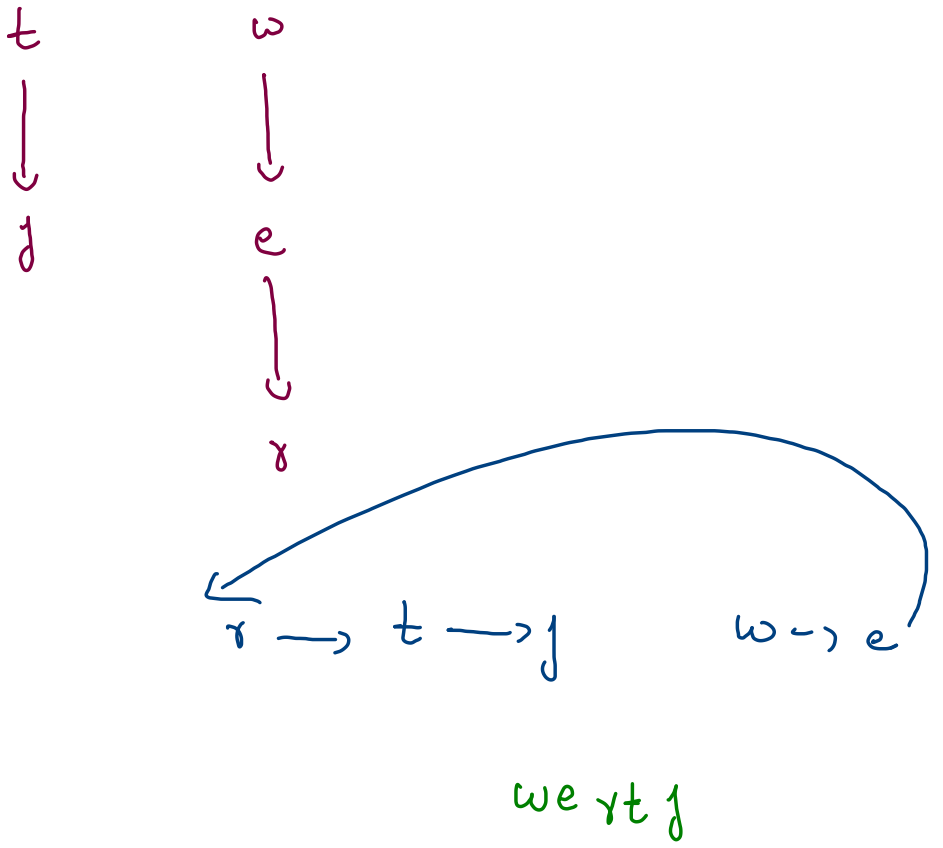
Alien dictionary

① wrt, wrf, er, ertt, rftt

- tfwer
- wertf

② wrt, wrf, er, ett, rftt



wertf

graph construct

```
HashMap<Character,ArrayList<Character>>graph = new HashMap<>();


for(String word : words) {
    for(int i=0; i < word.length();i++) {
        char ch = word.charAt(i);
        if(graph.containsKey(ch) == false) {
            graph.put(ch,new ArrayList<>());
        }
    }
}

for(int k=0; k < words.length-1;k++) {
    String w1 = words[k];
    String w2 = words[k+1];

    int i=0,j=0;

    while(i < w1.length() && j < w2.length()) {
        char ch1 = w1.charAt(i);
        char ch2 = w2.charAt(j);

        if(ch1 != ch2) {
            graph.get(ch1).add(ch2);
            graph.put(ch1,graph.get(ch1));
            break;
        }

        i++;
        j++;
    }

    if(i < w1.length() && j == w2.length()) {
       return ""; //invalid order of words
    }

}
```

wrt, wrf, er, ett, rftt

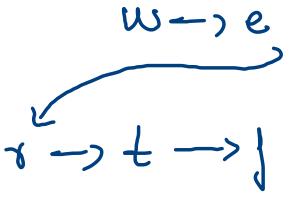
w → e

r → t

t → f

f →

e → r



```java
//topological sort using kahn's
HashMap<Character,Integer>indeg = new HashMap<>();

for(char v : graph.keySet()) {
    indeg.put(v,0);
}

for(char v : graph.keySet()) {
    for(char nbr : graph.get(v)) {
        if(indeg.containsKey(nbr) == false) {
            indeg.put(nbr,0);
        }
        indeg.put(nbr,indeg.get(nbr) + 1);
    }
}


ArrayDeque<Character>q = new ArrayDeque<>();

for(char v : indeg.keySet()) {
    if(indeg.get(v) == 0) {
        q.add(v);
    }
}

String order = "";
while(q.size() > 0) {
    char rem = q.remove();

    order += rem;

    for(char nbr : graph.get(rem)) {
        indeg.put(nbr,indeg.get(nbr)-1);

        if(indeg.get(nbr) == 0) {
            q.add(nbr);
        }
    }
}

if(order.length() == graph.size()) {
    return order;
}
else {
    return "";
}
```

graph

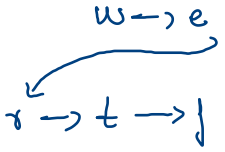
w → e

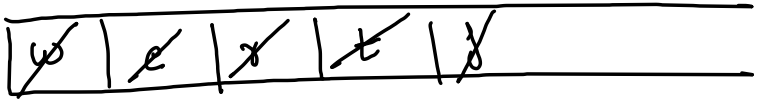
r → t

t → j

j →

e → r





indeg

w → 0

r → ~~0~~ ~~1~~ 0

t → ~~0~~ ~~1~~ 0

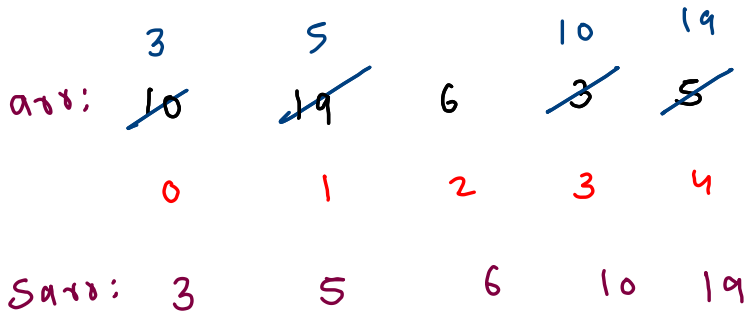
j → ~~0~~ ~~1~~ 0

e → ~~0~~ ~~1~~ 0

order : wertj

min swaps required to sort an array

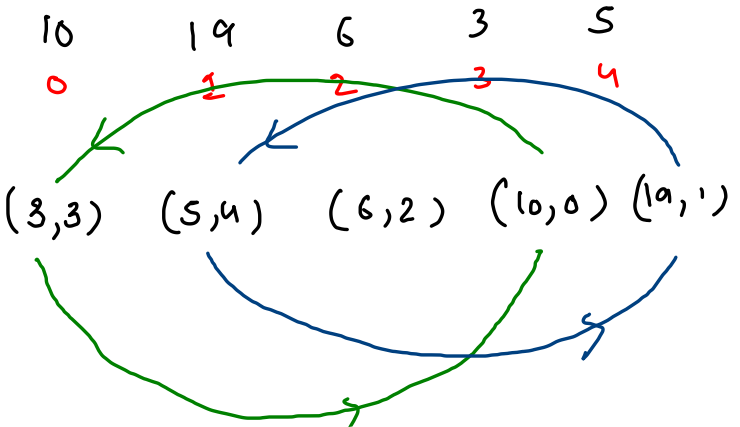
```
5
10 19 6 3 5
```



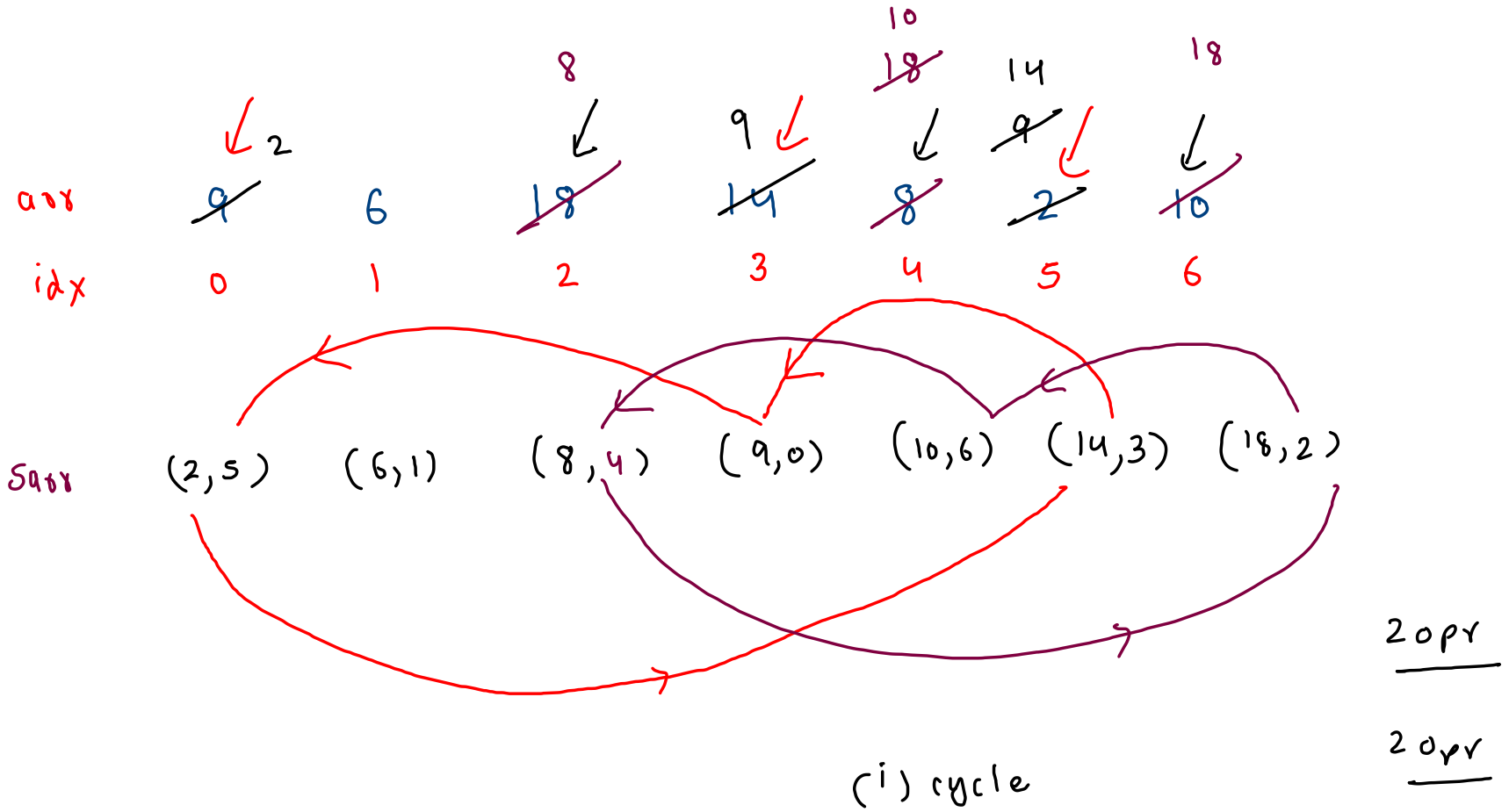
(2 swaps)

10 – 3

5 – 19



ans = 2


arr
idx
0
1
2
3
4
5
6
Sqrr
(2,5)
(6,1)
(8,4)
(9,0)
(10,6)
(14,3)
(18,2)
(i) cycle
2opr
2orr

```
for(int i=0; i < n;i++) {
    if(vis[i] == true || i == arr[i].idx) {
        continue;
    }

    int k = i;
    vis[k] = true;
    int count = 1;

    while(true) {
        k = arr[k].idx;

        if(vis[k] == true) {
            break;
        }

        count++;
        vis[k] = true;
    }

    minopr += (count - 1);
}
```