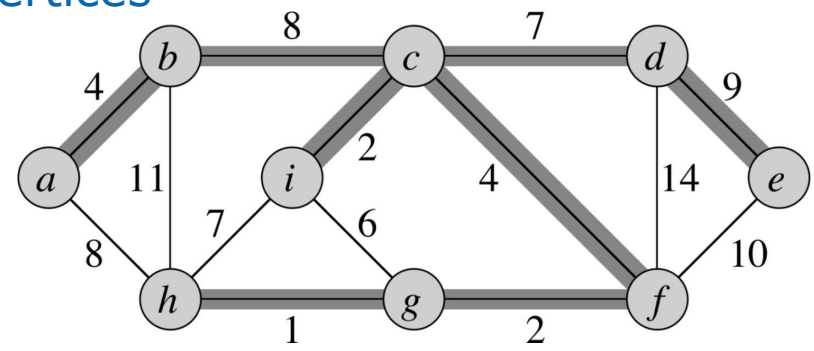
§4 Minimum Spanning Tree

Tree: connected (undirect.) graph without cycles

weighted undir. graph (V,E,w) where $E \subseteq V \times V$ and $w: E \to \mathbb{R}$ s.t. $(u,v) \in E \Rightarrow (v,u) \in E \ \wedge \ w(u,v) = w(v,u) < \infty$

MST (Minimum Spanning Tree) of (V,E,w):

a connected subset F of the edges E which (i) includes all vertices and (ii) has least weight $w(F) = \sum_{(u,v) \in F} w(u,v)$ among all F' satisfying (i).



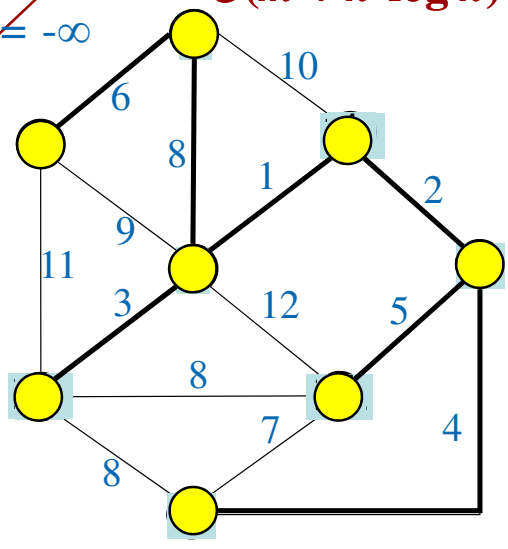
Prim's Algorithm for MST


```
// w : E ⊆ V×V → ℤ  edge weights
T := { }; FOREACH v∈V DO
   v.dist := ∞ ; v.neighbor := NIL; DONE
Q := V; WHILE Q≠{ } DO
  u := Q.ExtractMin(); u.dist := -∞
  IF u.neighbor ≠ NIL THEN
     T := T ∪ { (u.neighbor , u) } ;
  FOREACH v adjacent to u DO
     IF w(u,v)<v.dist THEN
     v.neighbor := u;
     Q.decreaseKey(v,w(u,v));
     END;
```

$n = |V|$ times

$m = |E| \geq n$ times

Fibonacci Heap: $O(m + n \cdot \log n)$



Kruskal's Algorithm for MST

KAIST CS500 M. Ziegler

// $w : E \subseteq V \times V \to \mathbb{Z}$ edge weights

$T := \{\}$; FOREACH $v \in V$ DO **MakeSet**(v); // singletons

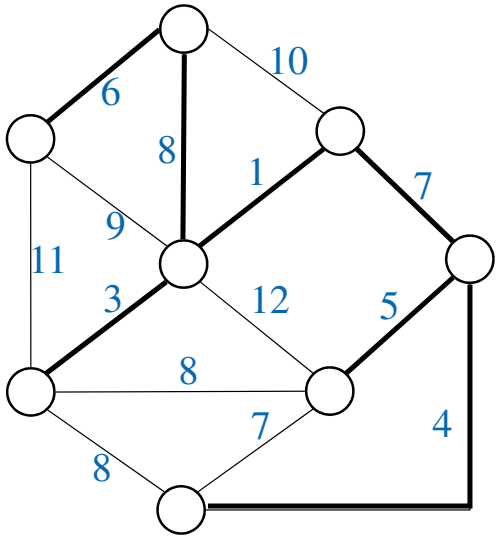
FOREACH edge $(u,v) \in E$ in order of increasing weight DO

IF u and v do NOT belong to the same subset

THEN $T := T \cup \{ (u,v) \}$;

Union(u,v);

END;



Disjoint-Set Data Structure:

$O(m \cdot \alpha(m)) < o(m \cdot \log^*(m))$

Prim's Algorithm with Fibonacci Heap: $O(m + n \cdot \log n)$

§4 Fast&Slowly Growing Functions

KAIST CS500 M. Ziegler

2^n exponential $\quad \lceil \log N \rceil = \min\{ n : 2^n \geq N \}$

2^{2^n} doubly exponential $\quad \lceil \log\log N \rceil = \min\{ n : 2^{2^n} \geq N \}$

...

$2^{2^{\cdot^{\cdot^{2}}}}$ tower of height n

aka tetration $2\uparrow\uparrow n = 2^{2\uparrow\uparrow(n-1)}$

$\log^*(N)$ = #iterations of log before argument ≤ 1

$= 1 + \log^*(\log N), \; N>1$

Example: $\log(2^{64})$=? $\quad \log\log(2^{128})$=? $\quad \log^*(2^{512})$=?

Wilhelm Ackermann (1896—1962) function

$A_0(n)=n+2, \quad A_{k+1}(0)=A_k(1), \quad A_{k+1}(n+1) = A_k(A_{k+1}(n))$

$A_1(n) = 2n+3, \quad A_2(n) = 2^{n+3}-3, \quad A_3(n) = 2\uparrow\uparrow(n+3)-3$

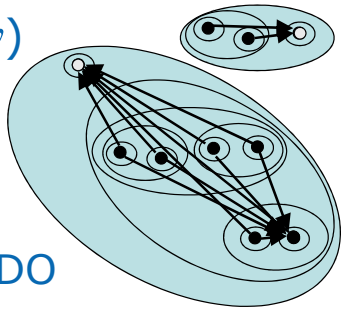
Inverse Ackermann $\alpha(N) = \min\{ n : A_n(n) \geq N \}$

§4 Disjoint-Set Data Structure

KAIST CS500 M. Ziegler

MakeSet(x) **FindSet**(x) **Union**(x,y)

return "handle" **Goal:** amortized $O^*(1)$



Recall Kruskal MST of $G=(V,E,w)$

FOREACH $(u,v)\in E$ in increasing weight w DO

IF **FindSet**$(u)\neq$**FindSet**(v)

THEN $T := T \cup \{ (u,v) \}$; **Union**(u,v) ;

Naïve implementation as forest of depth 1: n calls, $m:=n/2$ of which are **MakeSet** $\rightarrow$ total time $\Omega(n^2)$

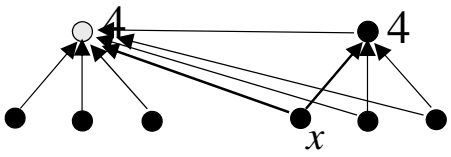
Weighted union heuristic: attach smaller to larger tree

Theorem: This yields total time $O(n+m\cdot\log m)$ for any sequence of n calls, m of which are **MakeSet**

§4 Analysis of Union-by-Weight

KAIST CS500 M. Ziegler

MakeSet(x) **FindSet**(x) **Union**(x,y)



Observation: An element's link is updated only when its set is combined with one of larger or equal size.

So update $\#k$ can occur only after $m\geq 2^k$ calls **MakeSet**.

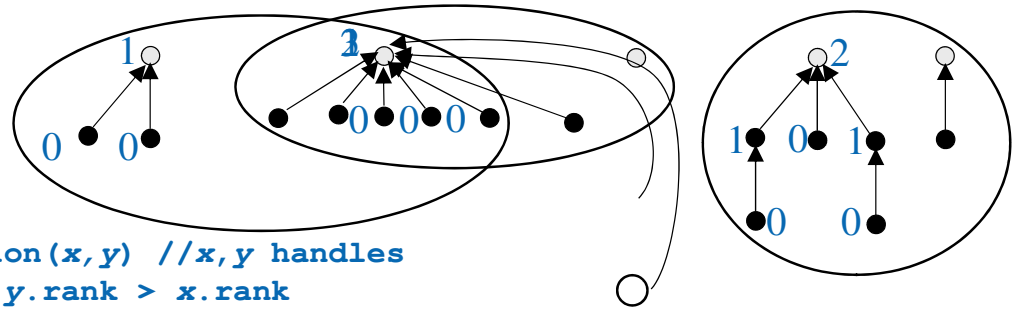
MakeSet, **FindSet**: $O(1)$, **Union** ?

Weighted union heuristic: attach smaller to larger tree

Theorem: This yields total time $O(n+m\cdot\log m)$ for any sequence of n calls, m of which are **MakeSet** ■

Lazy Union by Rank, Path Compression

KAIST CS500 M. Ziegler



```
Union(x,y) //x,y handles
if y.rank > x.rank
 then attach x to y
else attach y to x;
 if x.rank == y.rank
  then x.rank++; fi; fi;
```

```
function FindSet(x): if x.parent≠NIL
 then x.parent := FindSet(x.parent);
return(x.parent);
```

~~Naïve~~ implementation as forest of unbounded depth:

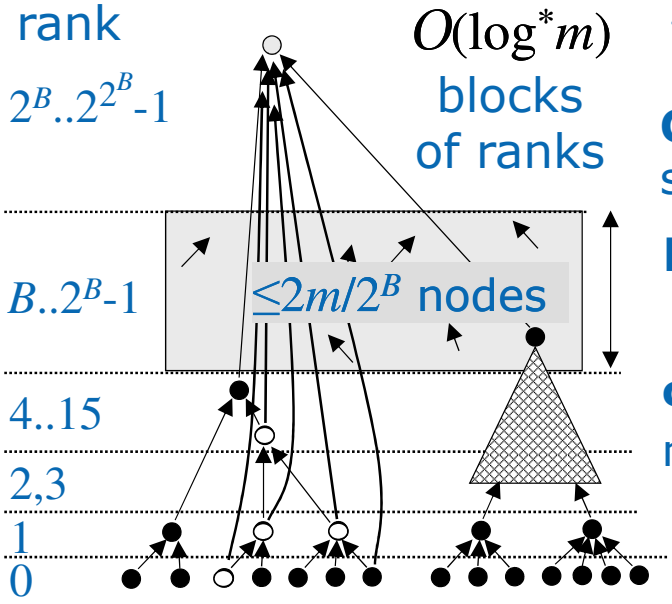
MakeSet, Union: $O(1)$, **FindSet** ? *Path compression*

Lazy Union-by-rank: attach lower rank tree to higher

Theorem: This algorithm makes m **MakeSet** and n-m **FindSet** and **Union** calls run in total time $O(n \cdot \alpha(m))$.

Amortized Analysis of Union-Find

KAIST CS500 M. Ziegler



Union(x,y) **FindSet**(x)

Claim a) Ranks increase strictly along each path.

b) Any node of rank r is *ancester* to $\geq 2^r$ nodes.

c) No more than $m/2^r$ nodes can have rank$\geq r$

Total cost ≤ #pairs of *related* vertices

= #pairs *transcending* block/s + #pairs *within* a block

Theorem: This algorithm makes m **MakeSet** and n-m **FindSet** and **Union** calls run in total time $O(n \cdot \log^* m)$.