

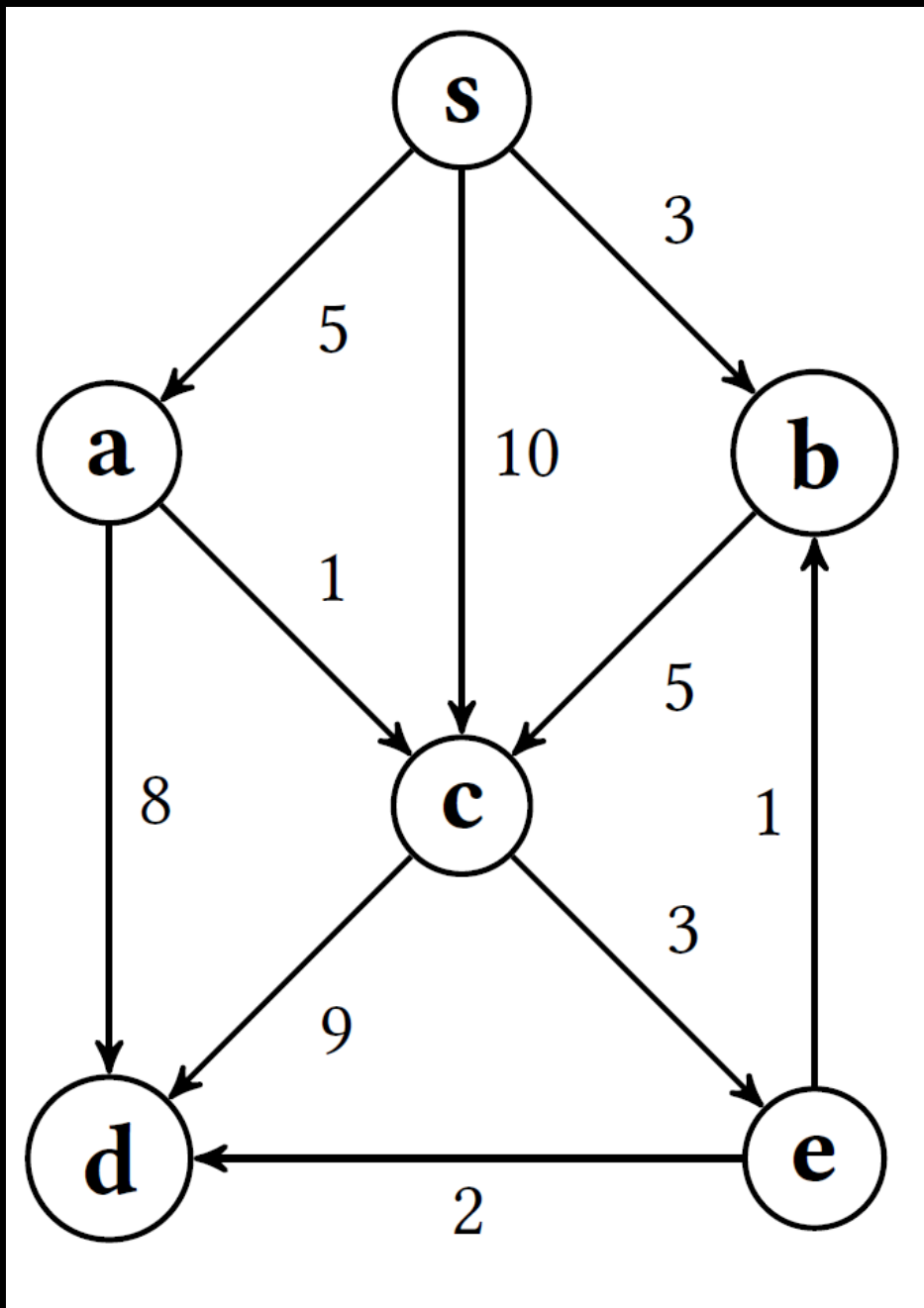
ALGORITHMS & DATA STRUCTURES

December 13th

TODAY'S AGENDA

- Bonus Exercises
- Short remark: Kruskal + Prim
- Tips and Tricks for Graph Problems

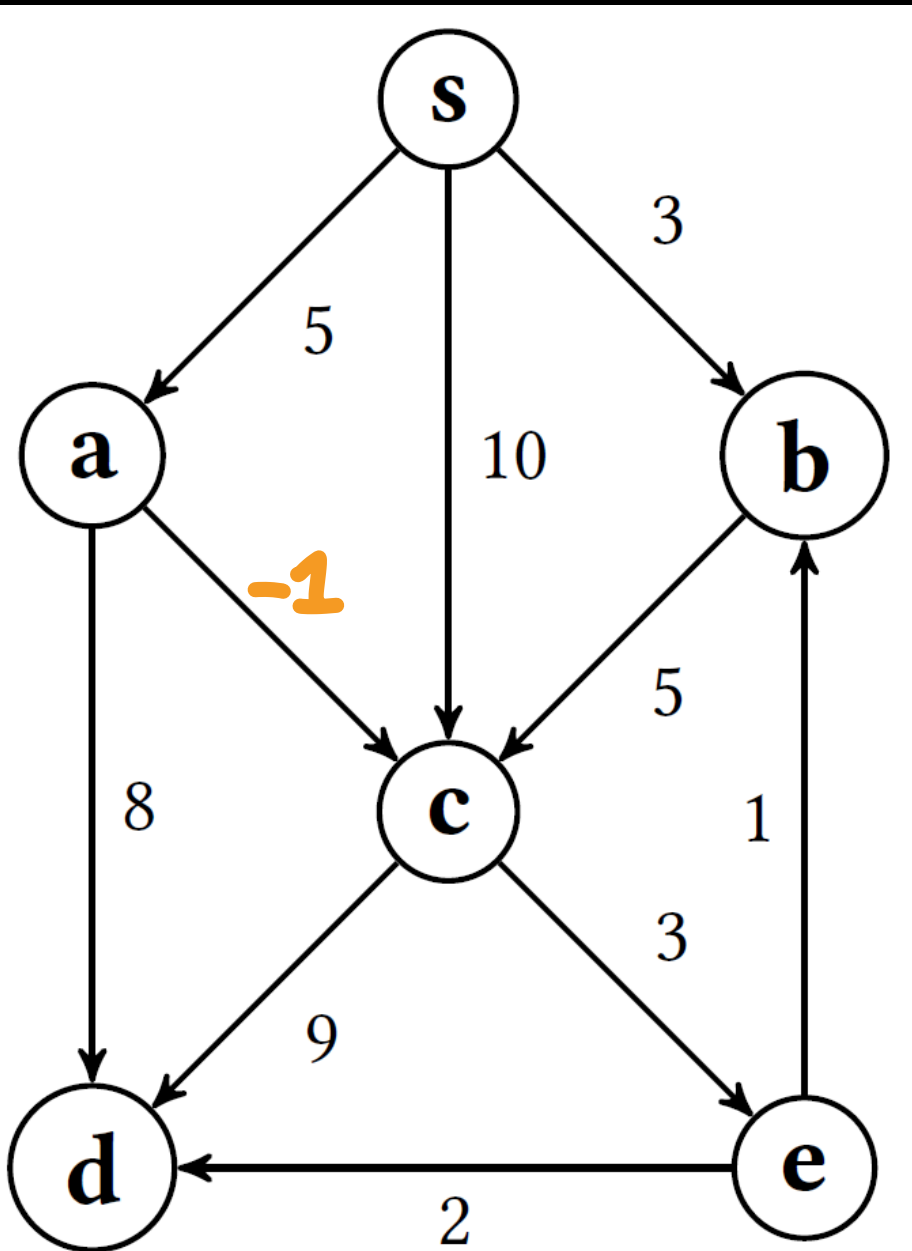
EXERCISE 11.1



S	d_s	d_a	d_b	d_c	d_d	d_e
\emptyset	0	∞	∞	∞	∞	∞
{s}	0	5	3	10	∞	∞
{s,b}	0	5	3	8	∞	∞
{s,b,a}	0	5	3	6	13	8
{s,b,a,c}	0	5	3	6	13	9
{s,b,a,c,e}	0	5	3	6	11	9
{s,b,a,c,e,d}	0	5	3	6	11	9

p_s	p_a	p_b	p_c	p_d	p_e
\emptyset	s	s	a	e	c

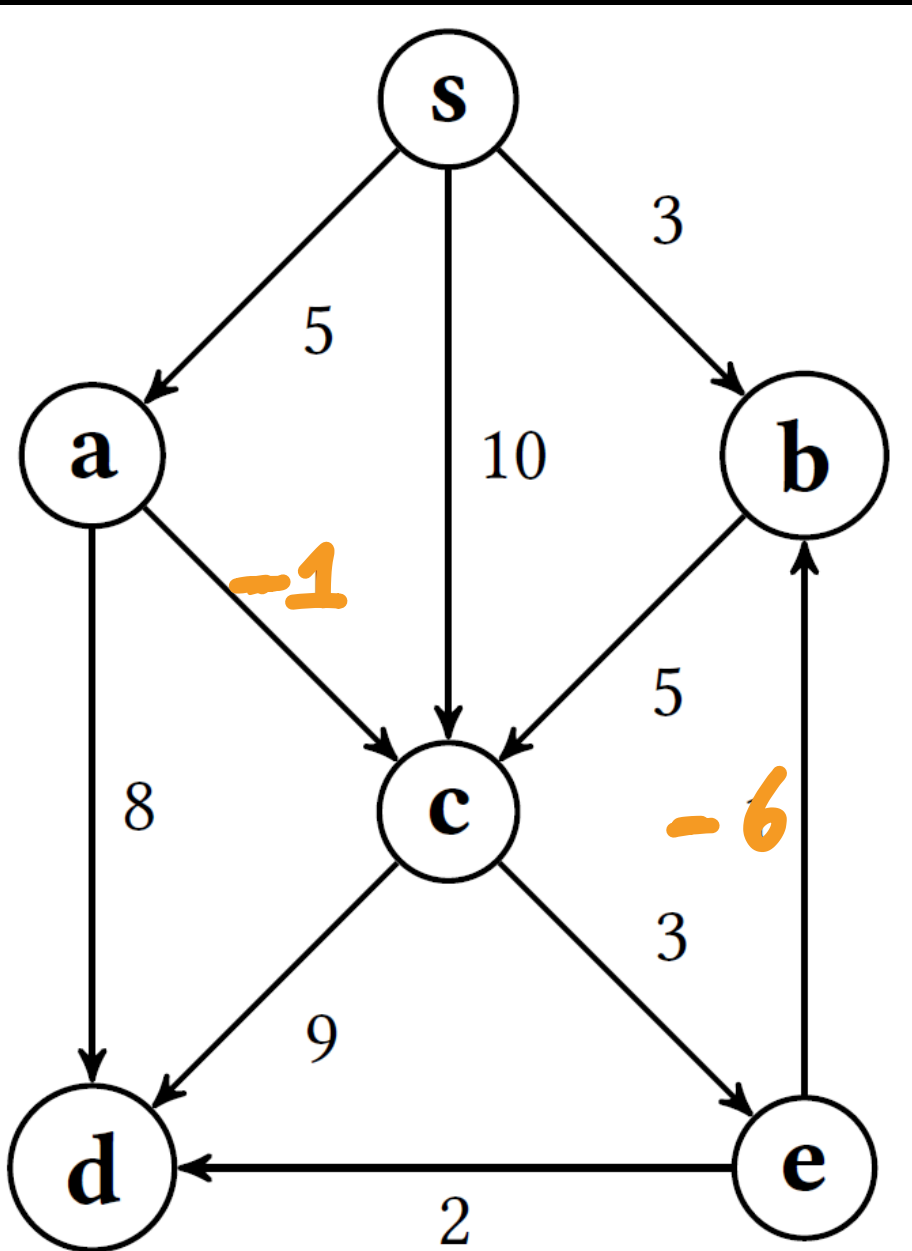
.



S	d_s	d_a	d_b	d_c	d_d	d_e	M
\emptyset	0	∞	∞	∞	∞	∞	{s}
{s}	0	3	5	∞	∞	∞	{s,a,b}
{s,a}	0	3	5	8	∞	∞	{s,a,b}
{s,a,b}	0	3	5	4	13	∞	{s,a,b,c,d}
{s,a,b,c}	0	3	5	4	13	7	✓
{s,a,b,c,e}	0	3	5	4	9	7	✓
{s,a,b,c,d,e}	0	3	5	4	9	7	✓

p_s	p_a	p_b	p_c	p_d	p_e
\emptyset	s	s	a	e	c





	s	d _s	d _a	d _b	d _c	d _d	d _e
∅	0	∞	∞	∞	∞	∞	∞
{s}	0	5	3	∞	∞	∞	∞
{s, b }	0	5	3	8	∞	∞	∞
{s, b, a}	0	5	3	4	13	∞	∞
{s, b, a, c}	0	5	3	4	13	7	7
{s, b, a, c, e}	0	5	1	4	9	7	7
{s, b, a, c, e, a}	0	5	1	4	9	7	7

EXERCISE 11.3

Consider a weighted directed graph $G = (V, E)$ that is given in the adjacency-list representation. The graph G has n vertices and m edges. The weights $w : E \rightarrow \mathbb{R}$ of the edges are not necessarily positive, but the graph does not contain any negative weight cycle. Every vertex is colored either red or green, so that the vertex set is partitioned as $V = V_{red} \cup V_{green}$. A path is said to be *allowable* if it contains at most one red vertex.

We are given a source $s \in V$ and would like to find the weights of the *shortest allowable paths* from s to all vertices in V . That is, we are interested in $(\delta_a(s, v))_{v \in V}$, where $\delta_a(s, v)$ is the weight of the shortest allowable path from s to v . If v is not reachable from s by an allowable path, then $\delta_a(s, v) = \infty$.

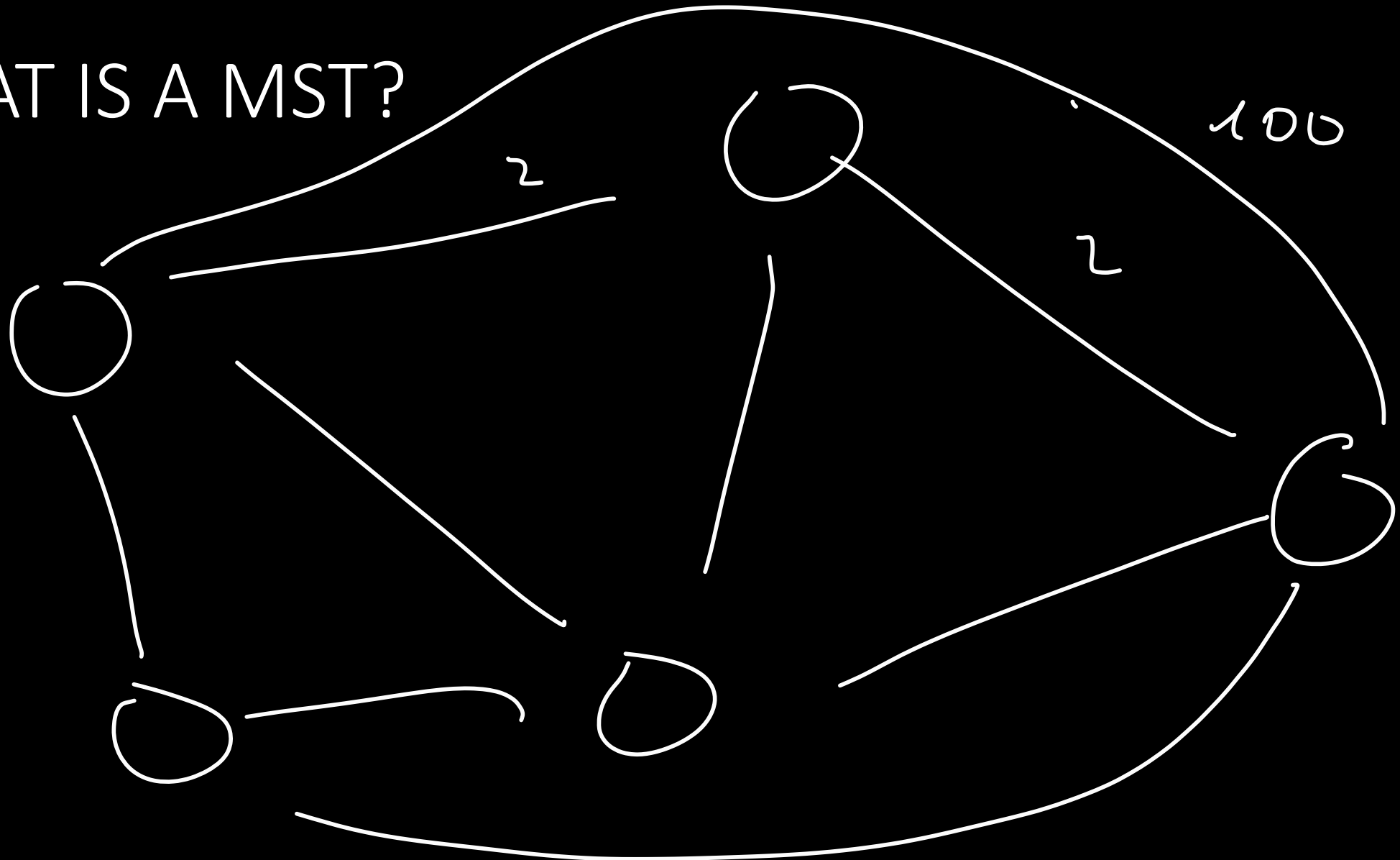
Describe an algorithm that can efficiently compute $(\delta_a(s, v))_{v \in V}$. In order to get full points, the runtime of your algorithm should be $O(mn)$.

Hint: Construct a weighted directed graph G' with $2n$ vertices and apply an algorithm that you learned in class on it.

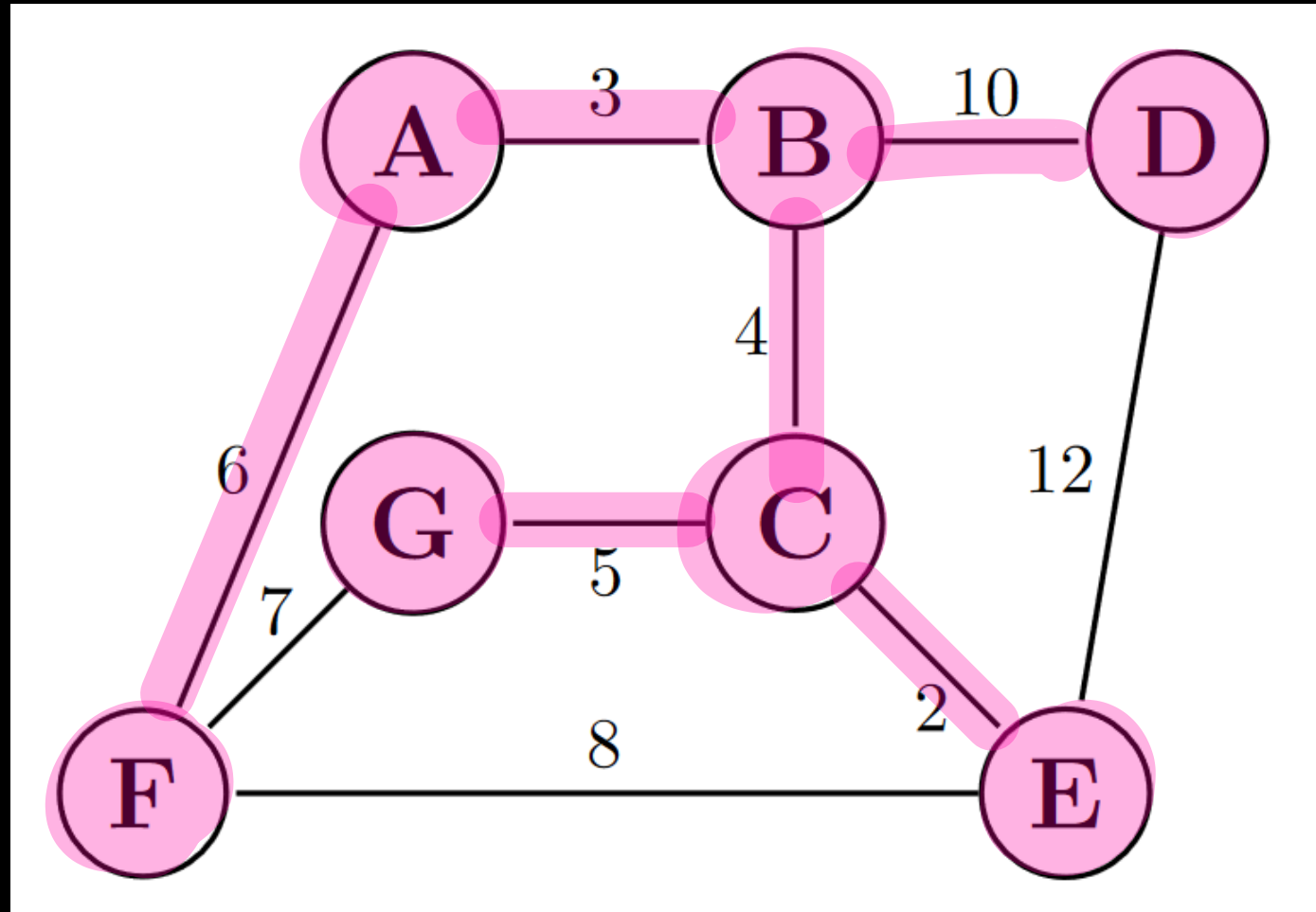
GOLDEN RULE: BUILD A NEW GRAPH AND USE A
SIMPLE ALGORITHM TO SOLVE THE PROBLEM

MST: KRUSKAL + PRIM

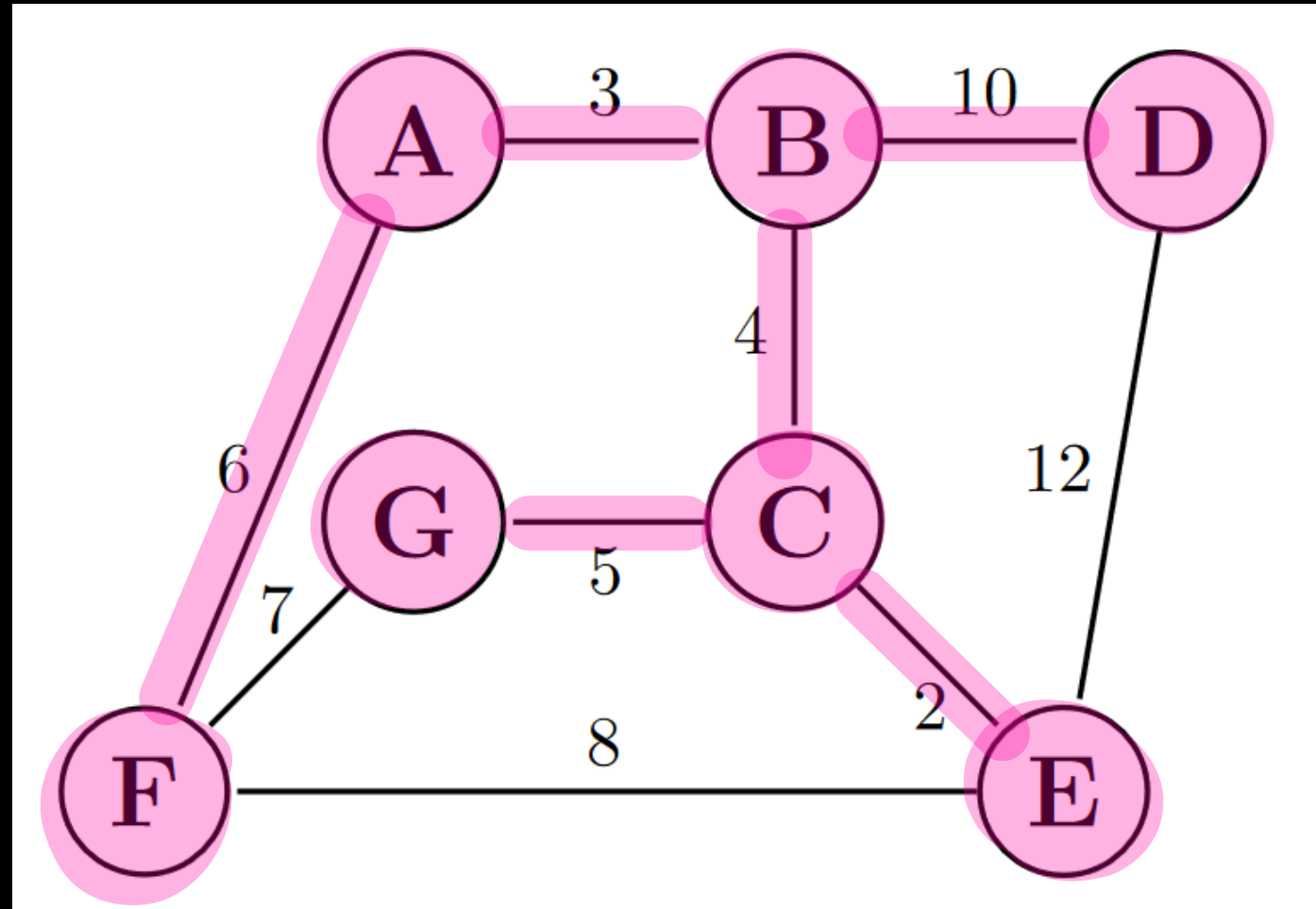
WHAT IS A MST?



KRUSKAL \Rightarrow Union-Find (langsam: compression)



PRIM



GRAPH THEORY: TIPS AND TRICKS


IS EDGE $e=(u,v)$ ON AN (ARBITRARY)
SHORTEST PATH FROM S TO T?

$$d = \text{SP}(s)$$

$$d' = \text{SP}(t)$$

$\{u,v\}$ ist auf einem "beliebigen" KP

\Leftrightarrow

 $d_u + w(\{u,v\}) + d'_v = d_t$



LONGEST PATH IN DIRECTED ACYCLIC GRAPH

NP-complete

$$d[s, t] = \min_{\{u, t\} \in E} \left[d[s, u] + w(u, t) \right]$$

DA $G \Rightarrow$ topologische Sortierung

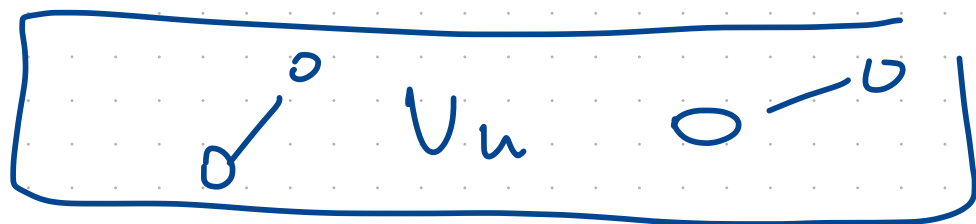
\Rightarrow longest path kann gelöst werden, wenn wir
benutzen \max (statt \min)

LABEL ON SUBSET OF EDGES. SHORTEST PATH FROM S TO T, USING AT LEAST K LABELED EDGES

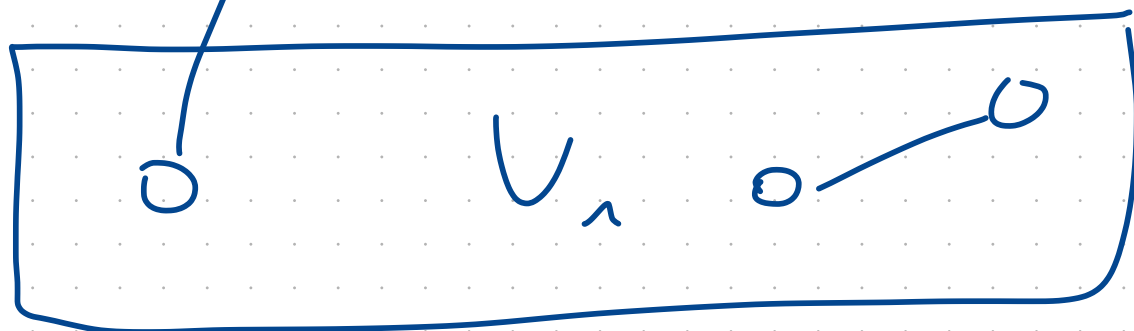
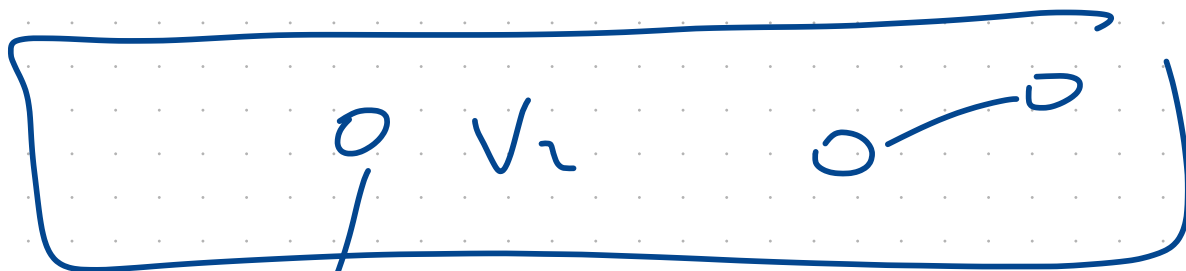
Input The first line of the input contains the number $t \leq 100$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains five integers $n \ m \ k \ x \ y$, separated by a space and such that $1 \leq n \leq 10^4$, $0 \leq m \leq 10^5$, $k \in \{1, \dots, 10\}$, and $x, y \in \{0, \dots, n-1\}$. Here n denotes the number of cities, m denotes the number of roads between cities, k denotes the minimum number of rivers Moran wants to travel along, x denotes the city that is the starting point, and y denotes the city that is the final destination of the journey.
- The following m lines define the roads. Each road is defined by four integers $a \ b \ c \ d$, separated by a space and such that $a, b \in \{0, \dots, n-1\}$, $1 \leq c \leq 2^{10}$, and $d \in \{0, 1\}$. The corresponding road connects cities a and b within a travel time of c minutes. There are no roads that arrive at the same town they depart from (i.e. $a \neq b$). The last parameter d indicates whether ($d = 1$) or not ($d = 0$) the road goes along a river.

Output For each test case output a line that contains the travel time (in minutes) of a shortest journey between the cities x and y such that at least k legs (roads) of the journey go along a river. You may assume that there is a reachable river road and that y is reachable from x . Using a river road multiple times counts multiple times towards the target k .



⋮



```

class Node<T>{
    T item;
    Node parent;

    public Node(T item) {
        this.item = item;
    }
}

public class UnionFind<T> {

    public void makeSet(Node n) {
        n.parent = n;
    }

    public Node find(Node a) {
        if(a.parent.equals(a))
            return a;
        else {
            Node tmp = find(a.parent);
            a.parent = tmp;
            return tmp;
        }
    }

    public void union(Node a, Node b) {
        Node t1 = find(a);
        Node t2 = find(b);

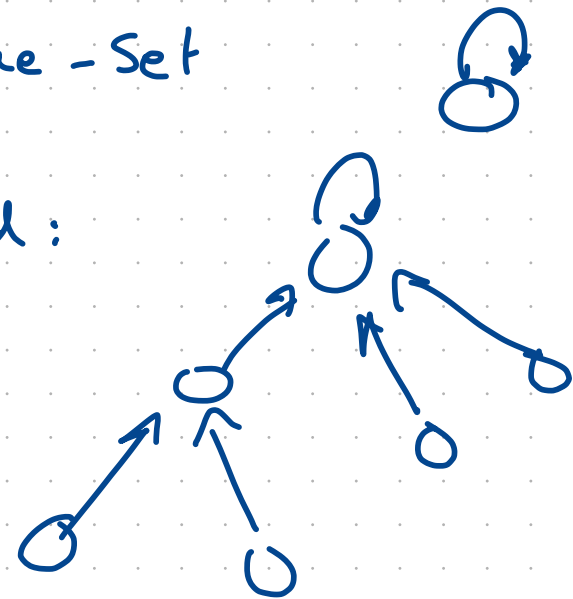
        a.parent = b;
    }

    public static void main(String[] args) {
        UnionFind<Integer> U = new UnionFind<Integer>();
        Node a = new Node("a");
        Node b = new Node("b");

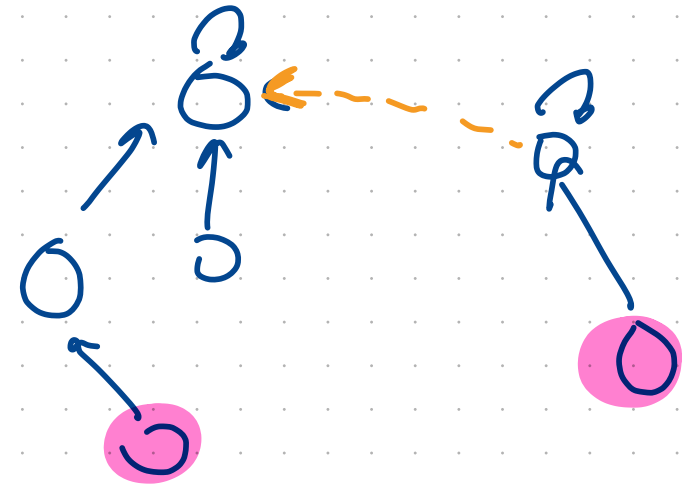
```

Make-Set

Find:



Union :



⇒ Path-Compression