Data Structure:

1. **struct Edge**: stores the edge(u,v,w) and in_set.
   a) **int u,v** :the vertices constructing the edge
   b) **int w**: the edge's weight.
   c) **bool in_set**: the Boolean telling us whether it is in the output edges (removed edges)
2. **char mode**: directed or undirected graph
3. **int total_num_vertices:** the given graph's number of vertices
4. **int total_num_edges:** the given graph's number of edges
5. i**nt removed_total_num**: how many edges we have removed
6. **int new_removed_total_num**: how many edges we have removed for diected graph
7. **int final_weight**: the total weight of edges we have removed
8. **int new_final_weight**: the total weight of edges we have removed for diected graph
9. **int acyclic_total_num:** the total number of edges in acyclic_graph
10. **bool add:** determine whether we can add the edge or not
11. **int* leader_of_vertices**: an array storing the leader of the disjoint forest set
12. **int* rank_of_vertices**: each vertices rank
13. **vector<Edge> edges:** the given graph's edges
14. **vector<Edge> final_edges**: the removed edges
15. **vector<Edge> new_final_edges**: the removed edges for directed graph
16. **vector<Edge> acyclic_graph**: the acyclic graph

Algorithms:

Undirected graph:
It must be the MST or otherwise it will form a cycle since it's an undirected graph
We use Kruskal's algorithm to form the MST: First, we make_set then union by the descending order of weight (Use counting_sort). For those not forming the MST will be add to the final_edges. Then the output of final_edges is the answer.

Directed edges:
We still do the steps first same as the undirected graph. But it won't be a good answer since we can still add some edges and still make the graph acyclic. To make

the total removed edges' weight be min, we don't need to consider the edges with negative weights. Hence, we add the edges with positive weights back to the MST and check if it will form a cycle. If it will form a cycle, then we add it to the new_final_edges, otherwise, it will be added to the acyclic graph.

Result from the given public cases:

```
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_0.in ./outputs/public_case_0.out
5
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_1.in ./outputs/public_case_1.out
21
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_2.in ./outputs/public_case_2.out
-3330
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_3.in ./outputs/public_case_3.out
-21468
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_4.in ./outputs/public_case_4.out
0
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_7.in ./outputs/public_case_7.out
-10515
alg24f078@edaU10:~/PA3$ ./checker/pa3_checker ./inputs/public_case_8.in ./outputs/public_case_8.out
-70938
```

```
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_0.in ./outputs/public_case_0.out
The total CPU time: 0.194ms
memory: 5932KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_1.in ./outputs/public_case_1.out
The total CPU time: 0.202ms
memory: 5932KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_2.in ./outputs/public_case_2.out
The total CPU time: 0.423ms
memory: 5932KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_3.in ./outputs/public_case_3.out
The total CPU time: 27.76ms
memory: 6068KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_4.in ./outputs/public_case_4.out
The total CPU time: 0.163ms
memory: 5932KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_7.in ./outputs/public_case_7.out
The total CPU time: 15.071ms
memory: 6064KB
alg24f078@edaU10:~/PA3$ ./bin/cb ./inputs/public_case_8.in ./outputs/public_case_8.out
The total CPU time: 378.187ms
memory: 6248KB
```

	case0	case1	case2	case3	case4	case7	case8
graph type	d	u	u	d	u	d	d
number of verteices	8	10	50	100	10	30	500
number of edges	9	30	300	1000	9	870	3000
removed total weight	5	21	-3330	-21468	0	-10515	-70938
CPU time (ms)	0.194	0.202	0.423	27.76	0.163	15.071	378.187
memory usage (kB)	5932	5932	5932	6068	5932	6064	6248

The README part pls refer to the README file in the zip.