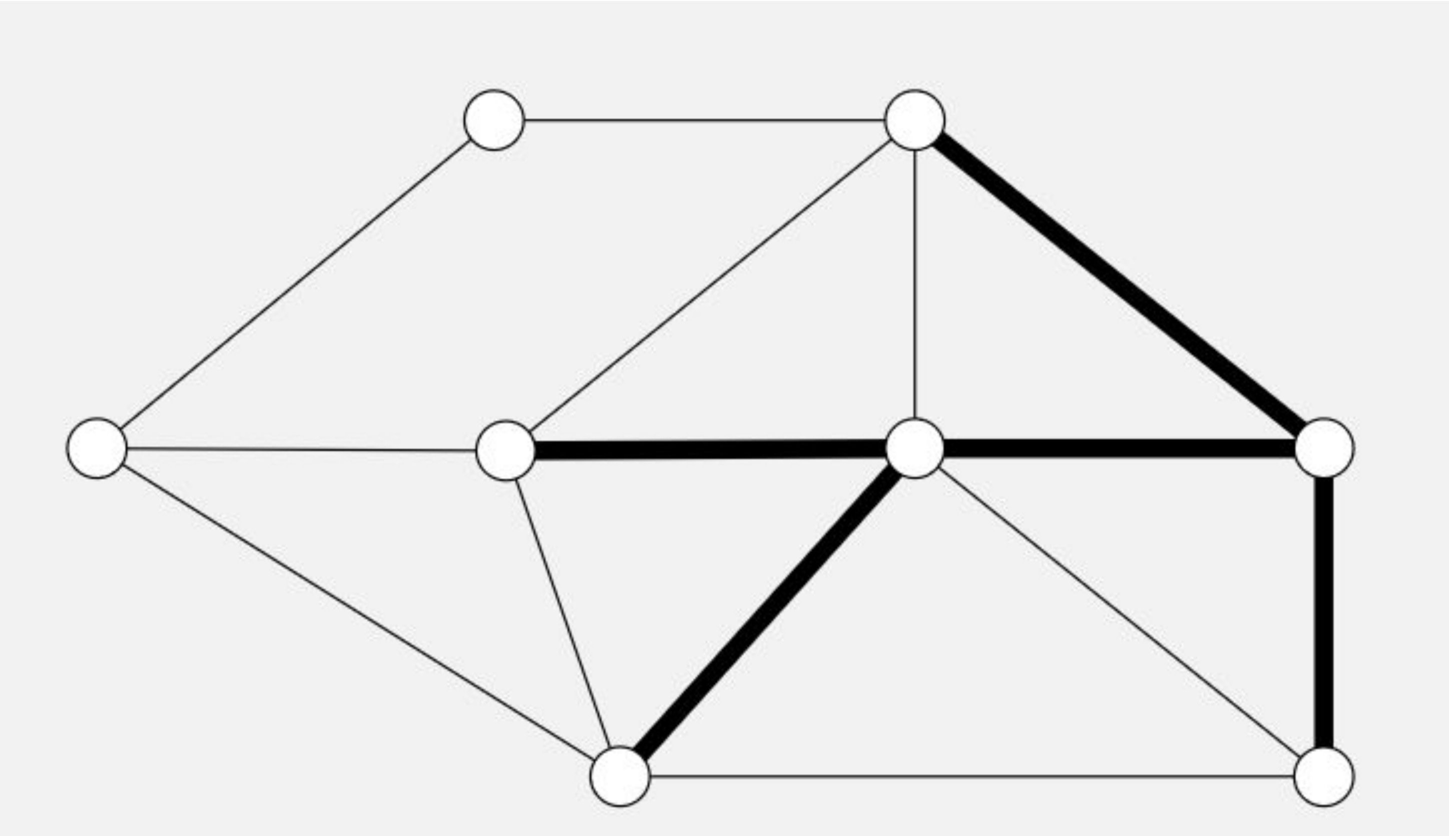
Това покриващо дърво ли е ?



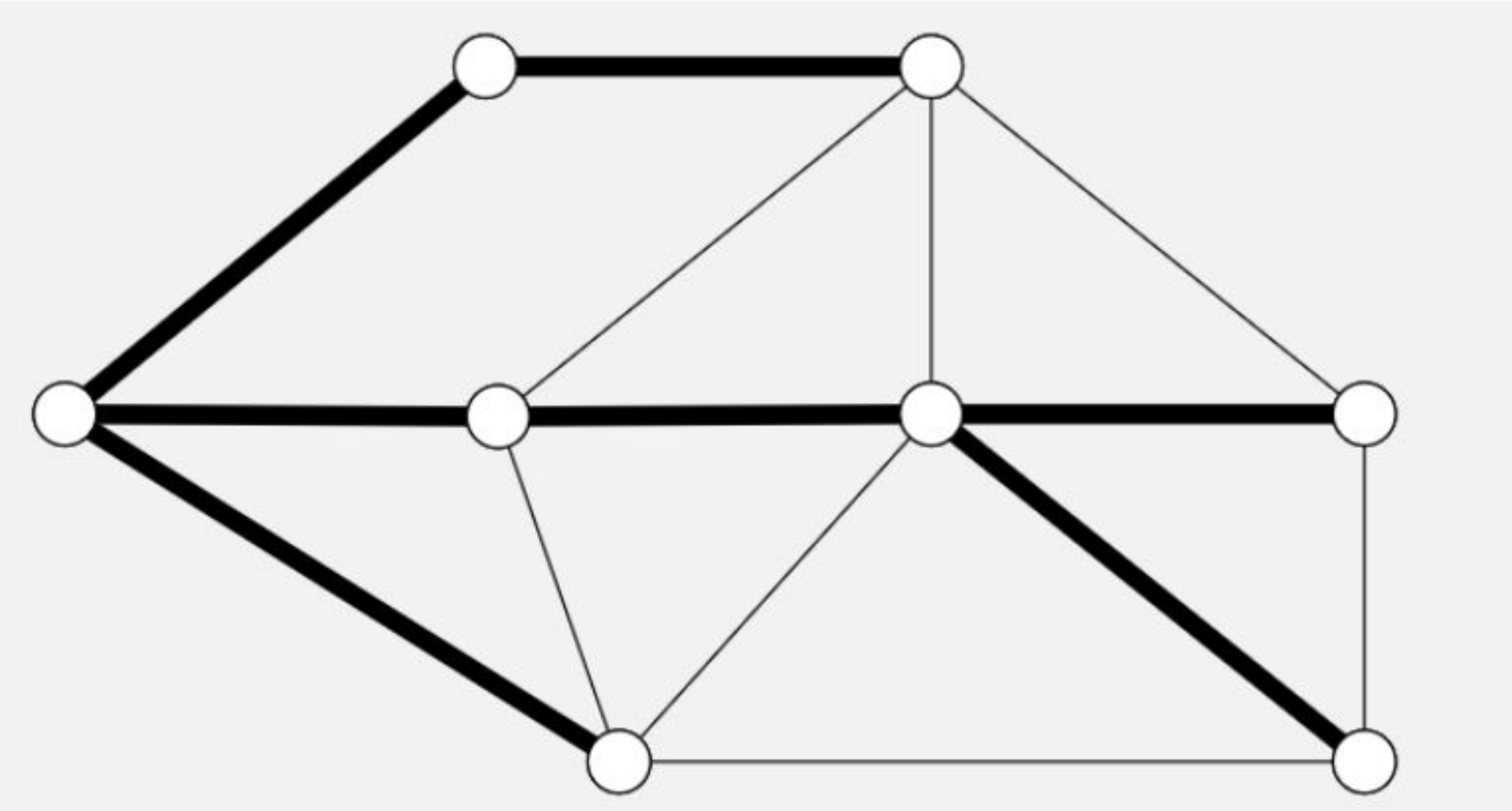
Това покриващо дърво ли е ?



Това покриващо дърво ли е ?



Това покриващо дърво ли е ?



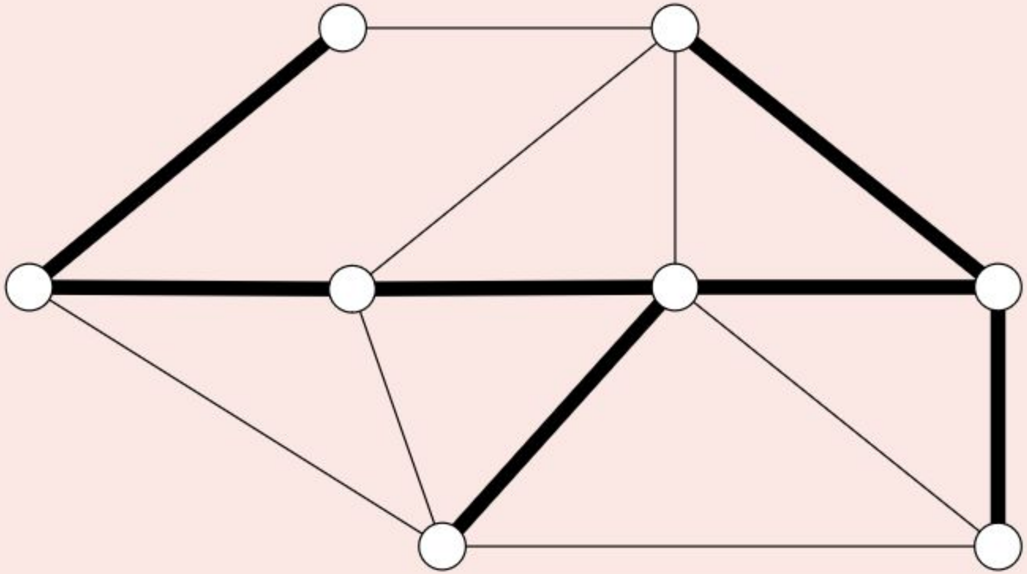
Let T be a spanning tree of a connected graph G with V vertices. Which of the following statements are true?

A. T contains exactly $V-1$ edges.

B. Removing any edge from T disconnects it.

C. Adding any edge to T creates a cycle.

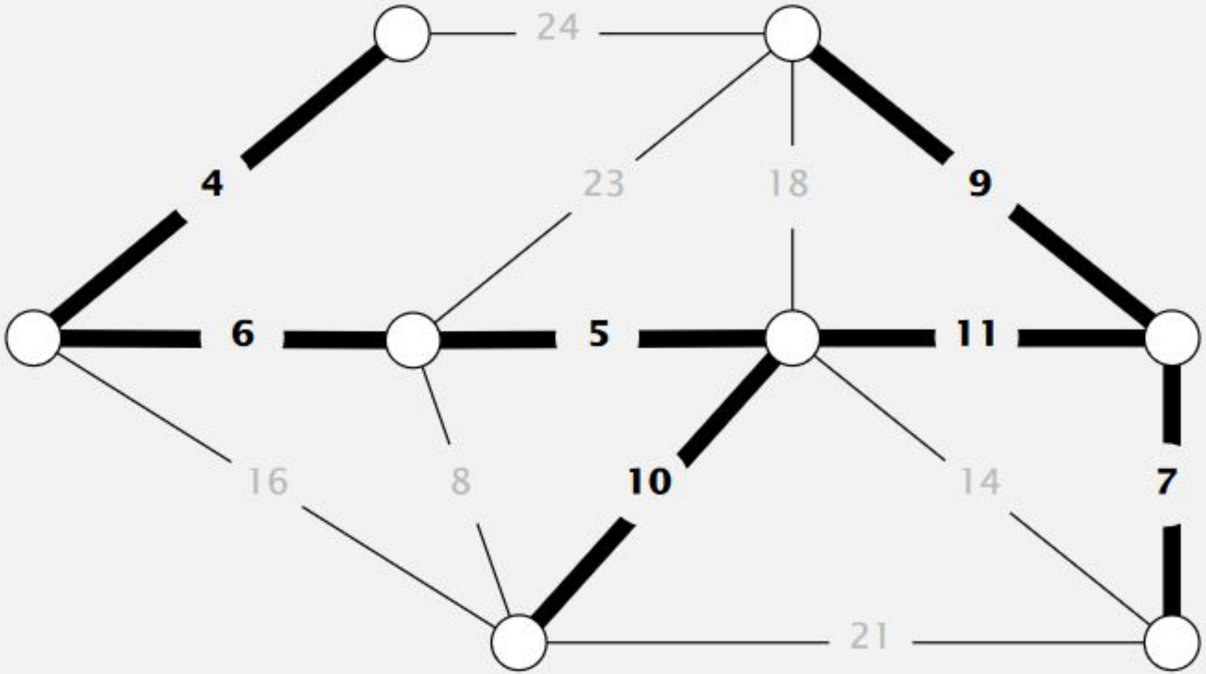
D. All of the above.



spanning tree T of graph G

Input. Connected, undirected graph G with positive edge weights.

Output. A spanning tree of minimum weight.

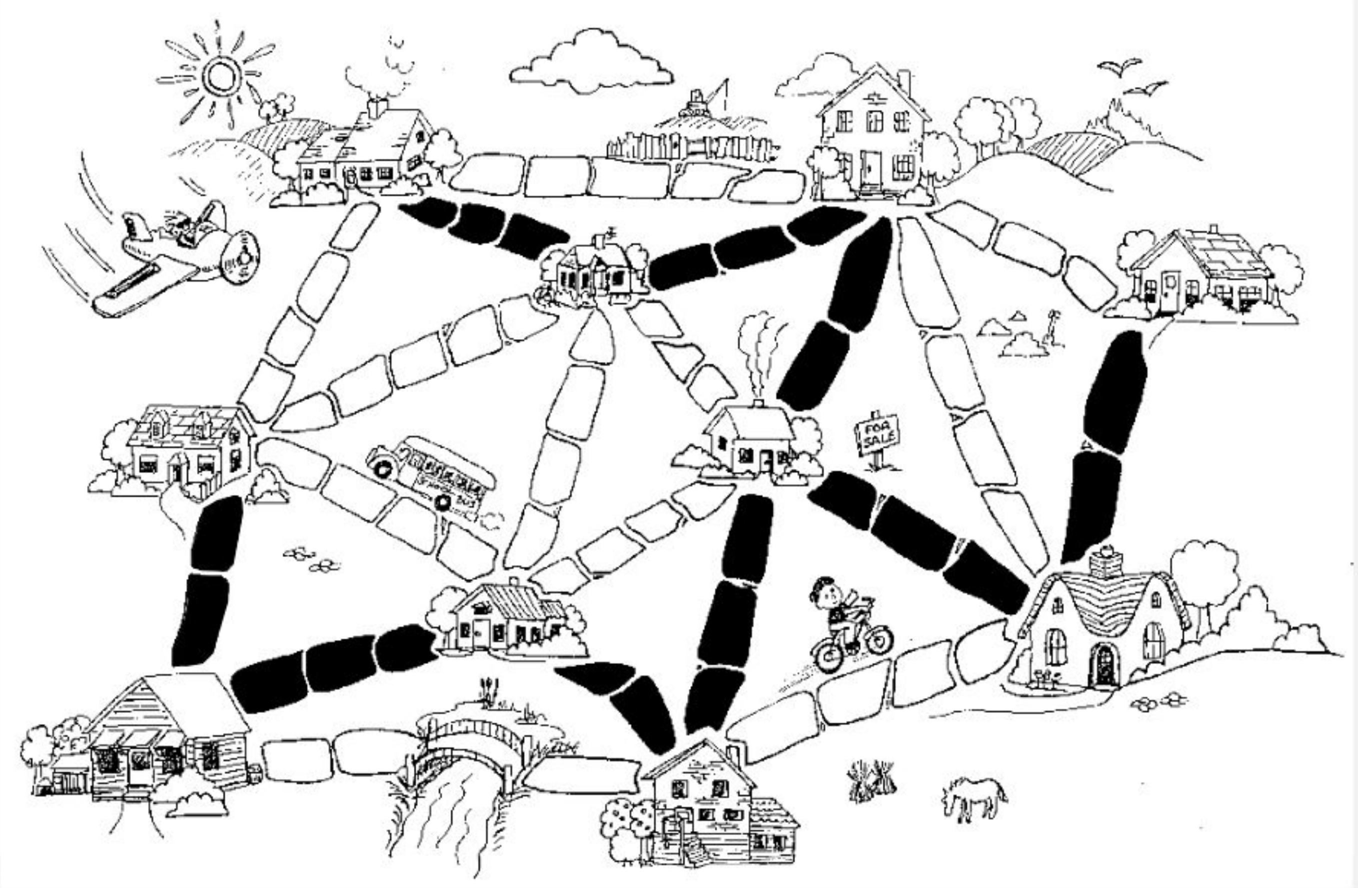


minimum spanning tree T

(weight = 52 = 4 + 6 + 10 + 5 + 11 + 9 + 7)

Brute force. Try all spanning trees?

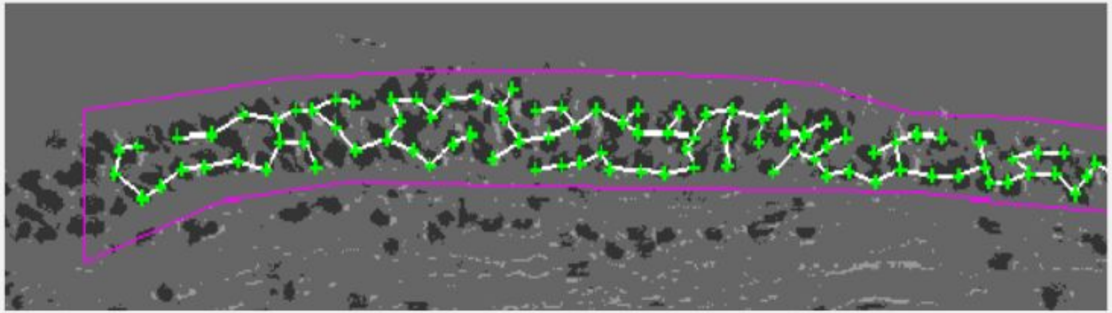
Network design

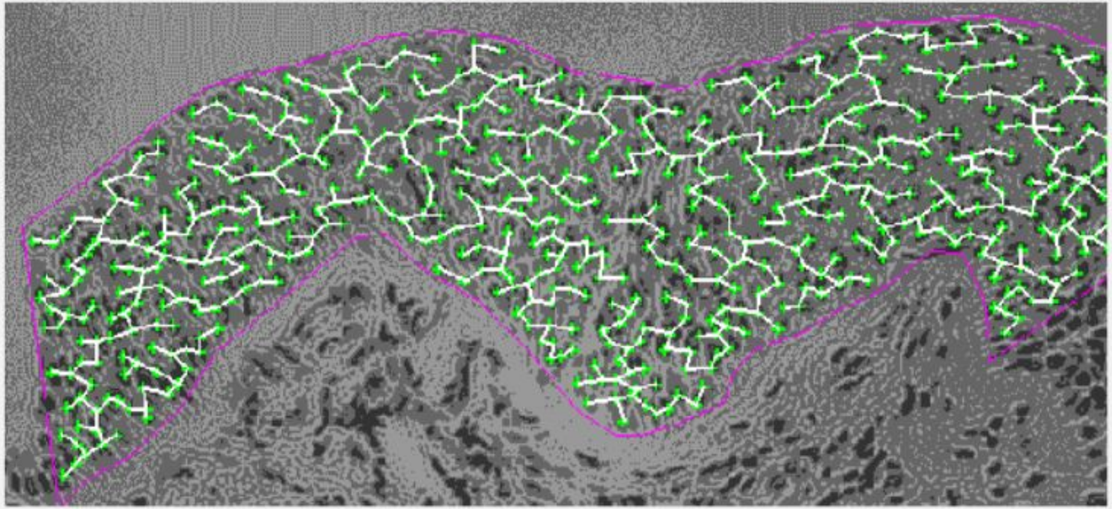


http://www.utdallas.edu/~besp/teaching/mst-applications.pdf

Medical image processing

MST describes arrangement of nuclei in the epithelium for cancer research





http://www.bccrc.ca/ci/ta01_archlevel.html

Applications

MST is fundamental problem with diverse applications.

- Cluster analysis.
- Real-time face verification.
- LDPC codes for error correction.
- Image registration with Renyi entropy.
- Curvilinear feature extraction in computer vision.
- Find road networks in satellite and aerial imagery.
- Handwriting recognition of mathematical expressions.
- Measuring homogeneity of two-dimensional materials.
  Model locality of particle interactions in turbulent fluid flows.
- Reducing data storage in sequencing amino acids in a protein.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Network design (communication, electrical, hydraulic, computer, road).
- Approximation algorithms for **NP**-hard problems (e.g., TSP, Steiner tree).

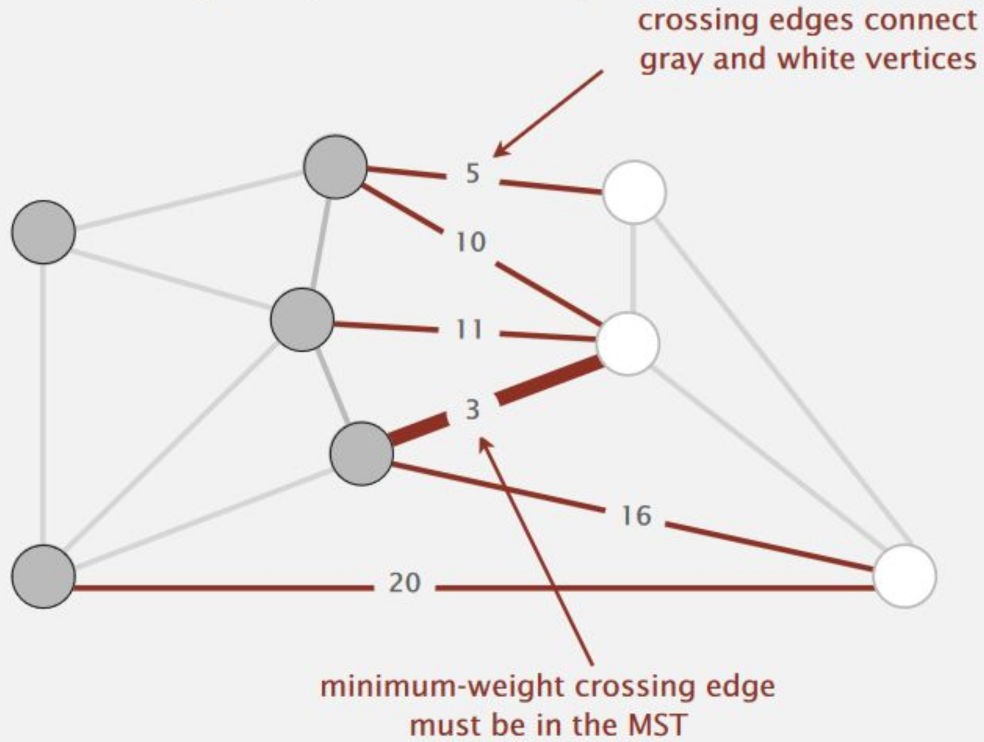
http://www.ics.uci.edu/~eppstein/gina/mst.html
http://www.utdallas.edu/~besp/teaching/mst-applications.pdf

Cut property

Def. A cut in a graph is a partition of its vertices into two (nonempty) sets.

Def. A crossing edge connects a vertex in one set with a vertex in the other.

Cut property. Given any cut, the crossing edge of min weight is in the MST.



Cut property: correctness proof

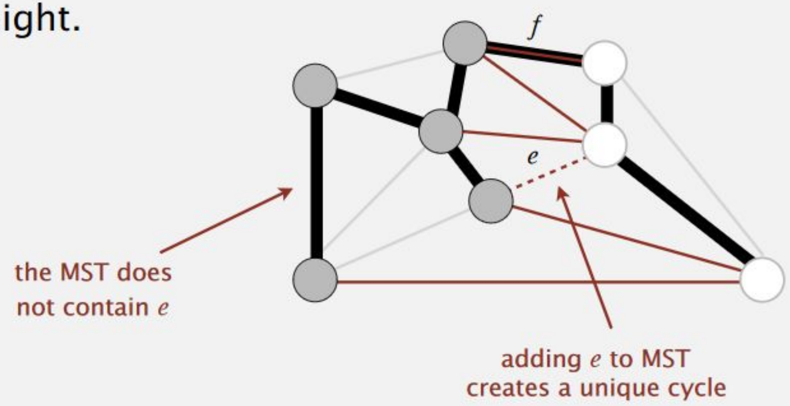
Def. A cut is a partition of a graph's vertices into two (nonempty) sets.

Def. A crossing edge connects two vertices in different sets.

Cut property. Given any cut, the min-weight crossing edge e is in the MST.

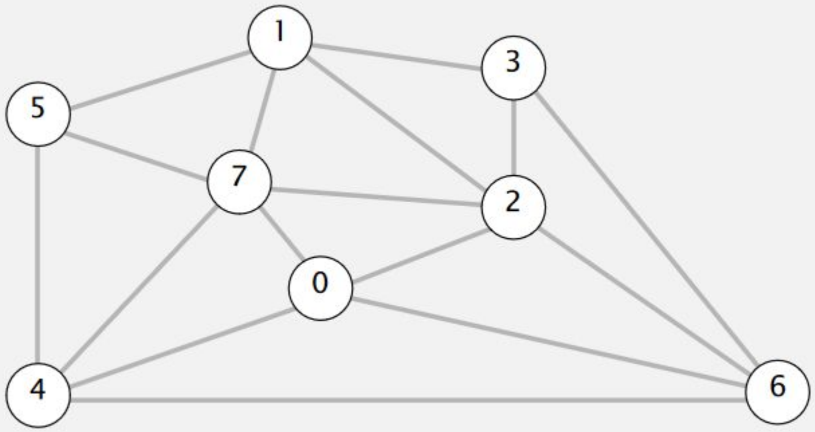
Pf. [by contradiction] Suppose e is not in the MST.

- Adding e to the MST creates a cycle.
- Some other edge f in cycle must be a crossing edge.
- Removing f and adding e is also a spanning tree.
- Since weight of e is less than the weight of f, that spanning tree has lower weight.
- Contradiction. ▪



Greedy MST algorithm

- Start with all edges colored gray.
- Find cut with no black crossing edges; color its min-weight edge black.
- Repeat until $V - 1$ edges are colored black.



an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

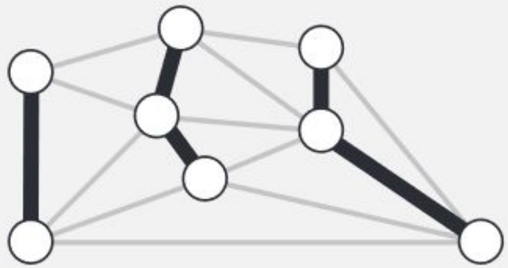
Greedy MST algorithm: correctness proof

Proposition. The greedy algorithm computes the MST.

Pf.

- Any edge colored black is in the MST (via cut property).
- Fewer than $V-1$ black edges $\Rightarrow$ cut with no black crossing edges.
  (consider cut whose vertices are any one connected component)

a cut with no black crossing edges



fewer than V-1 edges colored black

Алгоритъм на Крускал

- Сортираме всички ребра в нарастващ ред.
- Докато покриващото дърво няма V-1 ребра
  - Избираме следващо по-големина ребро от графа
  - Ако то не създава цикъл със вече избраните ребра за покриващото дърво, включваме реброто като част от покриващото дърво

Алгоритъм на Крускал - демо

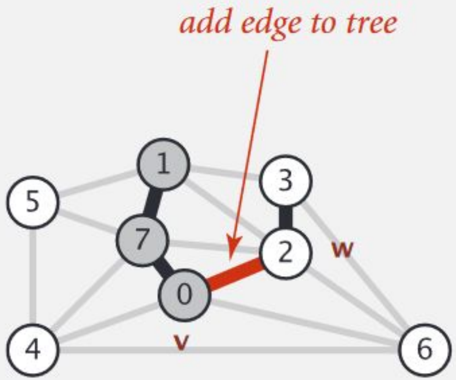
http://www.cs.princeton.edu/courses/archive/fall18/cos226/demos/43DemoKruskal.pdf

Kruskal's algorithm: correctness proof

Proposition. [Kruskal 1956] Kruskal's algorithm computes the MST.

Pf. [Case 1] Kruskal's algorithm adds edge $e = v\text{–}w$ to T.

- Vertices v and w are in different connected components of T.
- Cut = set of vertices connected to v in T.
- By construction of cut, no edge crossing cut is in T.
- No edge crossing cut has lower weight. Why?
- Cut property $\Rightarrow$ edge e is in the MST.

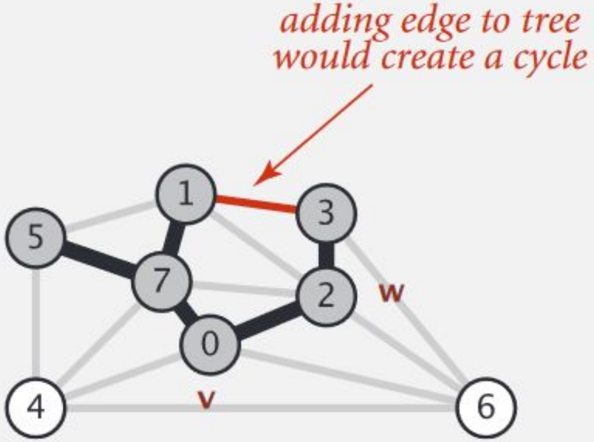


Kruskal's algorithm: correctness proof

Proposition. [Kruskal 1956] Kruskal's algorithm computes the MST.

Pf. [Case 2] Kruskal's algorithm discards edge $e = v\text{–}w$.

- From Case 1, all edges in T are in the MST.
- The MST can't contain a cycle. ▪

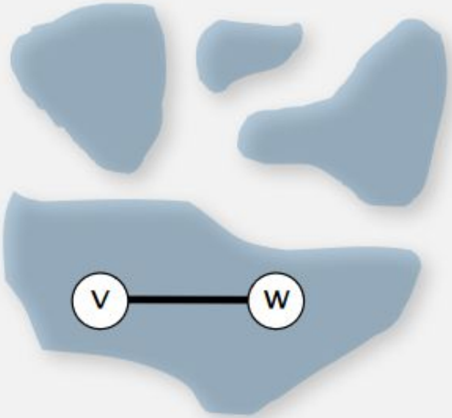


Kruskal's algorithm: implementation challenge

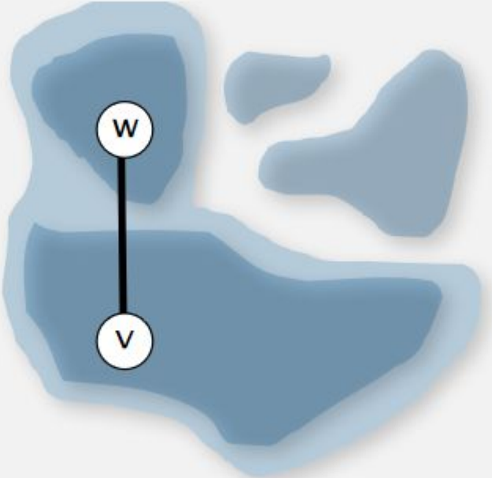
Challenge. Would adding edge v–w to tree T create a cycle? If not, add it.

Efficient solution. Use the union–find data structure.

- Maintain a set for each connected component in T.
- If v and w are in same set, then adding v–w would create a cycle.
- To add v–w to T, merge sets containing v and w.



Case 2: adding v–w creates a cycle



Case 1: add v–w to T and merge sets containing v and w

Kruskal's algorithm: running time

Proposition. Kruskal's algorithm computes MST in time proportional to $E \log V$ (in the worst case).

Pf.

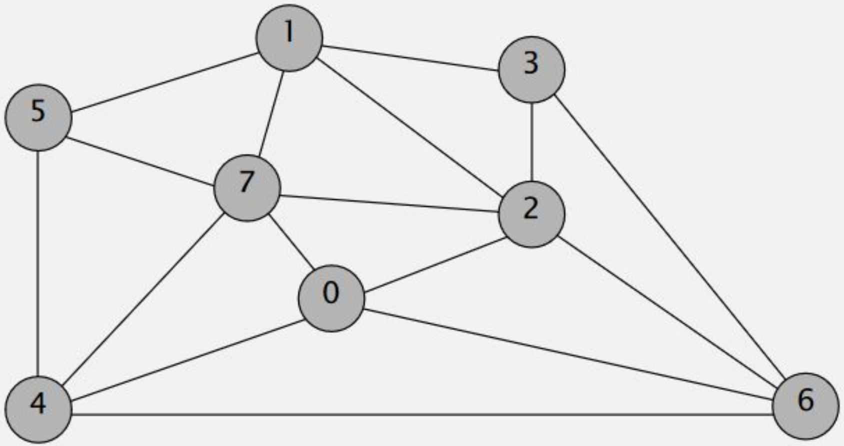
operation	frequency	time per op
SORT	1	$E \log E$
UNION	$V - 1$	$\log V$ †
FIND	$2\,E$	$\log V$ †

same as $E \log V$ if no parallel edges

† using weighted quick union

Prim's algorithm

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V-1$ edges.



an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Алгоритъм на Прим - демо

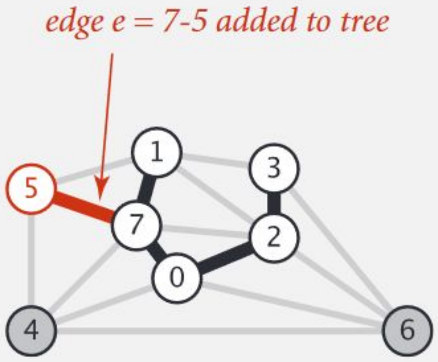
http://www.cs.princeton.edu/courses/archive/fall18/cos226/demos/43DemoPrim.pdf

Prim's algorithm: proof of correctness

Proposition. [Jarník 1930, Dijkstra 1957, Prim 1959]
Prim's algorithm computes the MST.

Pf. Let e = min weight edge with exactly one endpoint in T.

- Cut = set of vertices in T.
- No crossing edge is in T.
- No crossing edge has lower weight.
- Cut property $\Rightarrow$ edge e is in the MST. ▪



Minimum spanning trees:

Challenge. Find the min weight edge with exactly one endpoint in T.

How difficult to implement?

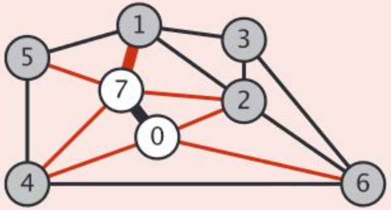
A. 1

B. $\log E$

C. V

D. E

1-7 is min weight edge with exactly one endpoint in T




```
1-7 0.19
0-2 0.26
5-7 0.28
2-7 0.34
4-7 0.37
0-4 0.38
6-0 0.58
```

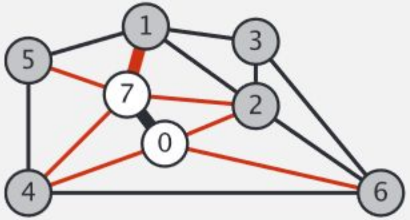
Prim's algorithm: lazy implementation

Challenge. Find the min weight edge with exactly one endpoint in T.

Lazy solution. Maintain a PQ of edges with (at least) one endpoint in T.

- Key = edge; priority = weight of edge.
- DELETE-MIN to determine next edge $e = v–w$ to add to T.
- If both endpoints v and w are marked (both in T), disregard.
- Otherwise, let w be the unmarked vertex (not in T):
  - add e to T and mark w
  - add to PQ any edge incident to w (assuming other endpoint not in T)

1-7 is min weight edge with exactly one endpoint in T

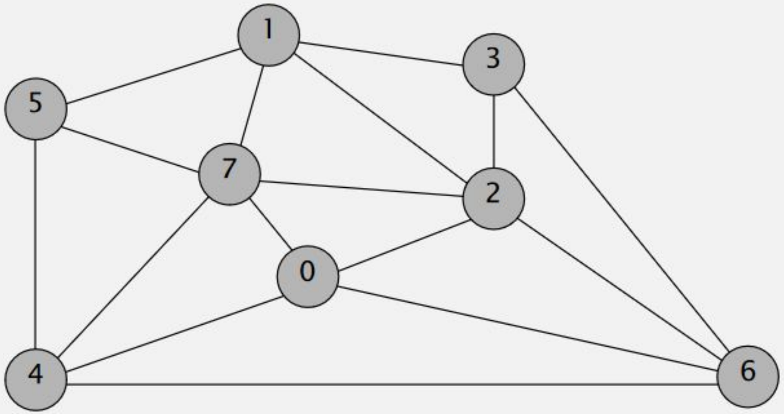


priority queue of crossing edges

```
1-7 0.19
0-2 0.26
5-7 0.28
2-7 0.34
4-7 0.37
0-4 0.38
6-0 0.58
```

Prim's algorithm: lazy implementation demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V-1$ edges.



an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Lazy Prim's algorithm: running time

Proposition. Lazy Prim's algorithm computes the MST in time proportional to $E \log E$ and extra space proportional to E (in the worst case).

minor defect

Pf.

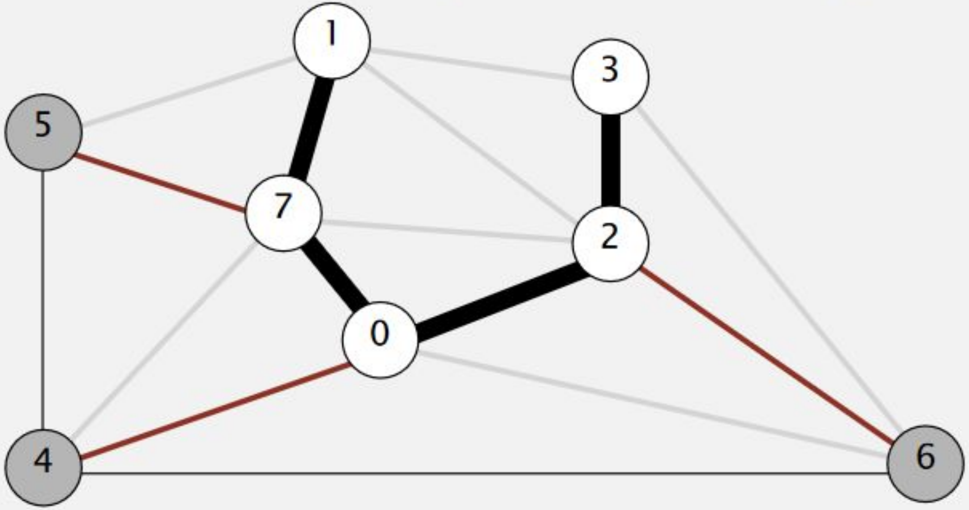
operation	frequency	binary heap
DELETE-MIN	E	$\log E$
INSERT	E	$\log E$

Prim's algorithm: eager implementation

Challenge. Find min weight edge with exactly one endpoint in T.

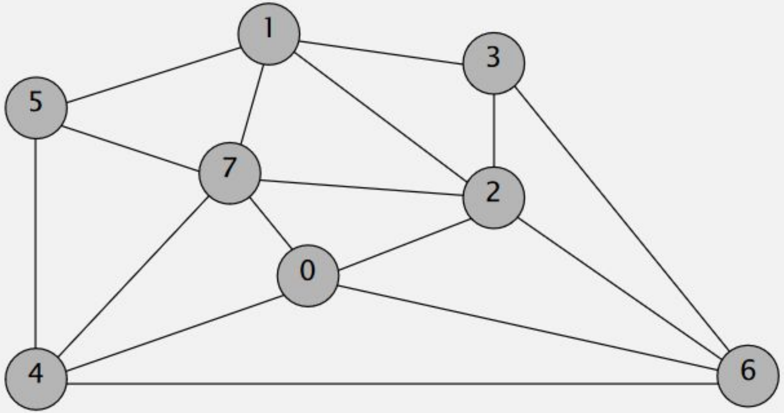
Observation. For each vertex v, need only lightest edge connecting v to T.

- MST includes at most one edge connecting v to T. Why?
- If MST includes such an edge, it must take lightest such edge. Why?



Prim's algorithm: eager implementation demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

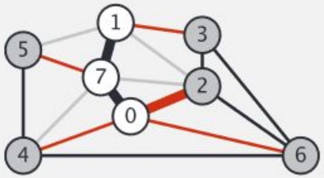
Prim's algorithm: eager implementation

Challenge. Find min weight edge with exactly one endpoint in T.

PQ has at most one entry per vertex

Eager solution. Maintain a PQ of vertices connected by an edge to T, where priority of vertex v = weight of lightest edge connecting v to T.

- Delete min vertex v; add its associated edge $e = v–w$ to T.
- Update PQ by considering all edges $e = v–x$ incident to v
  - ignore if x is already in T
  - add x to PQ if not already on it
  - decrease priority of x if $v–x$ becomes lightest edge connecting x to T



0		
1	1-7	0.19
2	0-2	0.26
3	1-3	0.29
4	0-4	0.38
5	5-7	0.28
6	6-0	0.58
7	0-7	0.16

red: on PQ

black: on MST

Prim's algorithm: Complexity

Depends on PQ implementation: V INSERT, V DELETE-MIN, $\leq E$ DECREASE-KEY.

PQ implementation	INSERT	DELETE-MIN	DECREASE-KEY	total
unordered array	1	V	1	V^2
binary heap	$\log V$	$\log V$	$\log V$	$E \log V$
d-way heap	$\log_d V$	$d \log_d V$	$\log_d V$	$E \log_{E/V} V$
Fibonacci heap	1 †	$\log V$ †	1 †	$E + V \log V$

† amortized

Bottom line.

- Array implementation optimal for complete graphs.
- Binary heap much faster for sparse graphs.
- 4-way heap worth the trouble in performance-critical situations.
- Fibonacci heap best in theory, but not worth implementing.

Това е всичко за днес.

Какво следва:

- Лекция 4 за графи. Ойлеров и Хамилтонов цикъл в граф. Дефиниция на NP complete проблем. Контролно 6.
- Преговорна лекция на материала. Примерни интервю въпроси. Контролно 7
- 27.01.2018 9:00 изпит