

1. Write a program to implement a Binomial Heap and perform the operations insertion, deletion, extract minimum and union. Your program should contain the following functions:

- MakeHeap() - Creates and returns a new heap H containing no elements.
- Insert(H, x) – Inserts a new node with key 'x' into the heap H.
- Minimum(H) – Return the value of the smallest key in the heap H.
- ExtractMin(H) – Deletes the node with minimum key value from heap H and prints the deleted node.
- DecreaseKey(H, x, k) – If the node of H with key 'x' is at least 'k', then decreases the value of node with key 'x' by 'k'. Otherwise, it prints -1.
- Delete(H, x) - Deletes the node with key 'x' from the heap H. If node is present, it prints the deleted node else it prints -1.
- Union(H1, H2) - Create and return a new heap H that contains all the nodes of heaps H1 and H2. Heaps H1 and H2 are “destroyed” by this operation.

Input Format:

- Each line contains a character from 'i', 'j', 'm', 'x', 'r', 'd' and 'e' followed by at most one integer. The integers, if given, are in the range $[-10^6, 10^6]$.
- i k - inserts k into the heap H1.
- j k - inserts k into the heap H2.
- d k - deletes the node with key k from the heap H1 and prints the deleted node's key.
- p1 - prints the binomial heap H1.
- p2 - prints the binomial heap H2.
- m - prints the minimum element in the binomial heap H1 (Note:- In print function, level order traversal is to be used).
- x - extracts and prints the minimum element from the heap H1.
- r y z - decreases the value of node with key y by z and print the new value in H1.
- u - combine two heaps (H1 and H2) and print a new heap.
- e - 'exit' from the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.

Sample Input:

```
i 10
i 20
i 30
i 40
i 50
p 1
m
x
p 1
r 50 4
p 1
r 70 5
j 60
j 70
j 80
```

p 2

u

e

Sample Output:

```
50 10 30 20 40
10
10
20 30 50 40
46
20 30 46 40
-1
80 60 70
10 50 30 20 60 80 40 70
```

2. Write a program that implements the Disjoint-set data structure using rooted forests. Also, write functions to implement the ranked union and path compression heuristics on your data structure. Your program should compute the efficiency of the Disjoint-set data structure find operation by applying neither, either or both of the heuristics. The efficiency is calculated by counting the total number of data accesses performed over the course of the program. Your program must support the following functions:
- `makeset(x)` - creates a singleton set with element `x`.
 - `find(x)` - finds the representative of the set containing the element `x`.
 - `union(x,y)` - merges the sets containing elements `x` and `y` into a single set. The representative of the resultant set is assigned with `find(x)`, unless the ranked union heuristic is used and the ranks of both `find(x)` and `find(y)` are different. Otherwise, the representative is assigned in accordance with the ranked union heuristic.

Note that looking up an element in the data structure must be done in $O(1)$ time.

Input Format:

- The input consists of multiple lines, each one containing a character from `{'m', 'f', 'u', 's'}` followed by zero, one or two integers separated by single space. The integer(s), if given, is in the range 0 to 10000.
 - Call the function `makeset(x)` if the input line contains the character `'m'` followed by an integer `x`. Print -1 if `x` is already present in some set, and the value of `x`, otherwise.
 - Call the function `find(x)` if the input line contains the character `'f'` followed by an integer `x`. Print the value of `find(x)` if `x` is present, and -1 if `x` is not present.
 - Call the function `union(x,y)` if the input line contains the character `'u'` followed by space separated integers `x` and `y`. Print -1, without terminating, if either `x` or `y` isn't present in the disjoint set. Print `find(x)` itself if `find(x)=find(y)`. Otherwise, print the representative of the resultant set. The representative of the resultant set is assigned with `find(x)`, unless the ranked union heuristic is used and the ranks of both `find(x)` and `find(y)` are different. Otherwise, the representative is assigned in accordance with the ranked union heuristic.
 - If the input line contains the character `'s'`, print the number of data accesses performed by the find function by each of the data structures over the course of the program and terminate.

Output Format:

- The output consists of multiple lines of single space separated columns. The columns correspond to the following disjoint-set data structures: (a) with neither ranked union nor path compression applied.

(a) with neither ranked union nor path compression applied

(b) with only ranked union applied.

(c) with only path compression applied.

(d) with both ranked union and path compression applied.

- Each line in the output contains the output of the corresponding line in the input, after applying to the respective data structures.

- The last line of the output contains the number of data accesses performed by the find function by each of the data structures over the course of the program.

Sample Input

m 1

m 2

m 3

m 4

m 5

m 6

m 7

m 8

m 9

u 1 2

u 3 4

u 5 6

u 7 8

u 9 8

u 6 8

u 4 8

u 2 8

f 9

m 10

u 10 9

s

Sample Output

1

2

3

4

5

6

7

8

9

1 1 1 1

3 3 3 3

5 5 5 5

7 7 7 7

9 7 9 7

5 5 5 5

3 5 3 5

1 5 1 5

1 5 1 5

10

10 5 10 5

38 32 33 30

3. A company manager wants to implement an onsite project. For the implementation of the project several teams have to be formed. To formulate the teams, the manager collects preferences as pairs from the employees, which consists of a pair of employee IDs, EID. For each of the preferences (a, b), the employees a and b are made part of the same team. All the

employees should take part in the selection process. One employee can give more than one preference in which he/she is part of. Based on the given preferences, formulate teams. Once the teams are created, several groups have to be sent out for data collection. It has to be ensured that each of the data collecting group consists of exactly two employees, from different teams.

Write a program using the disjoint set data structure to formulate the teams, print all possible data collecting groups and checks whether a given data collecting group is valid or not. The output should print the following:

- The number of teams formed using the preferences.
- Count of all possible data collecting groups (x, y) such that $x < y$.
- 1 if the given pair is a valid data collecting group, -1 otherwise.

Note that EID ranges from 1 to 10^3 and all EIDs are unique. If there is only one team formed at the end, then print -1 as no data collecting groups can be obtained.

Input Format:

- Each line contains a character t, T, d, v and e , followed by at most two integers.
- $t\ a\ b$ - create the team with EIDs a and b .
- T - Prints the number of teams formulated.
- d - Prints the count of all possible data collecting groups.
- $v\ x\ y$ - checks whether data collecting group (x, y) is valid or not. If it is valid, print 1 else -1 .
- e - 'exit' from the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.

Input 1:

```
t 2 1
t 4 1
t 5 6
t 6 12
t 109 34
T
d
v 4 5
v 34 2
```

Output 1:

```
3
21
1
-1
```

Input 2:

```
t 657 566
T
d
v 566 657
v 200 205
```

Output 2:

```
1
-1
```

-1
-1