Greedy Algorithms

- greedy algorithms work by making an irrevocable choice at each stage of the computation. then recurse on the smaller subproblem, and finally combine the sub-result with our choice.
- In order for a greedy approach to work the problem must exhibit two properties
  - optimal substructure: optimal solutions to sub-problems can be combined to produce an overall optimal solution.

shortest path

shortest path between 2 nodes

path from $u \to w$ is the shortest path

longest path

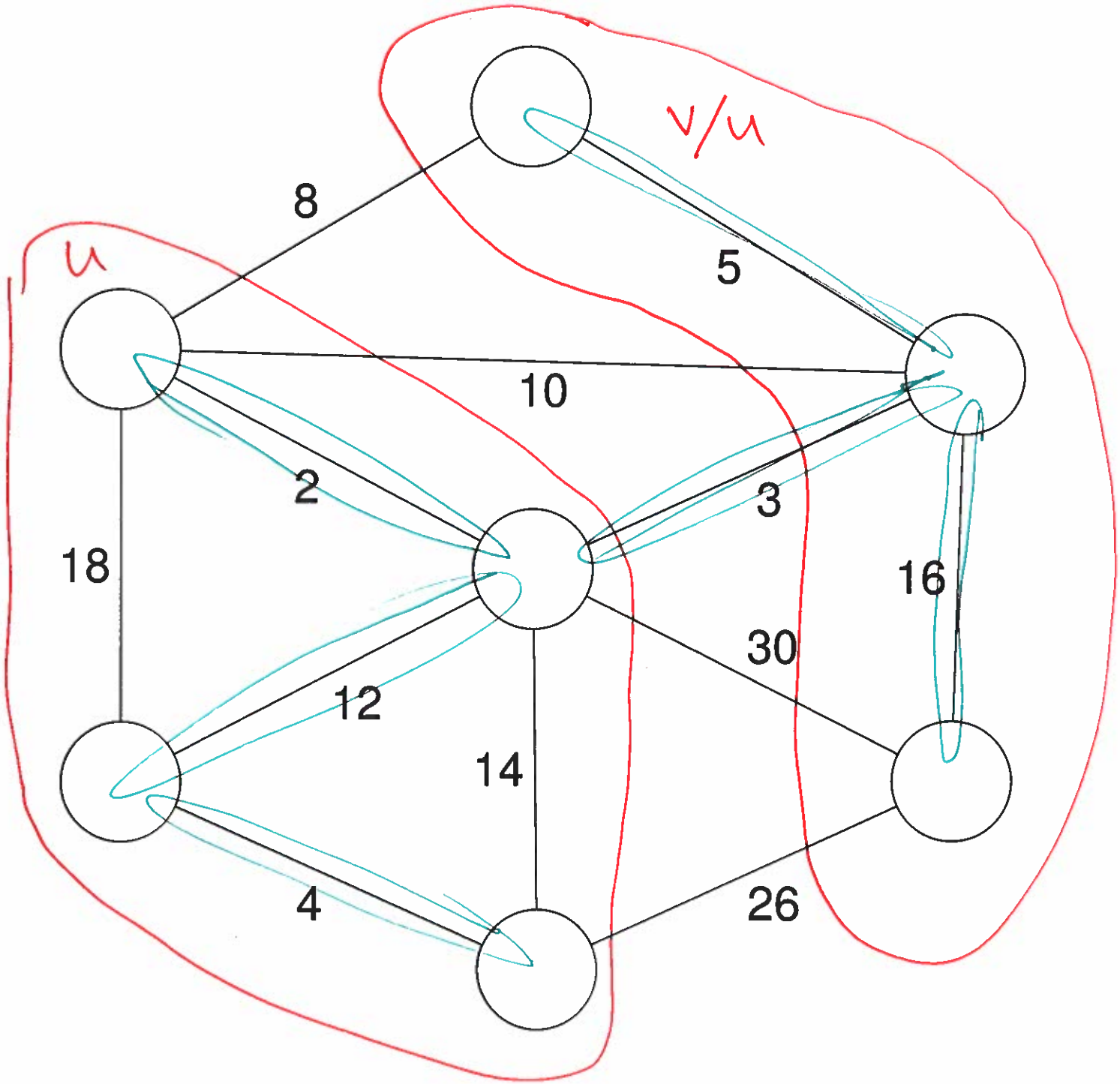
longest simple path between two nodes

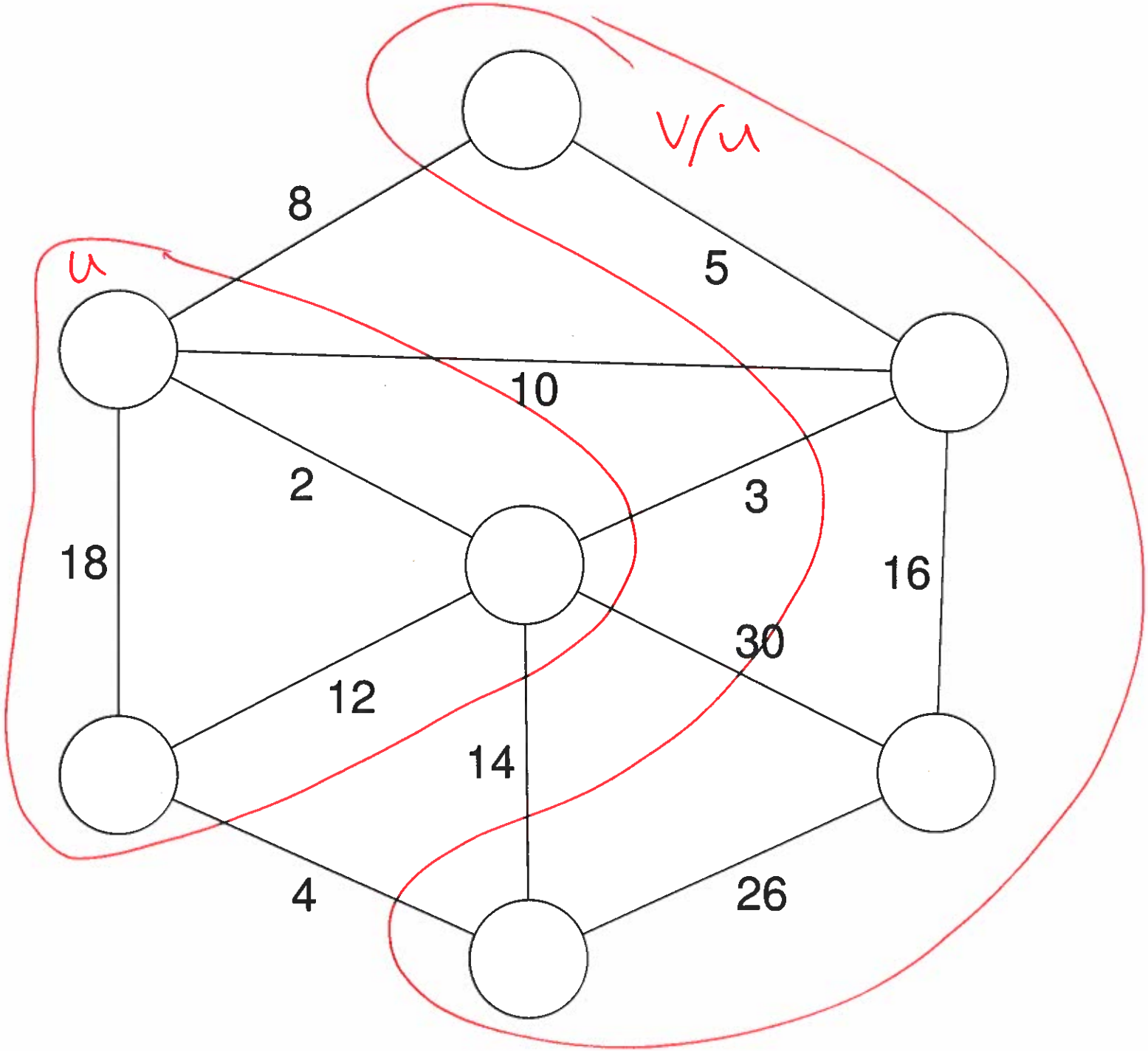
- Greedy Choice Property

  There must be a way to select a next step that generates an optimal overall solution. This choice can depend on previous choices, but not on future choices.

Minimum Spanning Trees

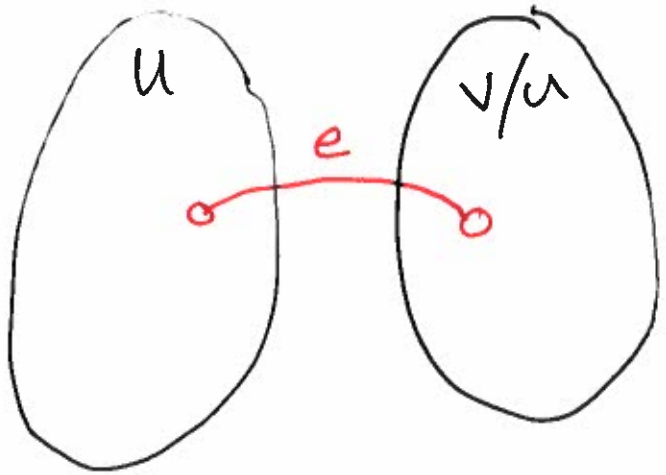
- A spanning tree for $G = (V, E)$ is a tree $T = (V, E')$ where $E' \subseteq E$
- The weight of a tree T is defined as $\sum_{e \in E'} w(e)$
- A MST is a Spanning Tree of minimum weight


V/U
U
8
5
10
2
3
18
16
30
12
14
4
26


V/U
U
8
5
10
2
3
18
16
30
12
14
4
26

- any non-empty proper subset U of vertices V of G defines a cut of G into two sets U and V/U

- a cut edge is any edge having one endpoint in U and the other in V/U



e crosses the cut

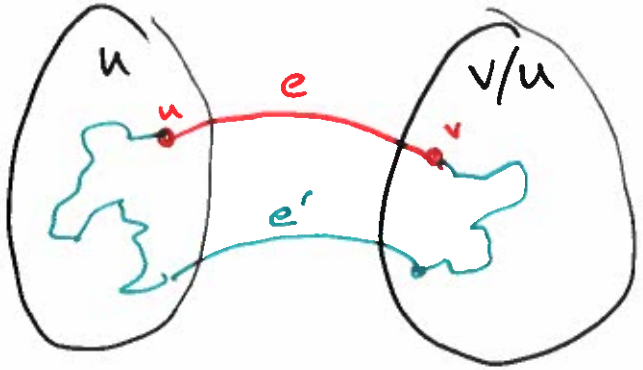
Cut Lemma

For any cut defined by a subset U of the vertices of G, the MST of G contains the minimum weight edge e that crosses the cut.

Proof

- consider any cut U, V/U of G
- let $e = (u, v)$ where $u \in U$ and $v \in V/U$ be the least weight edge crossing the cut
- Assume there exists a T that does not contain e. We will show that T is not the MST

- Since T spans G there is a path from u to v not involving e
- Because $u \in U$ and v is not there must be at least one edge e' on the path from u to v that crosses the cut.
- Removing e' from T splits T into two trees T_1 and T_2 where $u \in T_1$ and $v \in T_2$
- Adding e to these two trees gives us a new spanning tree $T' = T - e' + e$



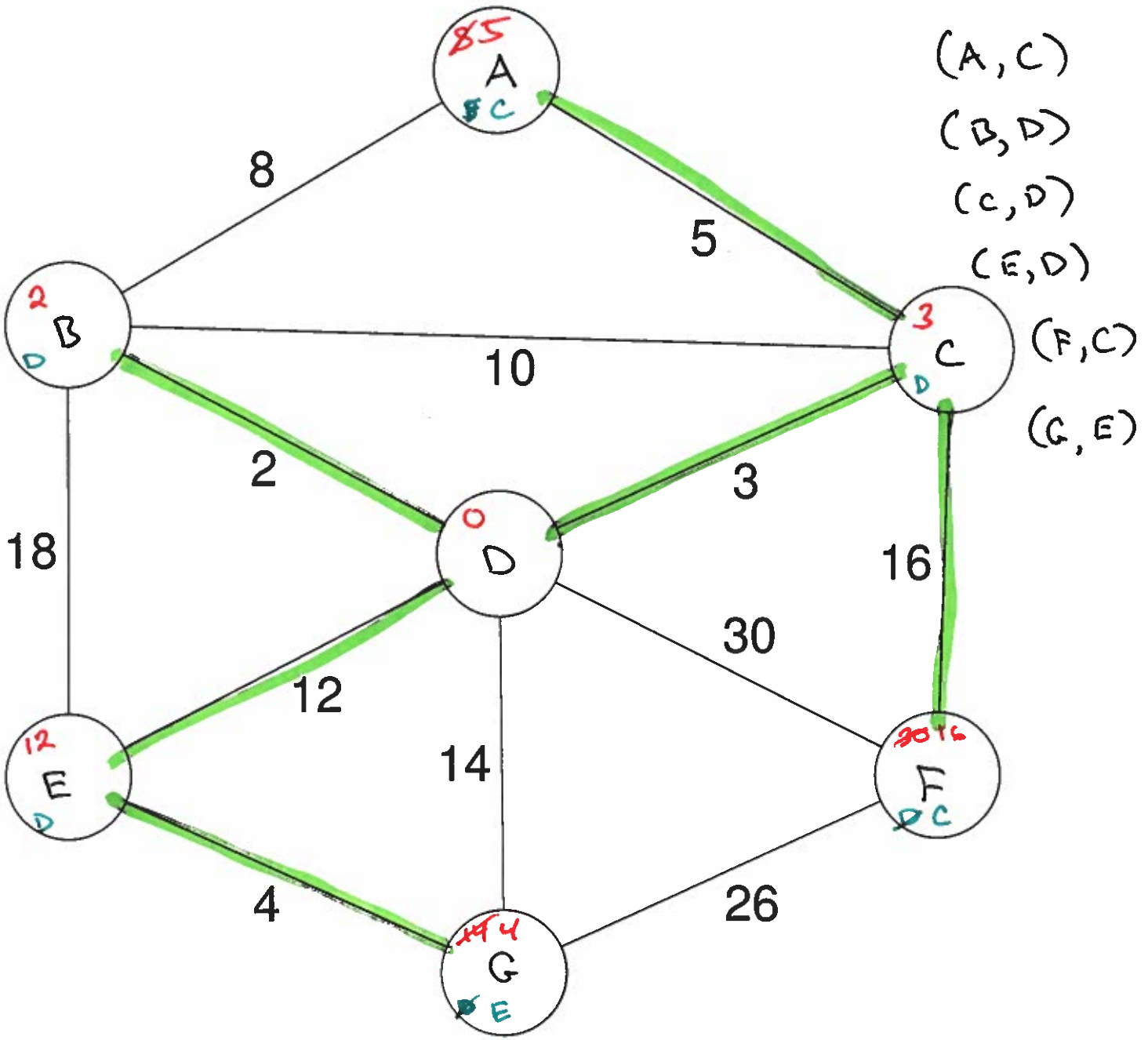
- Since e is the lightest weight edge crossing the cut $w(e) < w(e')$ so $w(T') < w(T)$
- Therefore T is not the MST for G

Prim's Algorithm

- Make an initial cut of size 1
- on each iteration expand the current tree by connecting the nearest vertex not in the tree
- maintain a Min priority Queue on V/U

$$v \in V/U$$

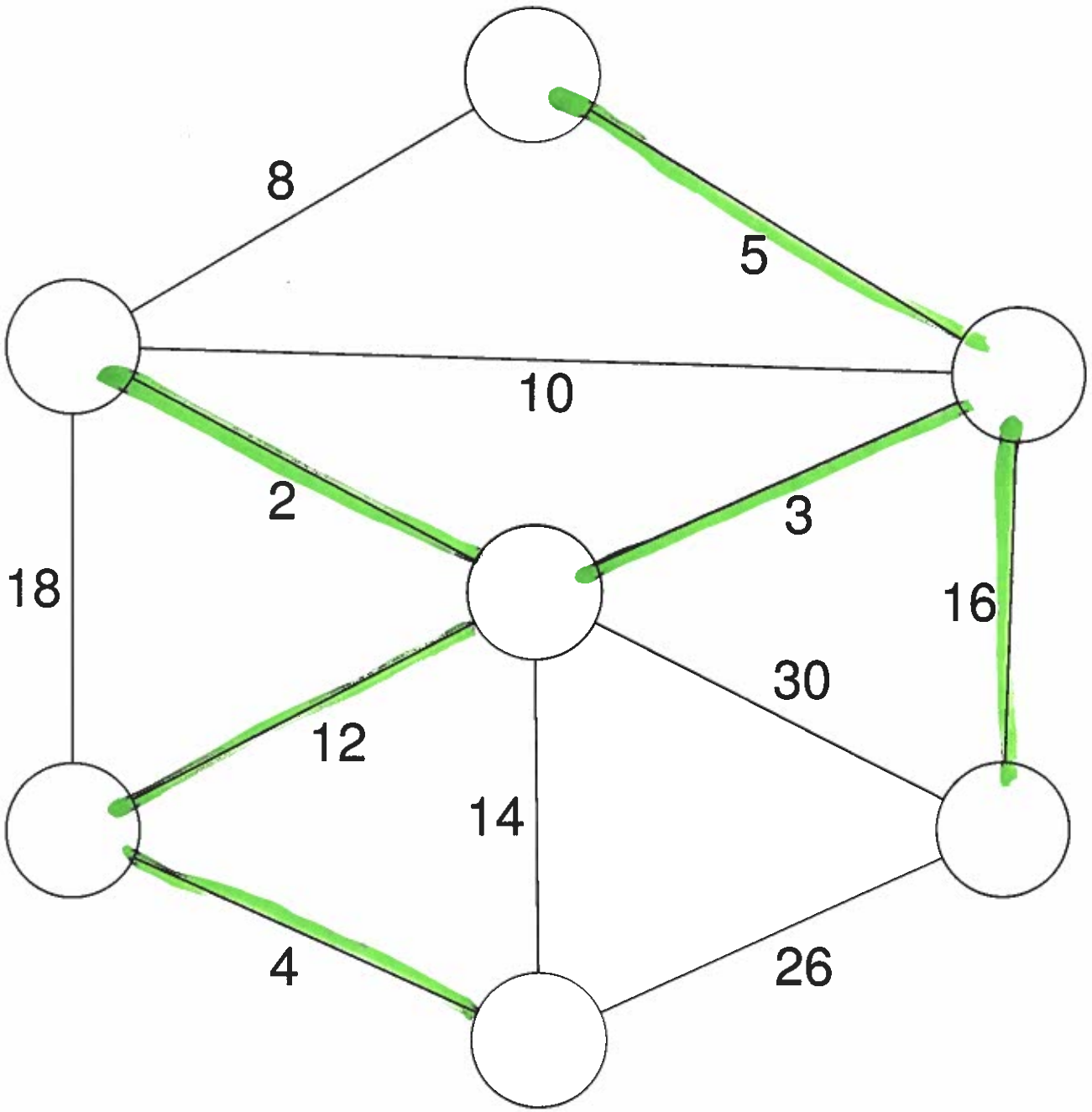
$$key(v) = \min\{w(u,v) \mid u \in U\}$$

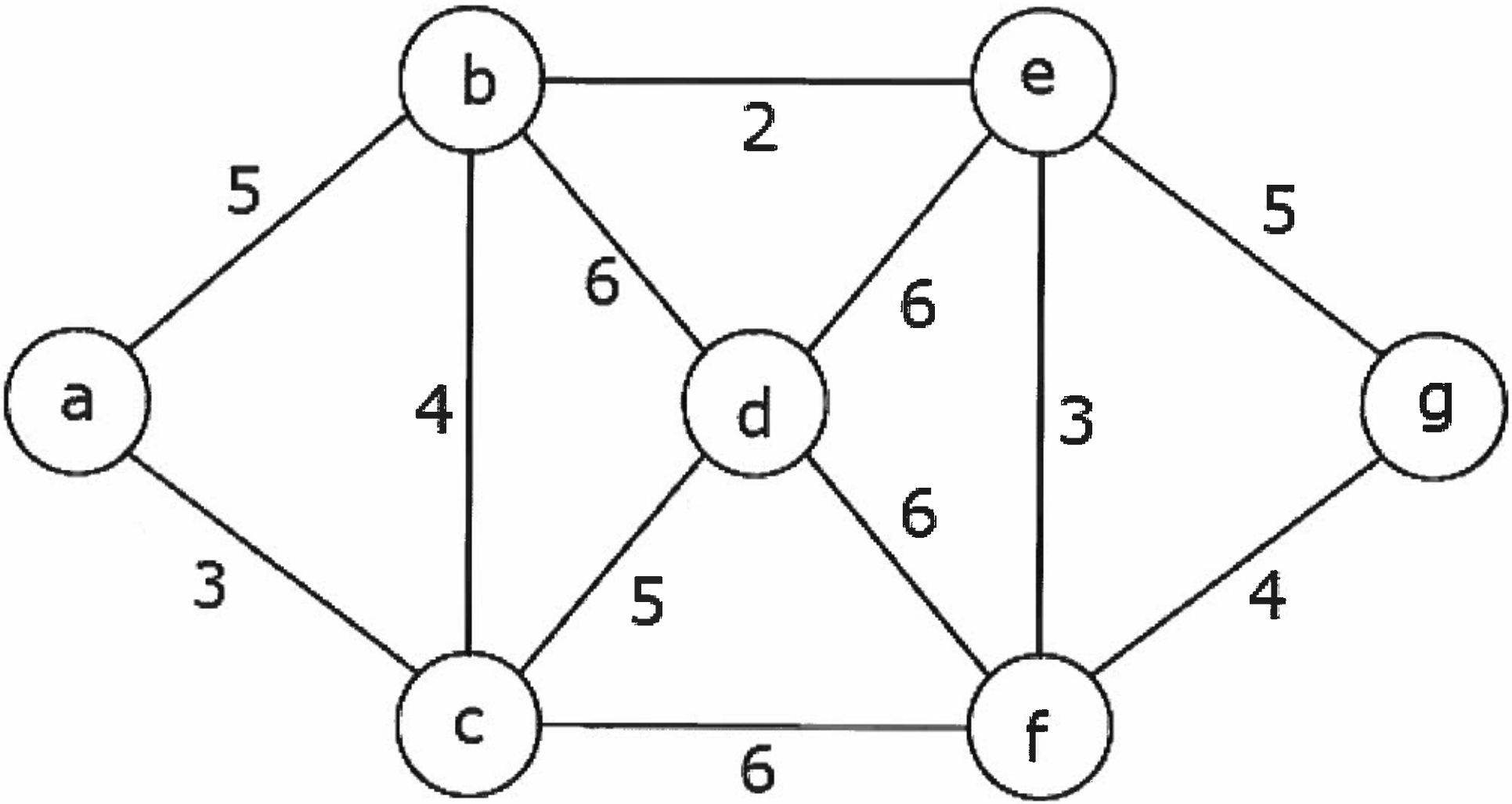

A
B
C
D
E
F
G
8
5
10
2
3
18
16
30
12
14
4
26
(A, C)
(B, D)
(C, D)
(E, D)
(F, C)
(G, E)

PRIM'S ALGORITHM(V, E, r)

```
1  for each u ∈ V
2        key[u] = inf
3        π[u] = NIL
4  key[r] = 0
5  Q = V
6  while Q is not empty
7        u = EXTRACTMIN(Q)                          ← Θ(log |V|)
8        for each v ∈ Adj[u]
9              if v ∈ Q and w(u,v) < key[v]
10                   DECREASEKEY(Q, key[v], w(u,v))  ← Θ(log |V|)
11                   π[v] = u
12 return {(v, π[v]) : ∀v ∈ V − {r}}
```

$\Theta(|V| \log |V|) + \Theta(|E| \log |V|) + \Theta(|V|)$


8
5
10
2
3
18
16
30
12
14
4
26


b
e
a
d
g
c
f
5
2
6
6
5
4
3
6
3
5
4
6