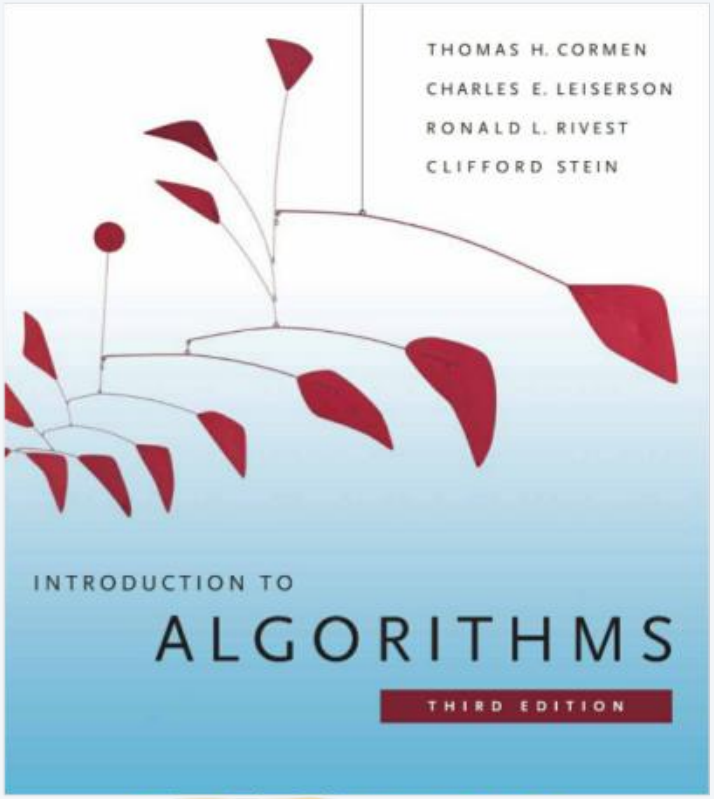


Chapter 23

4. Greedy Algorithms II

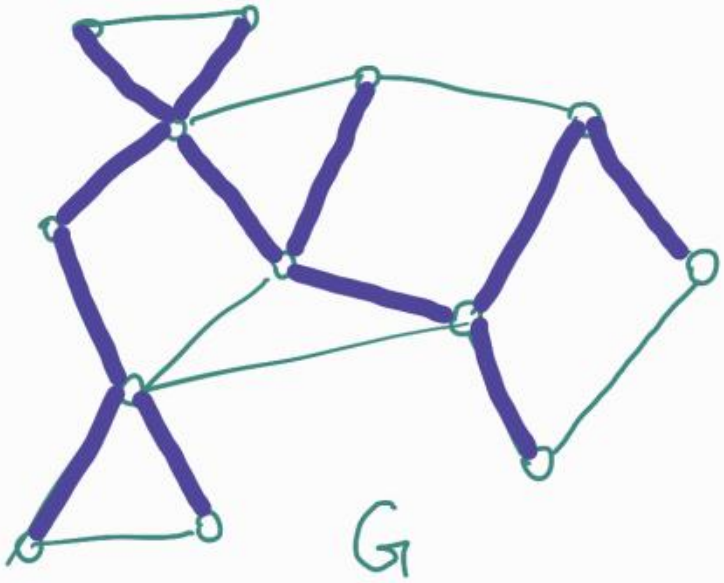
- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*
- *min-cost arborescences*

درخت فراگیر کمینه

الگوریتم پریم

الگوریتم کراسکال

درخت فراگیر Spanning Tree



G

یک گراف داده شده همبند داده شده است

$G = (V, E)$

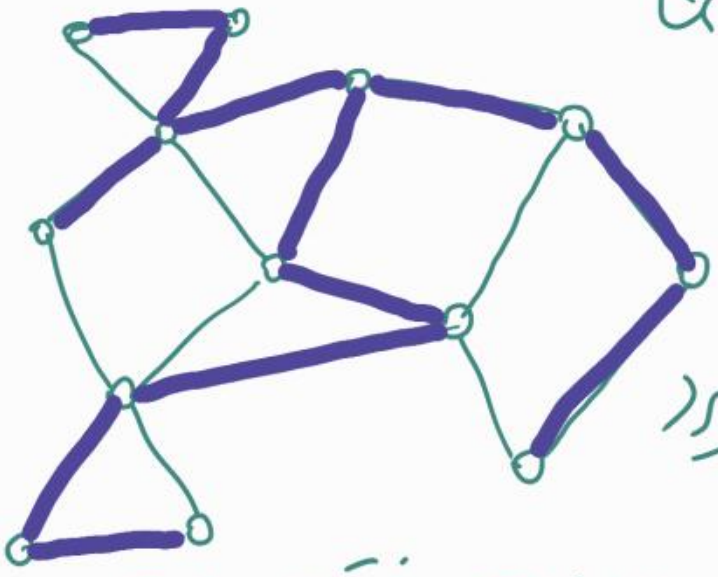
V: مجموعه رأس‌ها

E: مجموعه یال‌ها

یک درخت فراگیر برای گراف G

زیرگرافی از G است که

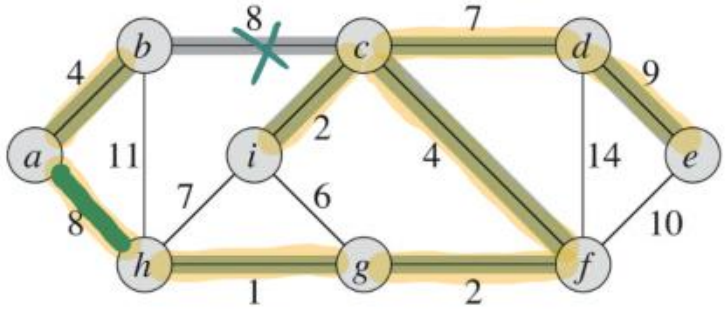
تمام رأس‌های G را در بر می‌گیرد



و یک درخت است؛ یعنی همبند است و فاقد دور است.

Minimum Spanning Tree درخت فراگیر کمینه

(MST)



$w(T) = 4+8+2+7+4+9+2+1$

علاوه بر گراف $G=(V,E)$

یک تابع وزن گذاری بر روی

مجموعهٔ یالها داده شده است:

$$w : E \rightarrow \mathbb{R}$$

- در مسئلهٔ MST هدف بدست آوردن درخت فراگیری است که مجموع وزن یالهای آن کمینه باشد.

Cycles and cuts

دورها و برش‌ها

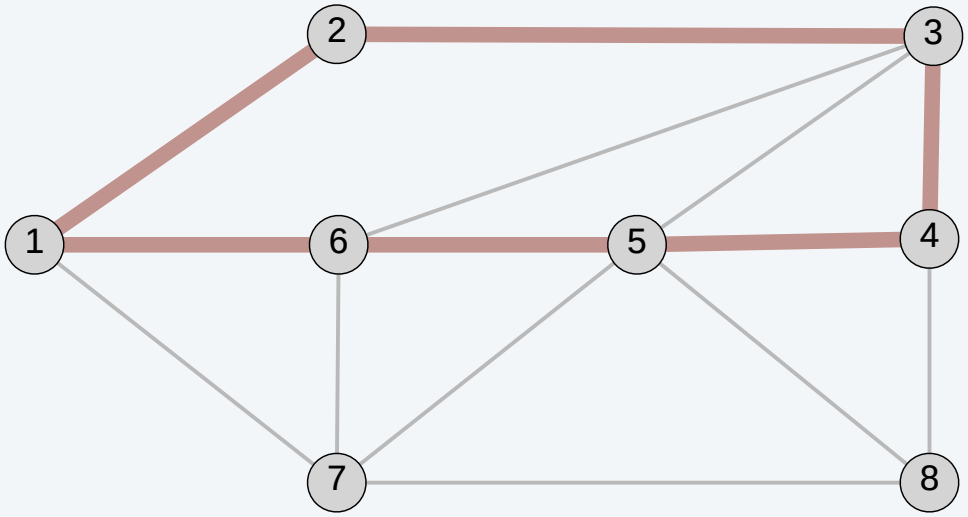
Def. A path is a sequence of edges which connects a sequence of nodes.

Walk

گشت یا قدم‌زنی رأس‌ها می‌توانند تکراری باشند

Def. A cycle is a path with no repeated nodes or edges other than the starting and ending nodes.

مسیر
یک قدم‌زنی که هیچ رأس تکراری ندارد



cycle C = { (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1) }

Cycles and cuts

برش

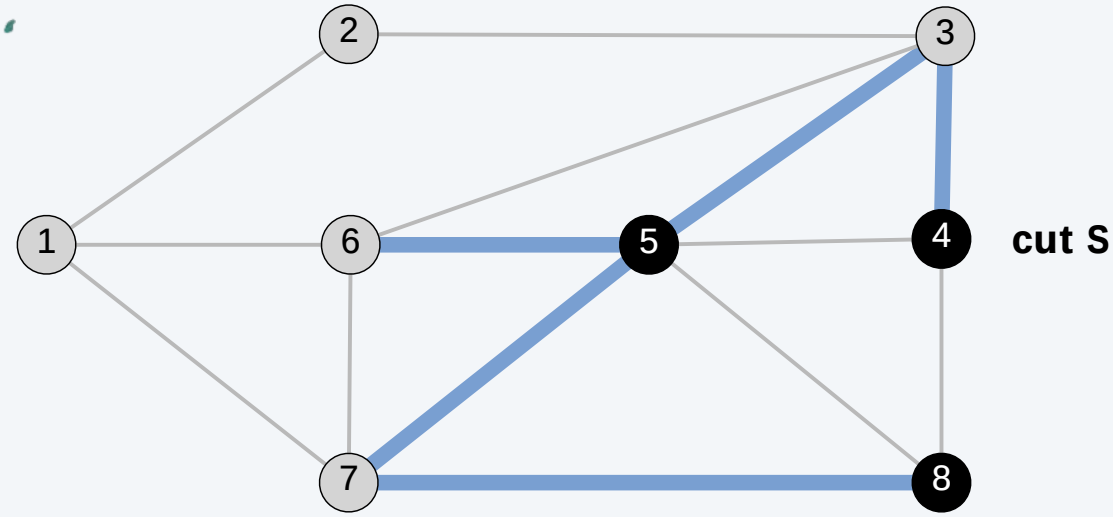
Def. A cut is a partition of the nodes into two nonempty subsets S and $V - S$.

Def. The cutset of a cut S is the set of edges with exactly one endpoint in S.

مجموعه برشی

$S = \{1, 2, 3, 6, 7\}$

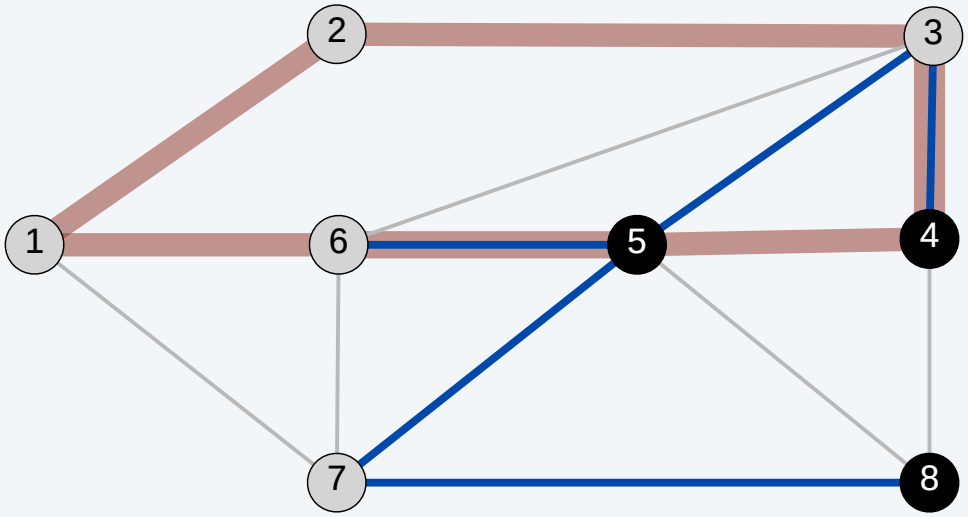
$V - S = \{4, 5, 8\}$



cutset D = { (3, 4), (3, 5), (5, 6), (5, 7), (8, 7) }

Cycle-cut intersection

Proposition. A cycle and a cutset intersect in an even number of edges.



cutset D = { (3, 4), (3, 5), (5, 6), (5, 7), (8, 7) }

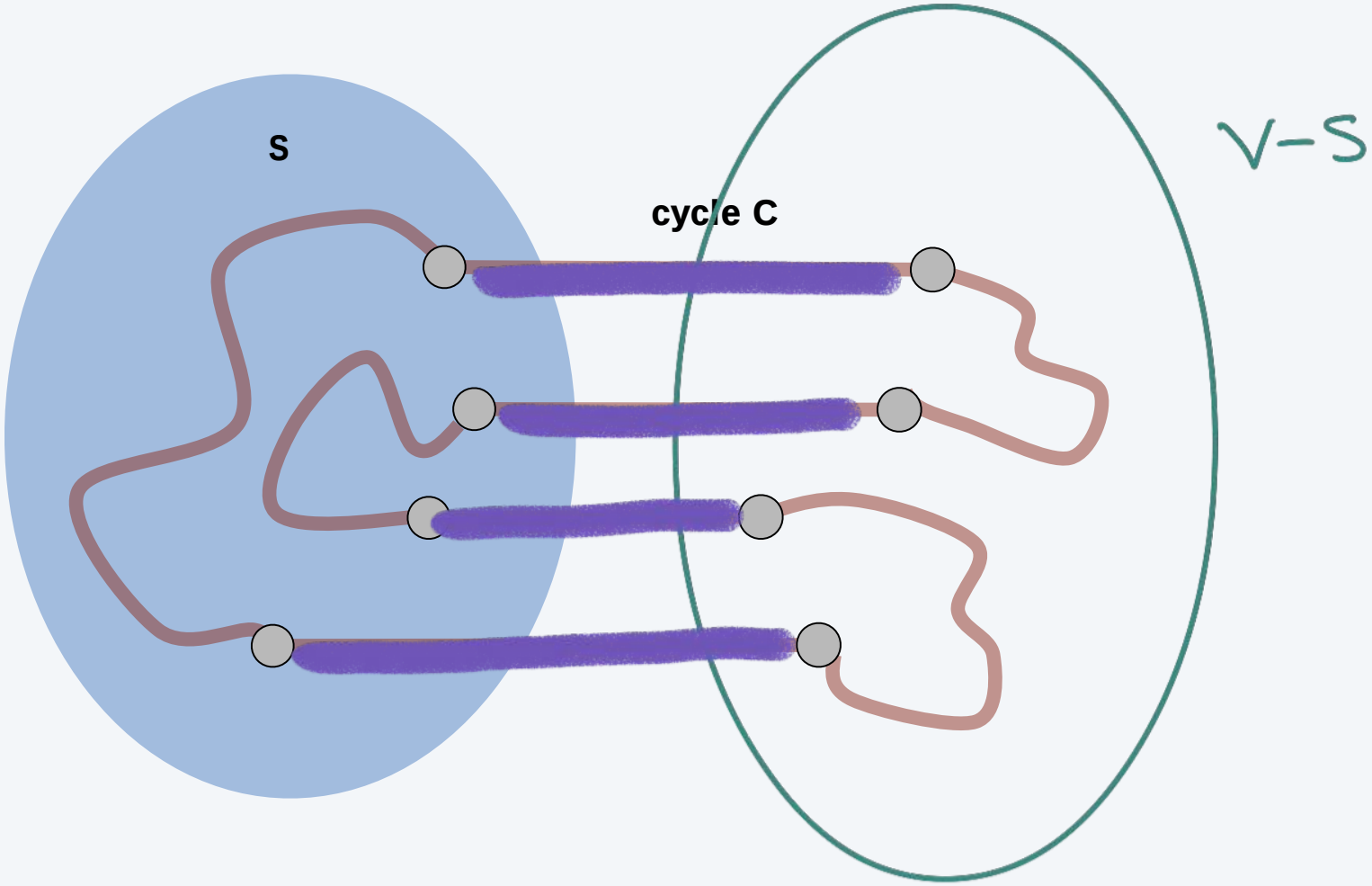
cycle C = { (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1) }

intersection C ∩ D = { (3, 4), (5, 6) }

Cycle-cut intersection

Proposition. A cycle and a cutset intersect in an even number of edges.

Pf. [by picture]

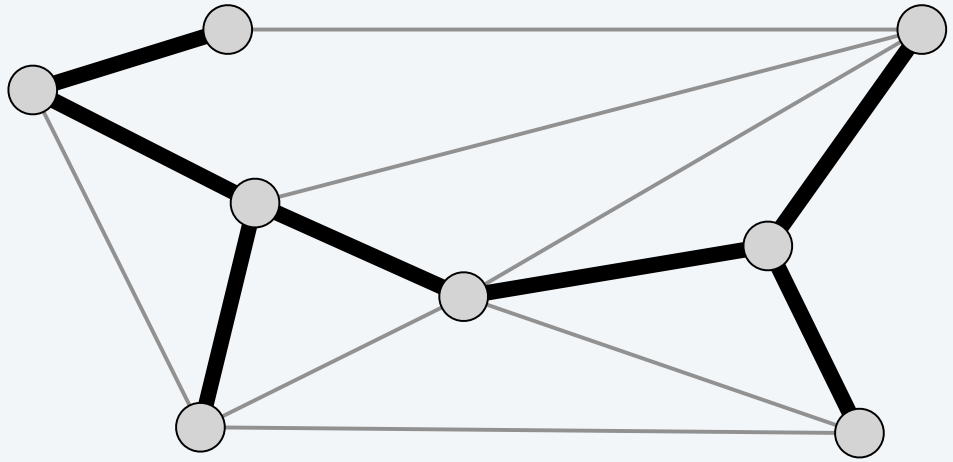


Spanning tree properties

Proposition. Let $T = (V, F)$ be a subgraph of $G = (V, E)$. TFAE:

1. T is a spanning tree of G.
   - T is acyclic and connected.
2. T is connected and has $n - 1$ edges.
3. T is acyclic and has $n - 1$ edges.
4. T is minimally connected: removal of any edge disconnects it.
5. T is maximally acyclic: addition of any edge creates a cycle.
6. T has a unique simple path between every pair of nodes.

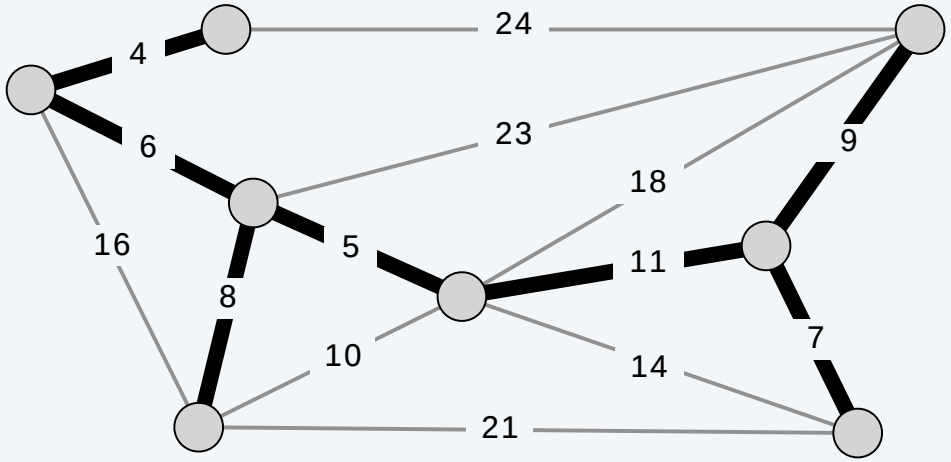
تعاریف هم ارز برای درخت فراگیر



T = (V, F)

Minimum spanning tree

Given a connected graph $G = (V, E)$ with edge costs c_e, an MST is a subset of the edges $T \subseteq E$ such that T is a spanning tree whose sum of edge costs is minimized.



MST cost = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7

گراف کامل n راسی

Cayley's theorem. There are n^{n-2} spanning trees of K_n. ← can't solve by brute force

Applications

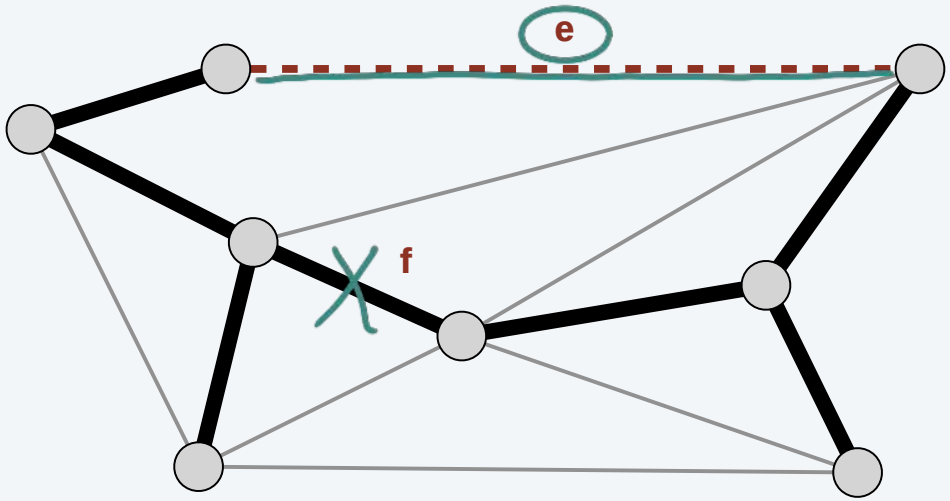
MST is fundamental problem with diverse applications.

- Dithering.
- Cluster analysis.
- Max bottleneck paths.
- Real-time face verification.
- LDPC codes for error correction.
- Image registration with Renyi entropy.
- Find road networks in satellite and aerial imagery.
- Reducing data storage in sequencing amino acids in a protein.
- Model locality of particle interactions in turbulent fluid flows.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Approximation algorithms for NP-hard problems (e.g., TSP, Steiner tree).
- Network design (communication, electrical, hydraulic, computer, road).

Fundamental cycle

Fundamental cycle.

- Adding any non-tree edge e to a spanning tree T forms unique cycle C.
- Deleting any edge $f \in C$ from $T \cup \{ e \}$ results in new spanning tree.



T = (V, F)

$T \rightarrow T \cup \{e\} - \{f\}$

Observation. If $c_e < c_f$, then T is not an MST.

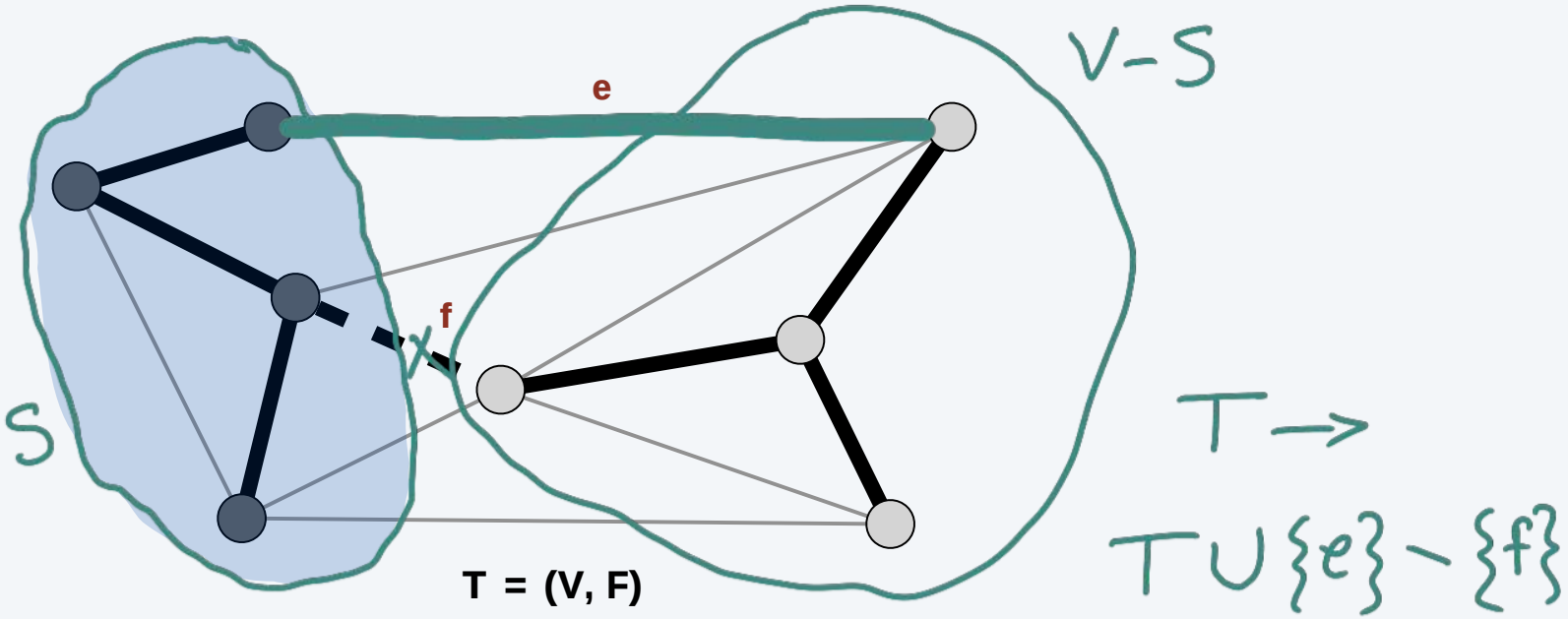
دور پایه‌ای

Fundamental cutset

مجموعه برشی پایه ای

Fundamental cutset.

- Deleting any tree edge f from a spanning tree T divide nodes into two connected components. Let D be cutset.
- Adding any edge $e \in D$ to $T - \{ f \}$ results in new spanning tree.



Observation. If $c_e < c_f$, then T is not an MST.

مجموعه برشی پایه ای

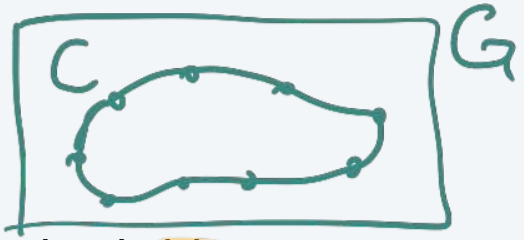
The greedy algorithm

می‌توان گفت تمام انواع الگوریتم‌های حریصانه برای حل MST یکی از دو قاعدهٔ حریصانه زیر یا ترکیبی از آن‌ها را استفاده می‌کنند

Red rule.

- Let C be a cycle with no red edges.
- Select an uncolored edge of C of max weight and color it red.

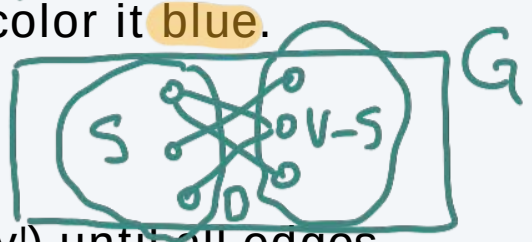
یال قرمز: یالی که تضمین داریم در هیچ MST قرار نمی‌گیرد



Blue rule.

- Let D be a cutset with no blue edges.
- Select an uncolored edge in D of min weight and color it blue.

یال آبی: یالی که تضمین داریم در یک MST قرار می‌گیرد.



Greedy algorithm.

- Apply the red and blue rules (non-deterministically!) until all edges are colored. The blue edges form an MST.
- Note: can stop once $n-1$ edges colored blue.

الگوی کلی الگوریتم‌های حریصانه مبتنی بر کاربست "قاعده آبی" برای حل مسئلهٔ MST

```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3      find an edge (u, v) that is "safe" for A
4      A = A ∪ {(u, v)}
5  return A
```

Greedy algorithm: proof of correctness

Color invariant. There exists an MST T^* containing all of the blue edges and none of the red edges.

Pf. [by induction on number of iterations]

Base case. No edges colored $\Rightarrow$ every MST satisfies invariant.

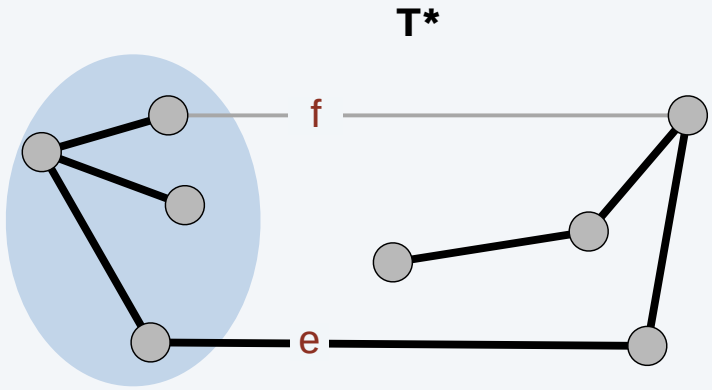
Greedy algorithm: proof of correctness

Color invariant. There exists an MST T^* containing all of the blue edges and none of the red edges.

Pf. [by induction on number of iterations]

Induction step (blue rule). Suppose color invariant true before blue rule.

- let D be chosen cutset, and let f be edge colored blue.
- if $f \in T^*$, T^* still satisfies invariant.
- Otherwise, consider fundamental cycle C by adding f to T^*.
- let $e \in C$ be another edge in D.
- e is uncolored and $c_e \geq c_f$ since
  - $e \in T^* \Rightarrow e$ not red
  - blue rule $\Rightarrow e$ not blue and $c_e \geq c_f$
- Thus, $T^* \cup \{ f \} - \{ e \}$ satisfies invariant.



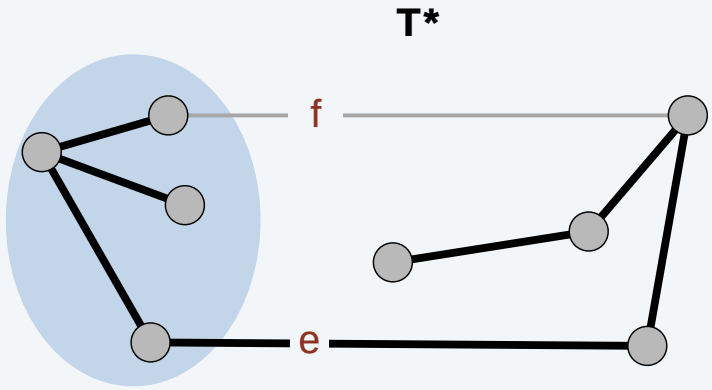
Greedy algorithm: proof of correctness

Color invariant. There exists an MST T^* containing all of the blue edges and none of the red edges.

Pf. [by induction on number of iterations]

Induction step (red rule). Suppose color invariant true before red rule.

- let C be chosen cycle, and let e be edge colored red.
- if $e \notin T^*$, T^* still satisfies invariant.
- Otherwise, consider fundamental cutset D by deleting e from T^*.
- let $f \in D$ be another edge in C.
- f is uncolored and $c_e \geq c_f$ since
  - $f \notin T^* \Rightarrow f$ not blue
  - red rule $\Rightarrow f$ not red and $c_e \geq c_f$
- Thus, $T^* \cup \{ f \} - \{ e \}$ satisfies invariant. ▪



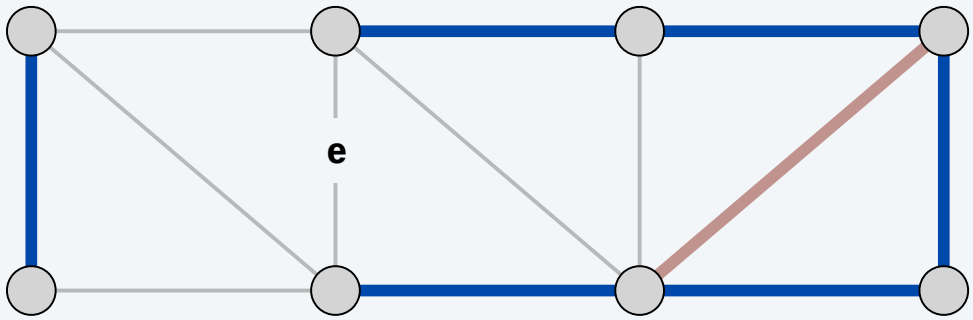
Greedy algorithm: proof of correctness

Theorem. The greedy algorithm terminates. Blue edges form an MST.

Pf. We need to show that either the red or blue rule (or both) applies.

- Suppose edge e is left uncolored.
- Blue edges form a forest.
- Case 1: both endpoints of e are in same blue tree.

  $\Rightarrow$ apply red rule to cycle formed by adding e to blue forest.



Case 1

Greedy algorithm: proof of correctness

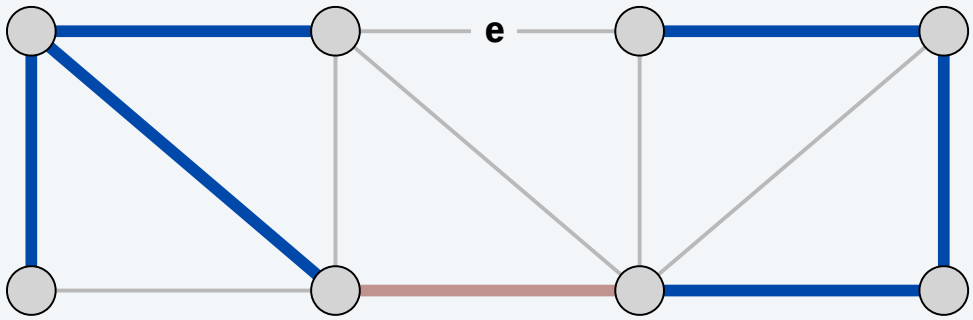
Theorem. The greedy algorithm terminates. Blue edges form an MST.

Pf. We need to show that either the red or blue rule (or both) applies.

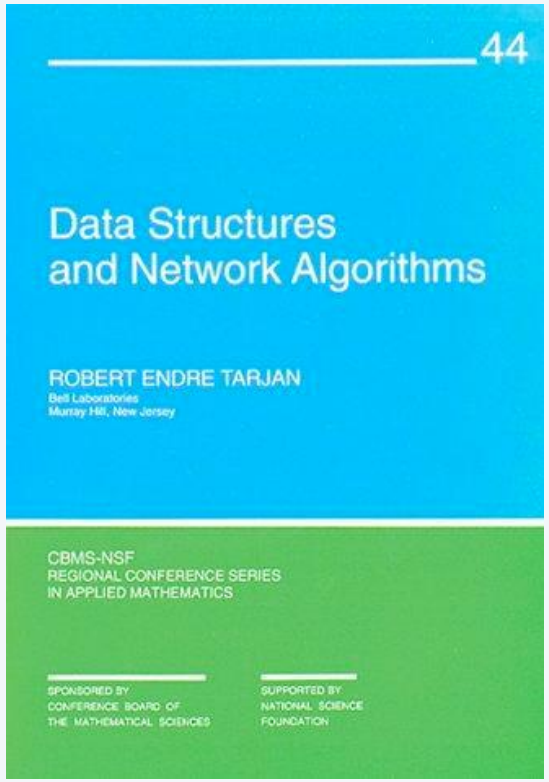
- Suppose edge e is left uncolored.
- Blue edges form a forest.
- Case 1: both endpoints of e are in same blue tree.

  $\Rightarrow$ apply red rule to cycle formed by adding e to blue forest.
- Case 2: both endpoints of e are in different blue trees.

  $\Rightarrow$ apply blue rule to cutset induced by either of two blue trees. ▪



Case 2



SECTION 6.2

4. GREEDY ALGORITHMS II

- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*
- *min-cost arborescences*

Prim's algorithm

الگوریتم پریم

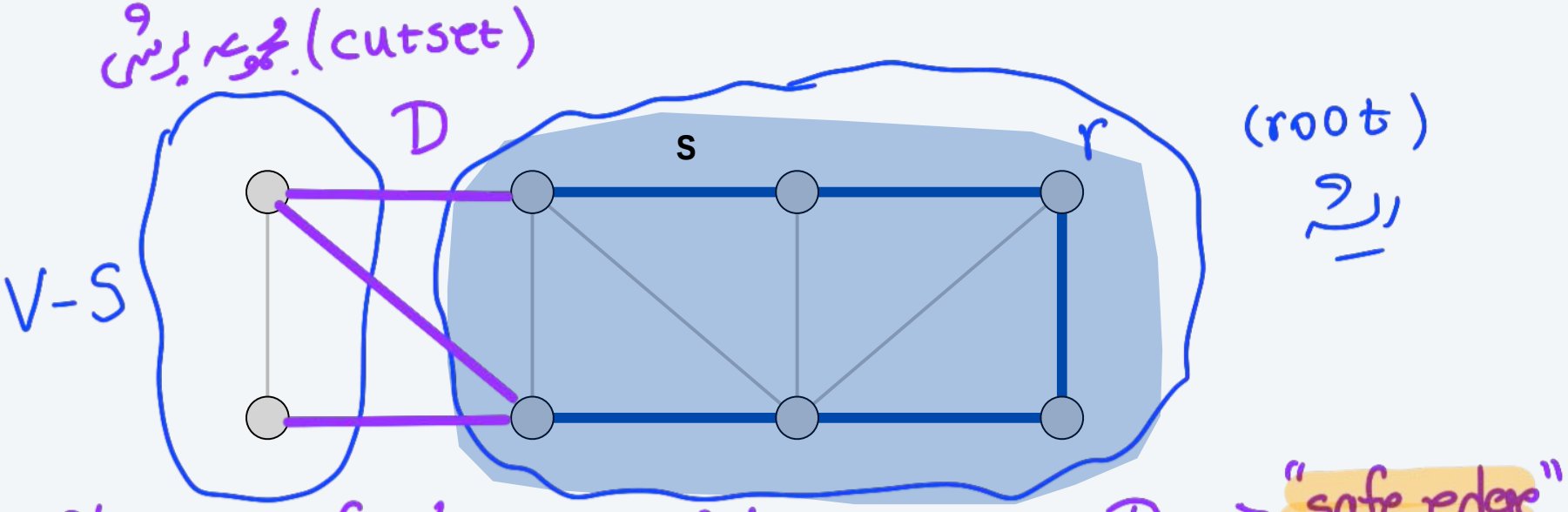
Initialize S = any node.

Repeat $n - 1$ times:

- Add to tree the min weight edge with one endpoint in S.
- Add new node to S.

Theorem. Prim's algorithm computes the MST.

Pf. Special case of greedy algorithm (blue rule repeatedly applied to S). ▪



Prim's algorithm: implementation

Theorem. Prim's algorithm can be implemented in $O(m \log n)$ time.

Pf. Implementation almost identical to Dijkstra's algorithm.

[$d(v)$ = weight of cheapest known edge between v and S]

PRIM (V, E, c)

Create an empty priority queue.

$s \leftarrow$ any node in V.

FOR EACH $v \neq s$: $d(v) \leftarrow \infty$; $d(s) \leftarrow 0$.

FOR EACH v : *insert* v with key $d(v)$ into priority queue.

WHILE (the priority queue *is not empty*)

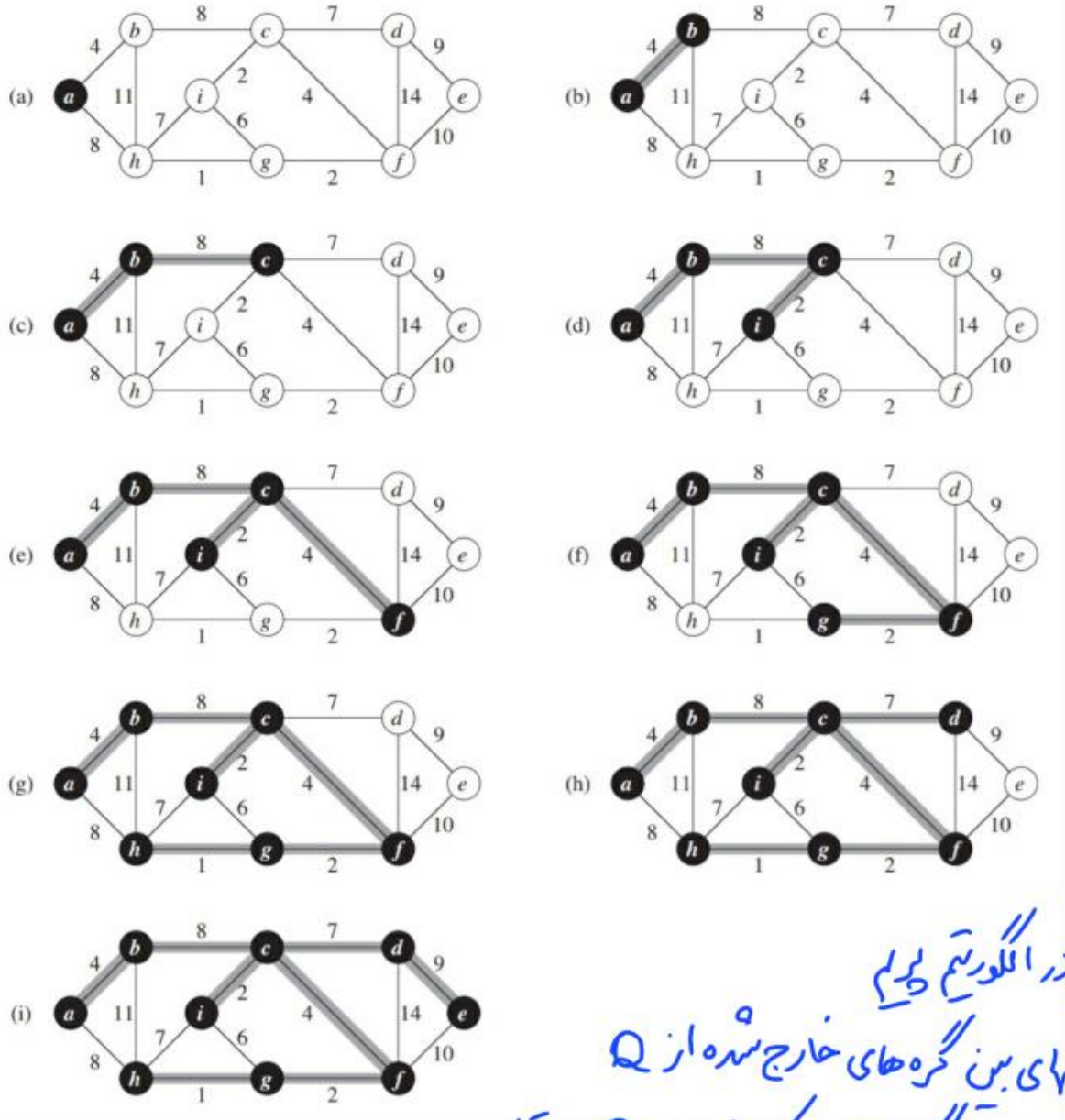
  $u \leftarrow$ *delete-min* from priority queue.

  FOR EACH edge $(u, v) \in E$ incident to u:

    IF $d(v) > c(u, v)$

      decrease-key of v to $c(u, v)$ in priority queue.

      $d(v) \leftarrow c(u, v)$.



مجموعه بررسی در الگوریتم پریم

مجموعه یالهای بین گره‌های خارج شده از Q و گره‌هایی است که هنوز در Q هستند.

شبه کد الگوریتم پریم

```
MST-PRIM(G, w, r)                                   Q: صف اولویت بندی
1   for each u ∈ G.V
2       u.key = ∞                                   u.key
3       u.π = NIL                                   مقدار کلید متناظر با گره u
4   r.key = 0                                       نشان دهنده وزن سبک ترین یالی است که می تواند u را به درخت فراگیر کمینه در حال گسترش متصل کند.
5   Q = G.V          //insert every node u∈G.V into Q
6   while Q ≠ ∅
7       u = EXTRACT-MIN(Q)          //find min weight
8       for each v ∈ G.Adj[u]
9           if v ∈ Q and w(u, v) < v.key
10              v.π = u                             u.π
11              v.key = w(u, v)          //decrease-key
                                                    گره والد: یعنی گره ای که از طریق آن u به ریشه می رسد.
```

مقداردهی اولیه

الگوی حلی الگوریتم حریصانه

تحلیل هزینه زمان الگوریتم پریم

```
MST-PRIM(G, w, r)
1  for each u ∈ G.V
2      u.key = ∞
3      u.π = NIL
4  r.key = 0
5  Q = G.V
6  while Q ≠ ∅
7      u = EXTRACT-MIN(Q)
8      for each v ∈ G.Adj[u]
9          if v ∈ Q and w(u, v) < v.key
10             v.π = u
11             v.key = w(u, v)
```

O(V)

O(1)

O(V)

تکرار V

O(V)

O(deg u)

O(1)

وارداد: (کتاب CLRS)

تعداد رأسهای گراف

$|V|$, V v , n

تعداد یالهای گراف

$|E|$, E e , m

تحلیل هزینه حلقه while

فرض کنید Q یک لیست پیوندی معمولی است

$$\text{worst case time} = V^2 + \sum_{u \in V} \deg u$$

تکرارهای حلقه for داخل حلقه while

$$= O(V^2 + E)$$

تحلیل هزینه زمان الگوریتم پریم

```
MST-PRIM(G, w, r)
1  for each u ∈ G.V                          O(V)
2      u.key = ∞
3      u.π = NIL
4  r.key = 0                                 O(1)
5  Q = G.V                                   O(V)
6  while Q ≠ ∅                               تکرار V
7      u = EXTRACT-MIN(Q)                    O(log V)
8      for each v ∈ G.Adj[u]                 O(deg u)
9          if v ∈ Q and w(u, v) < v.key
10             v.π = u                       O(log V)
11         v.key = w(u, v) //Decrease-key
```

قرارداد: (کتاب CLRS)

تعداد رأسهای گراف

$|V|$, V — v, n

تعداد یالهای گراف

$|E|$, E — e, m

فرض کنید Q را توسط ساختار داده‌ای Heap پیاده‌سازی کرده‌ایم:

$$\text{worst case time} = V.\log V + \sum_{u \in V} (\deg u).(\log V)$$

$$= O(V.\log V + E.\log V)$$

در گراف همبند $= \boxed{O(E.\log V)}$

توجه کنید عمدتا در مسائل دنیای کاربرد، گرافها، گرافهای خلوت یا تُنُک (sparse) هستند که

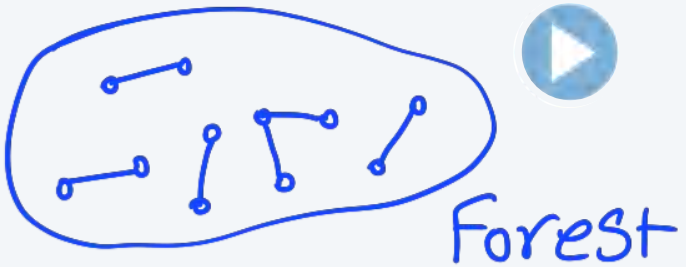
$E = O(V) \Rightarrow O(V.\log V)$

Kruskal's algorithm

الگوریتم کراسکال

Consider edges in ascending order of weight:

- Add to tree unless it would create a cycle.



Theorem. Kruskal's algorithm computes the MST.

Pf. Special case of greedy algorithm.

- Case 1: both endpoints of e in same blue tree.

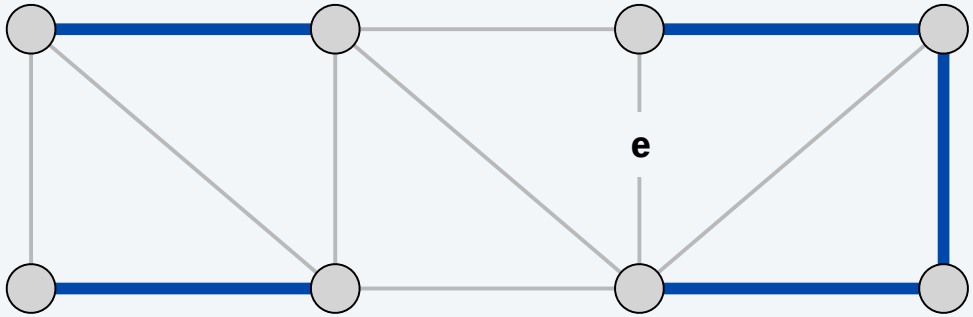
  ⇒ color red by applying red rule to unique cycle.

  all other edges in cycle are blue

- Case 2. If both endpoints of e are in different blue trees.

  ⇒ color blue by applying blue rule to cutset defined by either tree. ▪

  no edge in cutset has smaller weight (since Kruskal chose it first)



Kruskal's algorithm: implementation

Theorem. Kruskal's algorithm can be implemented in $O(m \log m)$ time.

- Sort edges by weight.
- Use union-find data structure to dynamically maintain connected components.

KRUSKAL (V, E, c)

SORT m edges by weight so that $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$

$S \leftarrow \phi$

FOREACH $v \in V$: MAKESET(v).

FOR $i = 1$ TO m

  $(u, v) \leftarrow e_i$

  IF FINDSET$(u) \neq$ FINDSET(v) ← are u and v in same component?

    $S \leftarrow S \cup \{ e_i \}$

    UNION(u, v). ← make u and v in same component

RETURN S

Reverse-delete algorithm

Consider edges in descending order of weight:

- Remove edge unless it would disconnect the graph.

Theorem. The reverse-delete algorithm computes the MST.

Pf. Special case of greedy algorithm.

- Case 1: removing edge e does not disconnect graph.

  $\Rightarrow$ apply red rule to cycle C formed by adding e to existing path between its two endpoints

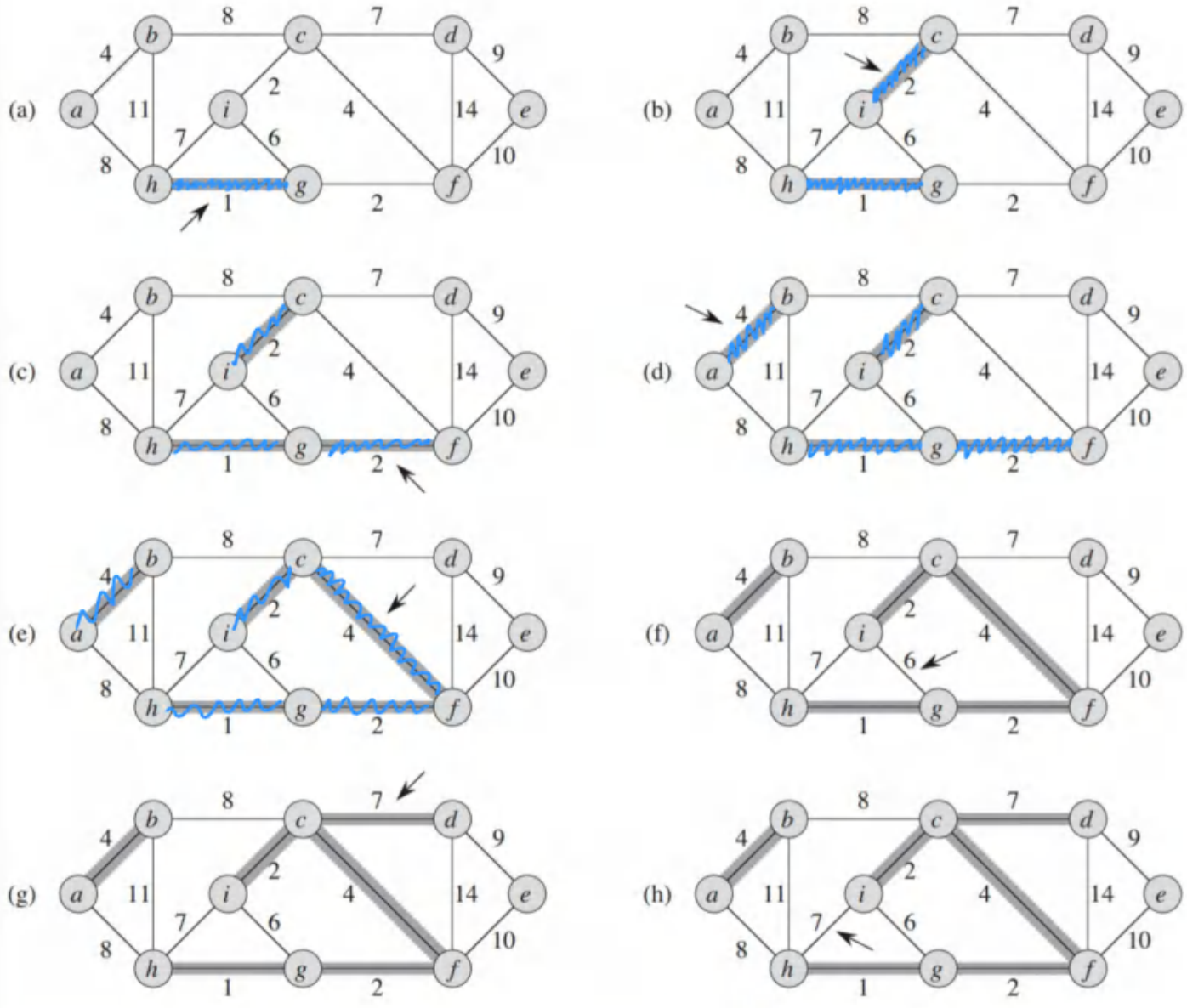
  any edge in C with larger weight would have been deleted when considered

- Case 2: removing edge e disconnects graph.

  $\Rightarrow$ apply blue rule to cutset D induced by either component. ▪

  e is the only edge in the cutset
  (any other edges must have been colored red / deleted)

Fact. [Thorup 2000] Can be implemented in $O(m \log n (\log \log n)^3)$ time.

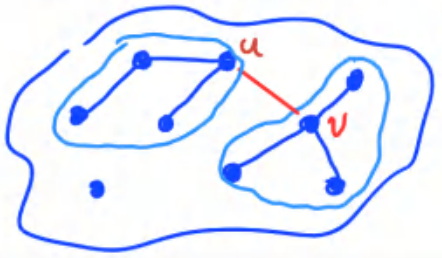



```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

در الگوریتم کراسکال در هر چرخهٔ حلقهٔ تکرار یک افراز از تمام رأس‌های گراف G نگه‌داری می‌شود

این افراز یک جنگل فراگیر Spanning Forest

مثال



برای پیاده‌سازی کراسکال به ساختار داده‌ای مجموعه‌های مجزا (Disjoint Sets) یا (Equivalnet classes)

تحلیل هزینهٔ زمان الگوریتم کراسکال

```
MST-KRUSKAL(G, w)
1  A = ∅                                                  → O(1)
2  for each vertex v ∈ G.V                                → O(V)
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w   → O(E log E)
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight   → تکرار E
6      if FIND-SET(u) ≠ FIND-SET(v)                       → بطور سرشکن شده O(1)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

Worst case "amortized" time $= O(V + E \log E + E)$

با توجه به اینکه در گرافهای ساده (بین هر دو رأس حداکثر یک یال است) داریم

$E = O(V^2) \Rightarrow \log E = O(\log V)$

بنابراین

هزینهٔ زمان کراسکال $= O(V + E \log V)$