Algorithms Theory

12 – Minimum Spanning Trees

Evaluation Period

Forms for evaluating this course will be available on our web-page.

P.D. Dr. Alexander Souza

Tomorrow

30 minute lecture.

Winter term 11/12

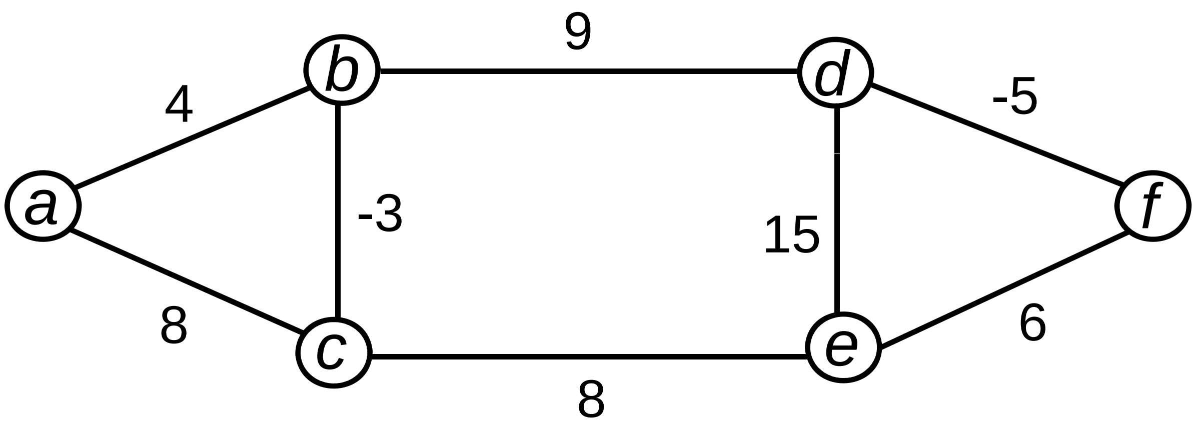
1. Minimum spanning trees

$G = (V, E)$ undirected graph $\qquad w: E \to R$ weight function

Let $T \subseteq E$ be a tree (connected, acyclic subgraph).

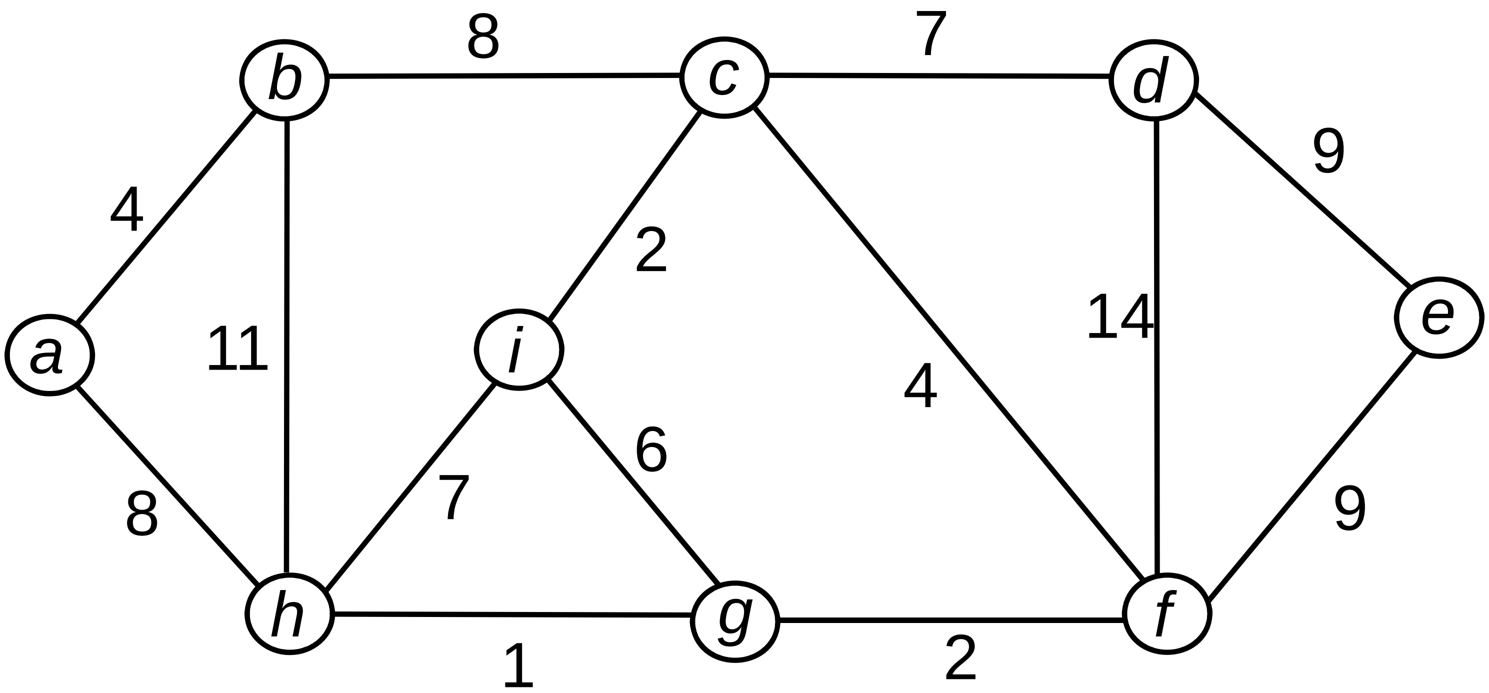
Total weight of T :

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$



Minimum spanning trees

A tree $T \subseteq E$ that connects all vertices in V and whose total weight is minimal is called a minimum spanning tree.



Growing a minimum spanning tree

Invariant: Maintain a set $A \subseteq E$ that is a subset of some minimum spanning tree.

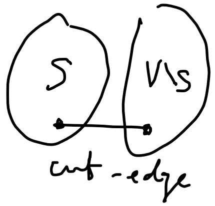
Definition: An edge $(u,v) \in E \setminus A$ is a safe edge for A if $A \cup \{(u,v)\}$ is also a subset of some minimum spanning tree.

Greedy approach

Algorithm Generic-MST(G,w);

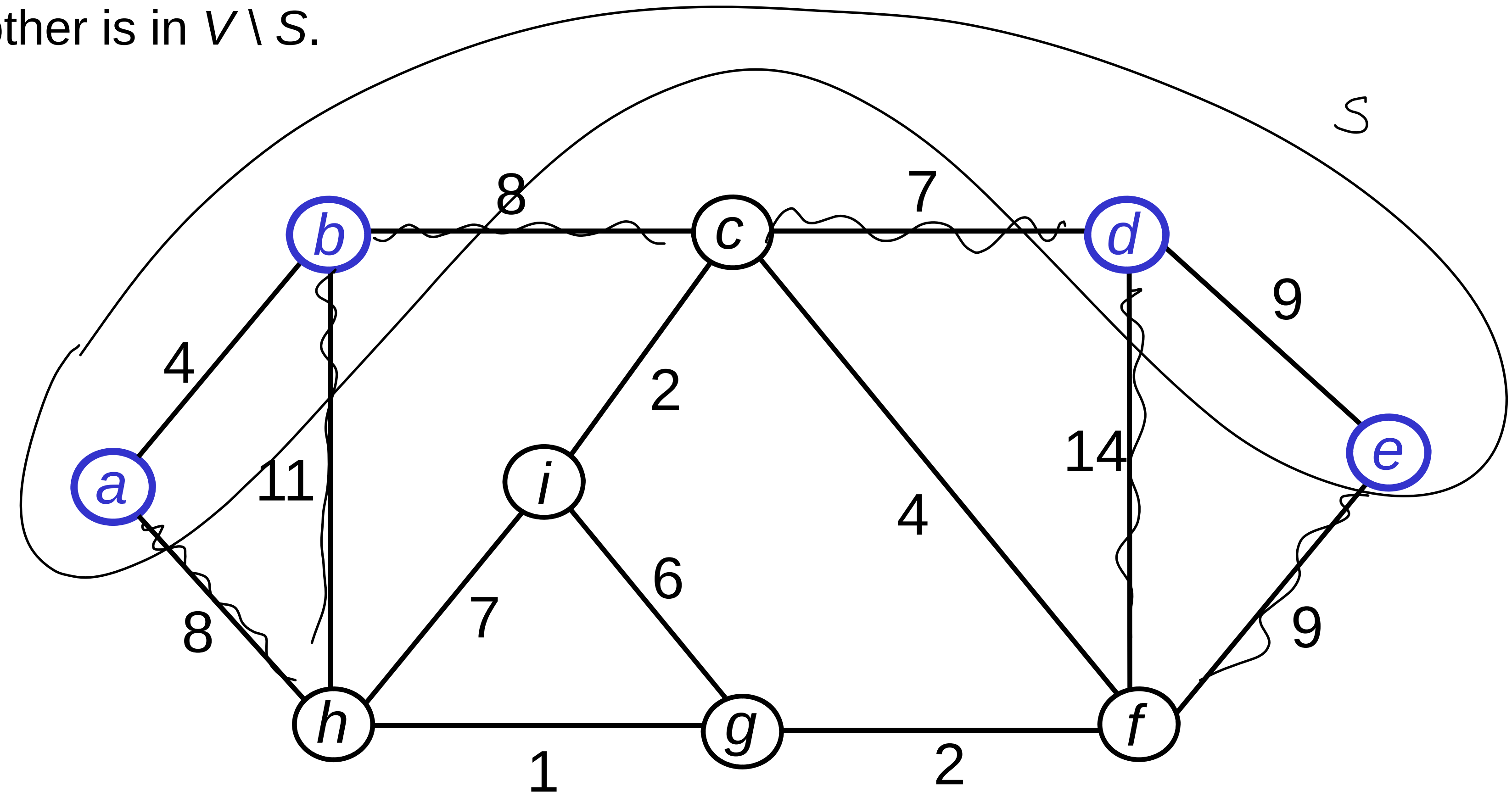
1. $A \leftarrow \emptyset$;
2. **while** A does not form a spanning tree **do**
3. Find an edge (u,v) that is safe for A;
4. $A \leftarrow A \cup \{(u,v)\}$;
5. **endwhile;**

2. Cuts



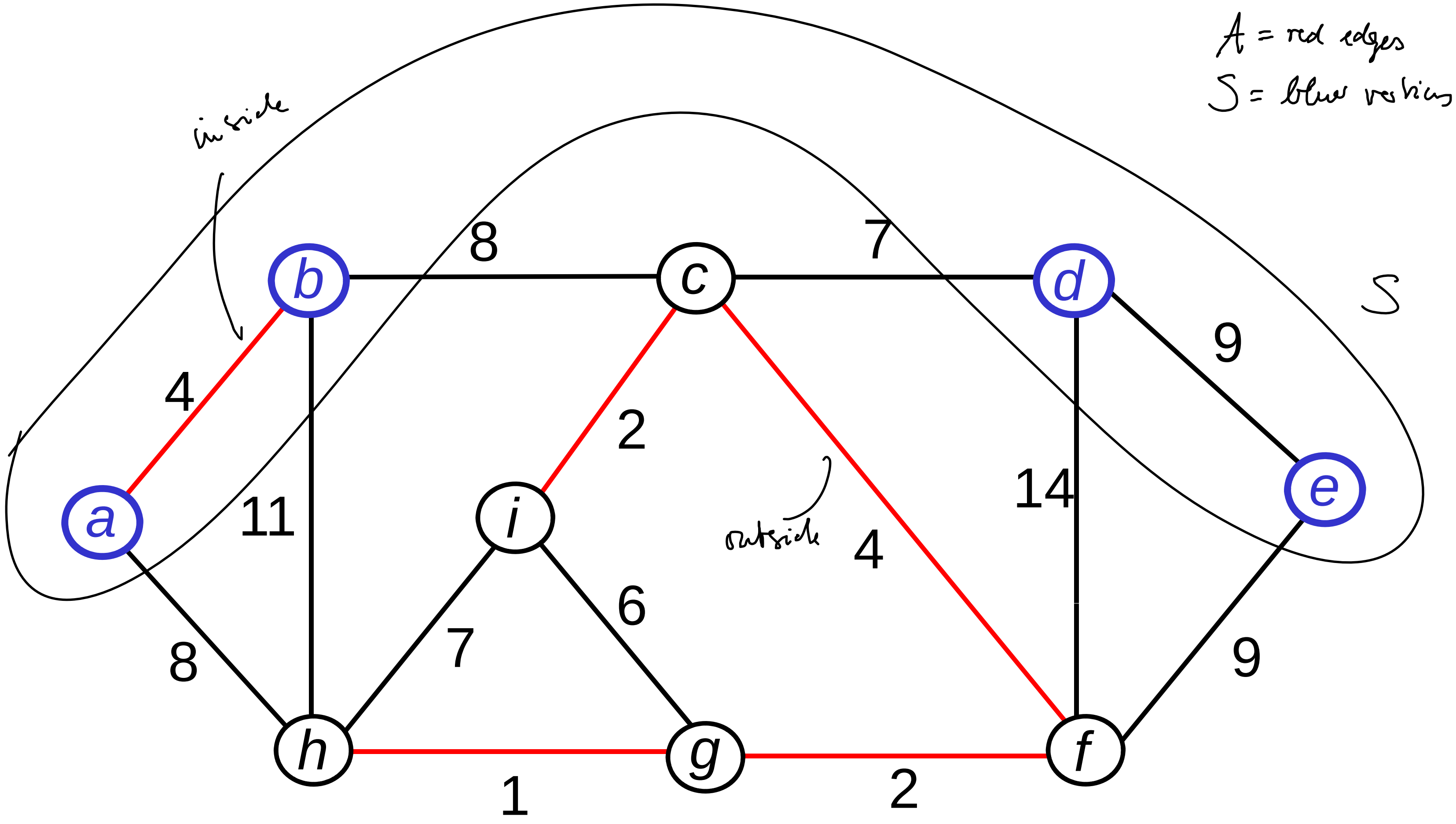
A cut $(S, V \setminus S)$ is a partition of V.

An edge (u,v) crosses $(S, V \setminus S)$ if one of its endpoints is in S and the other is in $V \setminus S$.



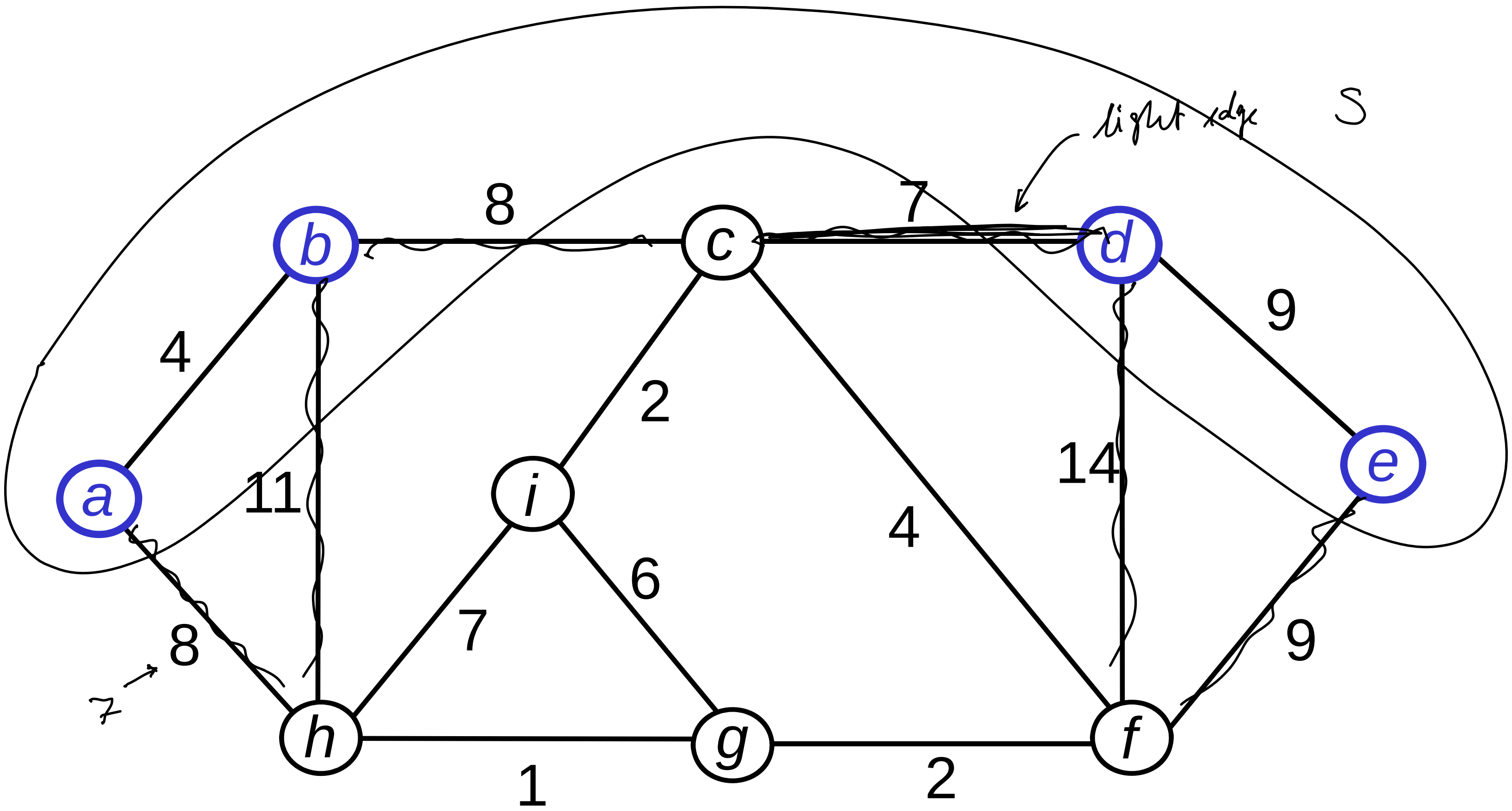
Cuts

A cut respects a set A of edges if no edge in A crosses the cut.



Cuts

An edge is a light edge crossing a certain cut if its weight is the minimum of any edge crossing the cut.



3. Safe edges

Theorem: Let A be a subset of some minimum spanning tree T, and let $(S, V \setminus S)$ be a cut that respects A. If (u,v) is a light edge crossing $(S, V \setminus S)$ then (u,v) is safe for A.

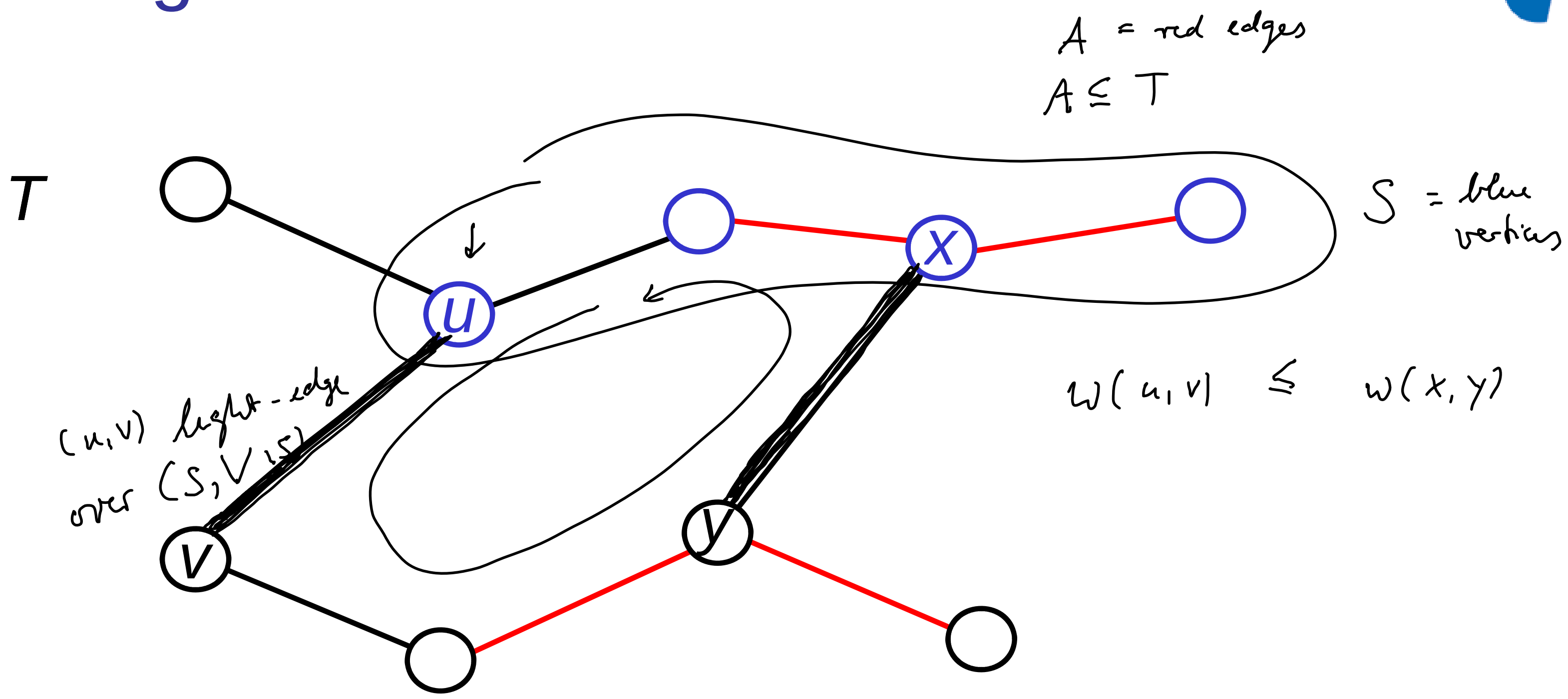
Proof: T fixed MST, $A \subseteq T$, to be shown $A' = A \cup \{(u,v)\} \subseteq T'$ for some MST T'!

Case 1: $(u,v) \in T$: ok ✓

Case 2: $(u,v) \notin T$:

We construct another minimum spanning tree T' with $(u,v) \in T'$ and $A \subseteq T'$.

Safe edges



Adding (u,v) to T yields a cycle.
exactly one

On this cycle, there is at least one edge (x,y) in T that also crosses the cut.

Safe edges

$$T' = T \setminus \{(x,y)\} \cup \{(u,v)\}$$ is a tree

$(u,v) \in T'$

is a minimum spanning tree, since

$$w(T') = w(T) - \overbrace{w(x,y) + w(u,v)}^{\leq 0} \leq w(T)$$

same cut-edge

light edge

$w(u,v) \leq w(x,y)$

$w(T') \geq w(T)$ T is a MST

$\Rightarrow$ $w(T') = w(T)$

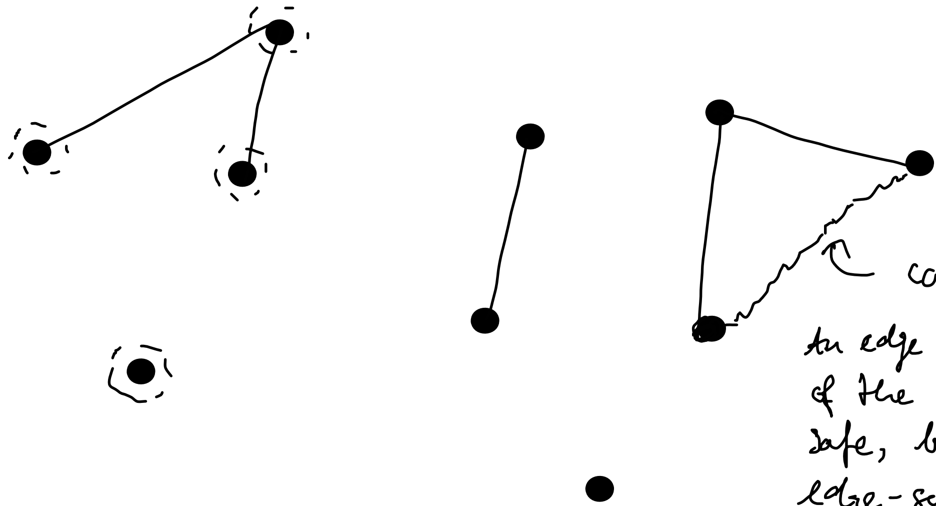
$\Rightarrow$ T' is a MST

$A' = A \cup \{(u,v)\} \subseteq T'$ ✓ $(x,y) \notin A$

4. The graph G_A

$G_A = (V, A)$

- is a forest, i.e. a collection of trees
- at the beginning, when $A = \emptyset$, each tree consists of a single vertex
- any safe edge for A connects distinct trees



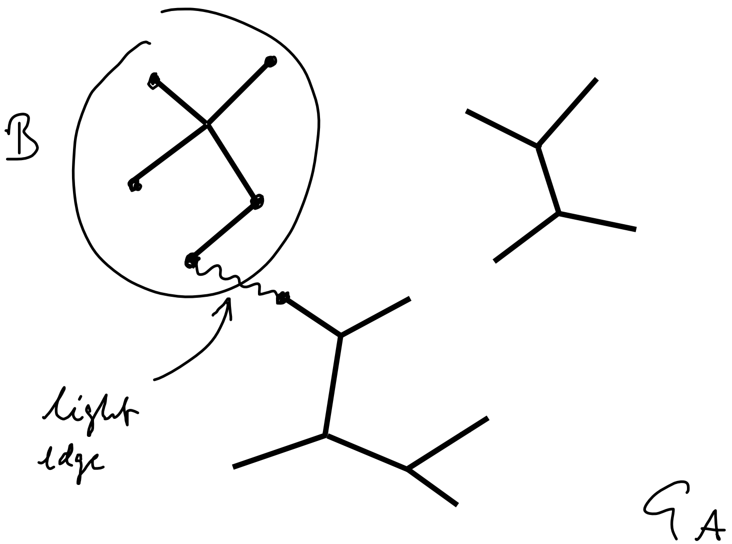
can not happen!

An edge connecting vertices of the same tree is not safe, because the resulting edge-set has a cycle and is hence not subset of a MST.

The graph G_A

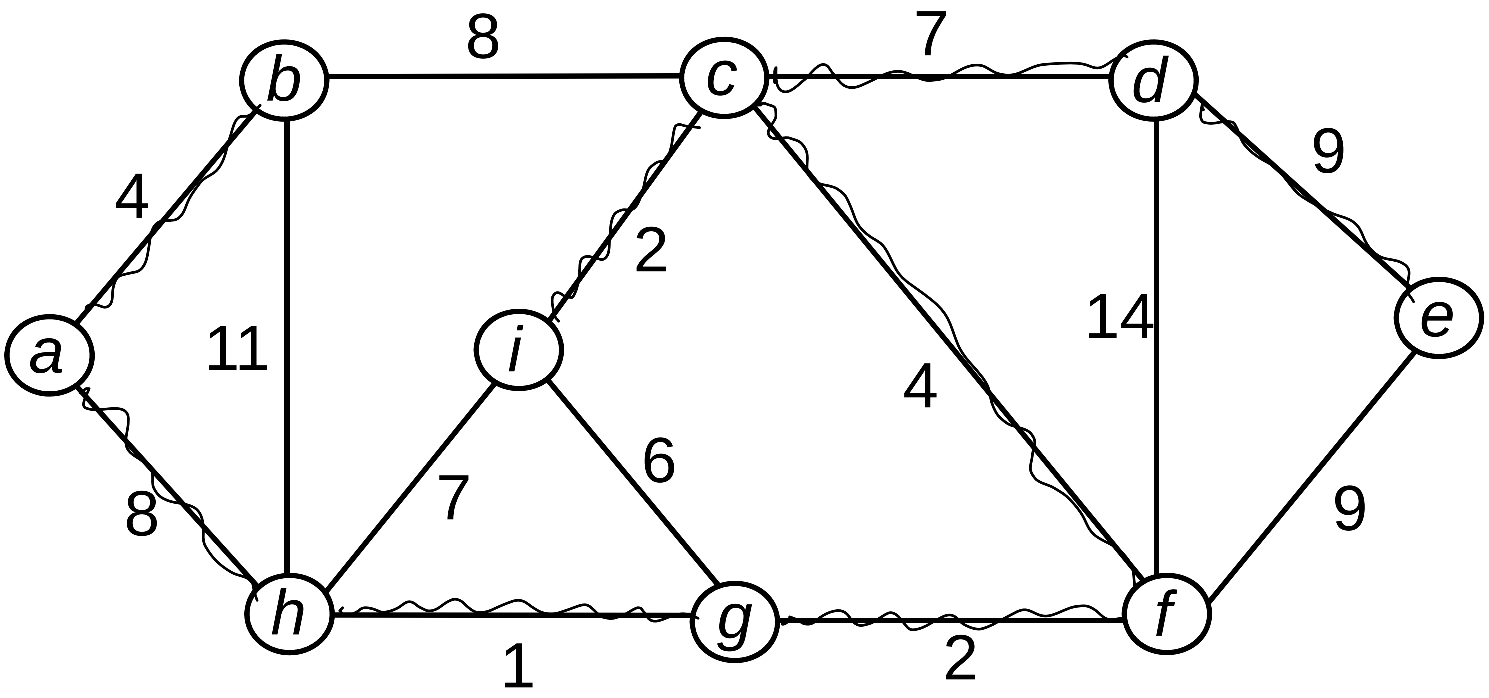
Corollary: Let B be a tree in $G_A = (V, A)$. If (u,v) is a light edge connecting B to some other tree in G_A, then (u,v) is safe for A.

Proof: $(B, V \setminus B)$ respects A and (u,v) is a light edge for this cut. $\xRightarrow{\text{Theorem}}$ (u,v) is safe



5. Kruskal's algorithm

Always choose an edge of smallest weight that connects two trees B_1 and B_2 in G_A.

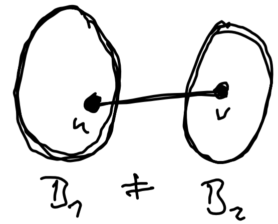


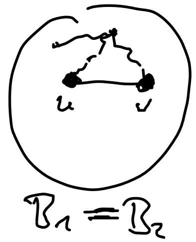
Kruskal's algorithm

In $G_A = G_\emptyset$ all vertices correspond to trees B_v

1. $A \leftarrow \emptyset$;
2. **for all** $v \in V$ **do** $B_v \leftarrow \{v\}$; **endfor;**
3. Generate a list L of all edges in E, sorted in non-decreasing order of weight;

   Union Find Data structure
4. **for all** (u,v) in L **do**
5. $\quad B_1 \leftarrow$ FIND(u); $B_2 \leftarrow$ FIND(v);
6. $\quad$ **if** $B_1 \neq B_2$ **then**
7. $\quad\quad A \leftarrow A \cup \{(u,v)\}$; UNION ($B_1$, B_2);
8. $\quad$ **endif;**
9. **endfor;**





Sorting by weight.

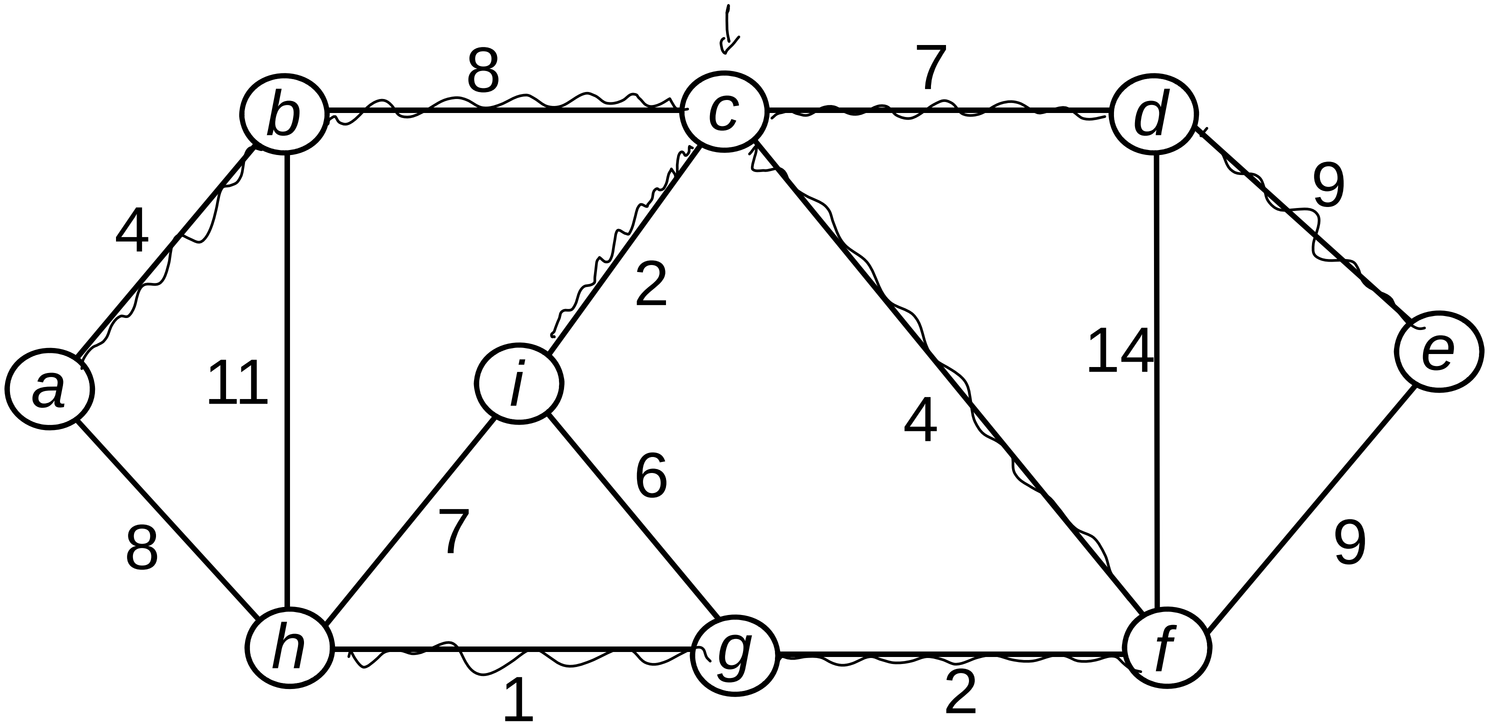
Running time: $O(m\,\alpha(m,n) + m + m \log m)$

Sequence of $2m$ finds and $n-1$ unions, takes time $O(m \cdot \alpha(n,m))$ on an n-element universe

6. Prim's algorithm

A is always a single tree. Start from an arbitrary root vertex *r*. In each step, add a light edge to *A* that connects *A* to a vertex in *V* \ *A*.

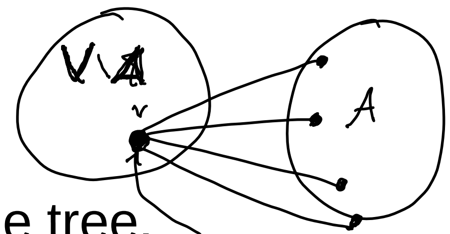
root



Implementation

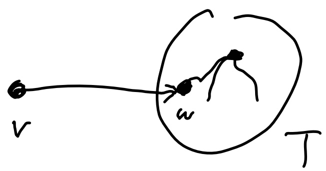
Q : priority queue containing all vertices $v \in V \setminus A$.

key of vertex v : minimum weight of any edge connecting v to a vertex in A (i.e. in the tree)



$\text{key}(v) = \min_{w \in A} w(v, w)$

For a vertex v, let $p[v]$ denote the parent of v in the tree.



$A = \{ (v, p[v]) : v \in V - \{r\} - Q\}$

Prim's algorithm

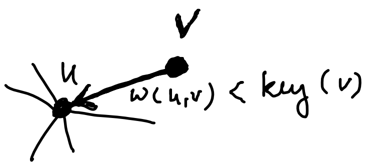
key(v)

```
for all v ∈ V do Insert(Q, ∞, v); endfor;
Choose a root vertex r ∈ V;
DecreaseKey(Q, 0, r); p[r] ← nil;
while ¬Empty(Q) do
    (d, u) ← DeleteMin(Q);
    for all (u,v) ∈ E do
        if v ∈ Q and w(u,v) < key of v then
            DecreaseKey(Q, w(u,v), v);    p[v] ← u;
        endif;
    endfor;
endwhile;
```

root does not have a parent.

key

vertex



n delete min

≤ m decrease key.

Running time: $O(n \log n + m)$ Prim's Alg. $O(m \cdot \alpha(n,m) + m \cdot \log n)$

Kruskal's

Fibonacci heap: O(1) for decrease key

O(log n) for delete min