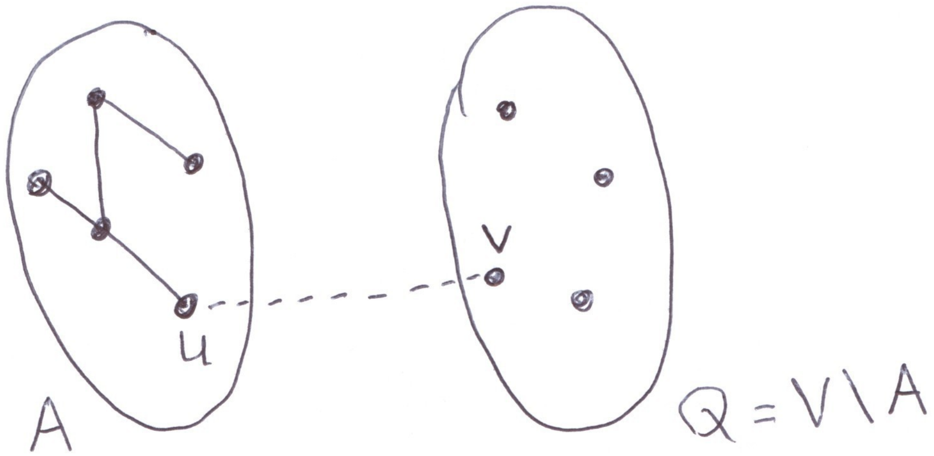
Prim (1957) [Jarník (1930), Dijkstra (1959)]

Start: A = set consisting of one (arbitrary) vertex of V

T = empty edge set

One iteration:



* take edge $\{u,v\}$ of minimum weight such that $u \in A$, $v \in Q$.
* add the edge $\{u,v\}$ to T.
* move v from Q to A.

Repeat until $A = V$ (i.e., $Q = \emptyset$).

Prim:

r = arbitrary vertex of V;

$A = \{r\}$;

$T = \phi$;

while $A \neq V$:

  find edge $\{u,v\}$ of minimum weight such that

  $u \in A$, $v \in V \setminus A$;

  $A = A \cup \{v\}$;

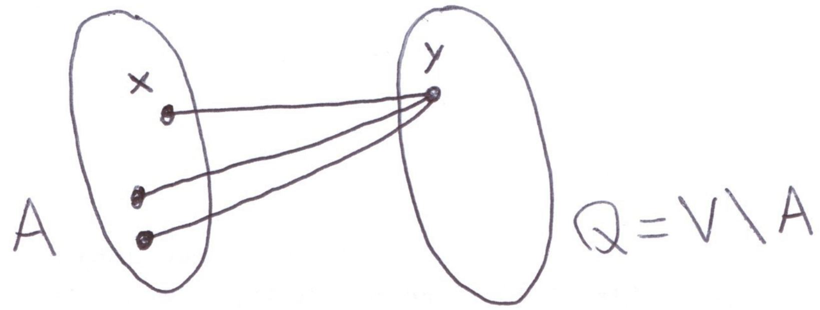
  $T = T \cup \{\{u,v\}\}$

How to find the edge $\{u,v\}$: by brute force in $O(m)$ time.

Total running time $= O(mn)$,

where $n = |V|$, $m = |E|$.

To improve the running time: maintain extra information.



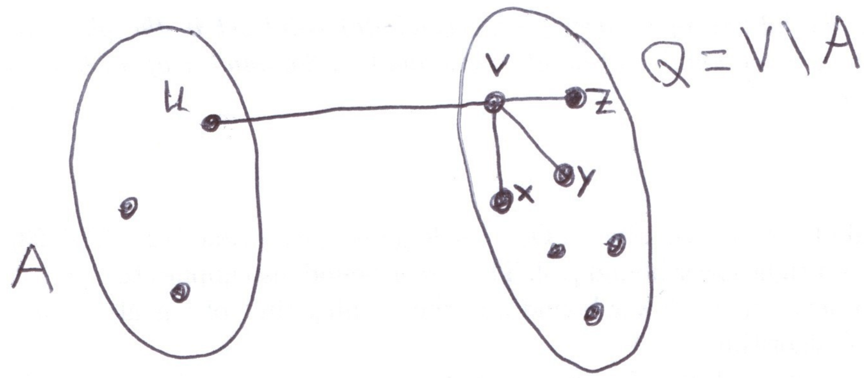
For each vertex y in Q:

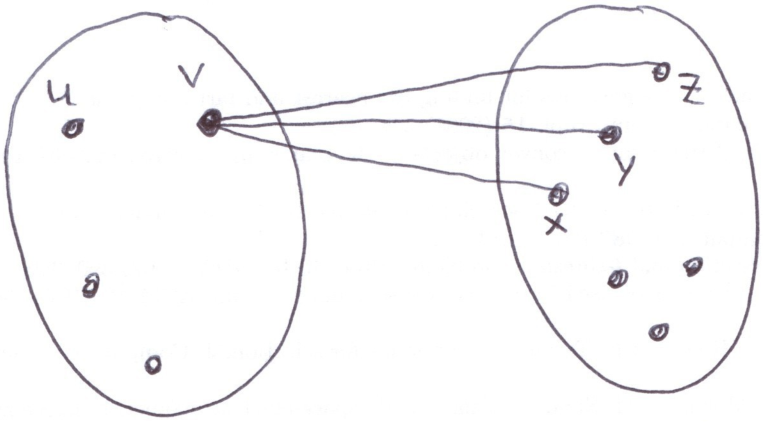
minweight(y) = minimum weight of any edge between y and a vertex of A.

closest(y) = vertex x in A for which $wt(x,y)$ = minweight(y).

<u>Observe</u>: Shortest edge $\{u,v\}$ connecting A and Q has weight $\min\{\text{minweight}(y) : y \in Q\}$.

What happens if we move v from Q to A:





update minweight(w) and closest(w) for w = x, y, z.

Prim:

```
r = arbitrary vertex of V;
A = {r};
T = ϕ;
for each vertex y ≠ r: minweight(y) = ∞; closest(y) = nil;

for each edge {r,y}: minweight(y) = wt(r,y);
                     closest(y) = r;

Q = V \ {r}; k = 1;
while k ≠ n:    // k = |A|
    v = vertex of Q for which minweight(v) is minimum;
    u = closest(v);
    A = A ∪ {v}; Q = Q \ {v}; T = T ∪ {{u,v}};
    k = k+1;
    for each edge {v,y}:
        if y ∈ Q and wt(v,y) < minweight(y):
            minweight(y) = wt(v,y);
            closest(y) = v;
```

Store the vertices of Q in a min-heap, for each v, the key of v is minweight(v).

Store T in a list.

With each vertex of V: store one bit indicating whether the vertex belongs to A or to Q.

Running time:

Up to the while-loop: $O(n)$ (this includes the time to build the heap)

One iteration of the while-loop:

extract_min : $O(\log n)$ time

$\leq$ degree(v) many decrease_key operations:

$O(\text{degree}(v) \cdot \log n)$ time.

Total time for the while-loop:

$$O\Big(\underbrace{\sum_{v \in V} \text{degree}(v)}_{=2m} \cdot \log n\Big) = O(m \log n).$$

<u>Conclusion</u>: Prim's algorithm computes MST in $O(m \log n)$ time.