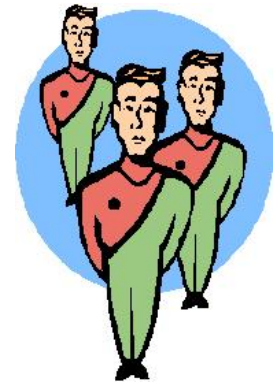




# Conception Avancée de Bases de Données



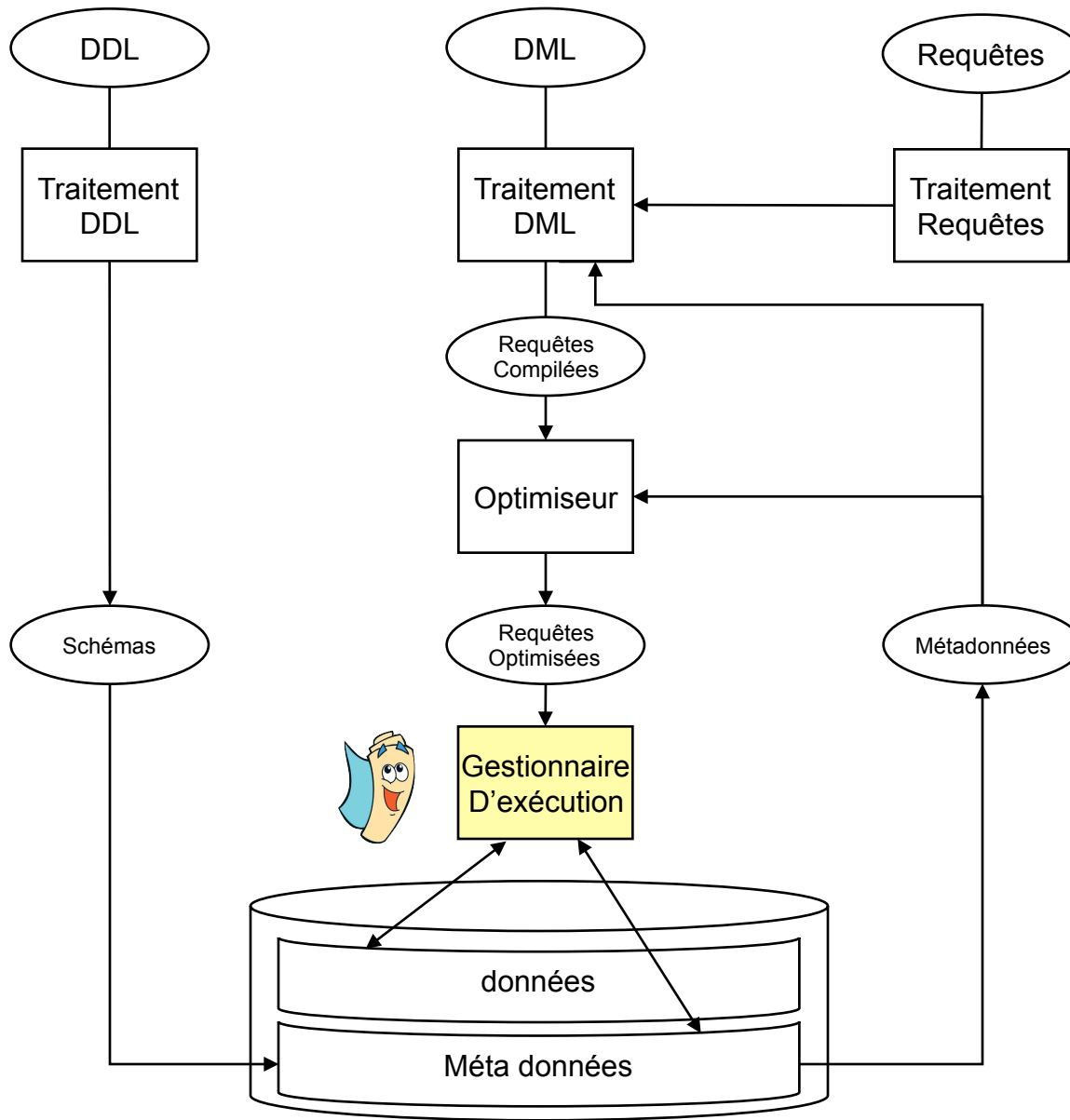
Merge join with duplicates



**Traduction en cours**



D'après C.J DATE



*DDL : langage de définition des données; DML : langage de manipulation des données*

# Algorithme Merge Join avec doublon



$i=j=1$

while  $i < (\text{\#tuples in } R)$  and  $j < (\text{\#tuples in } S)$

if  $R(i) = S(j)$  then

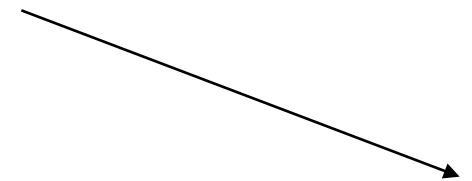
MergeDuplicate()

endif

else if  $R(i) > S(j)$  then  $j=j+1$

else  $R(i) < S(j)$  then  $i=i+1$

end while



## Merge Join

# MergeDuplicate()



k=j

while R(i) = S(j)

    while R(i) = S(j) then

        output(R(i), S(j))

        j= j+1

    end while

    i=i+1

    j=k

end while

## Merge Join

# MergeDuplicate()



**Outer Loop**

**Inner Loop**

k=j

while R(i) = S(j)

while R(i) = S(j) then

output(R(i), S(j))  
j= j+1

end while

i=i+1

j=k

end while

**Merge Join**

# R Triée

A
B
G
J
U
K
E
Z
V
B



A
B
B
E
G
J
K
U
V
Z



# S triée



B
U
E
K
X
V
N
B
M
U



B
B
E
K
M
N
U
U
V
X

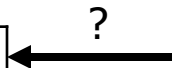
A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X

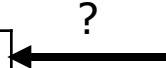





A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X






A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X


**A < B**

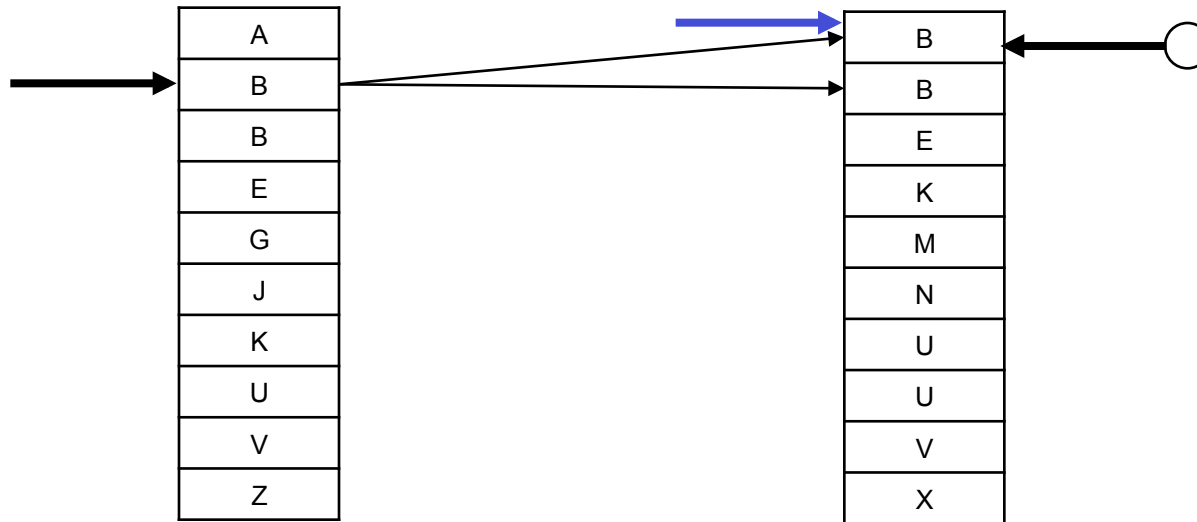


	A
→	B
	B
	E
	G
	J
	K
	U
	V
	Z

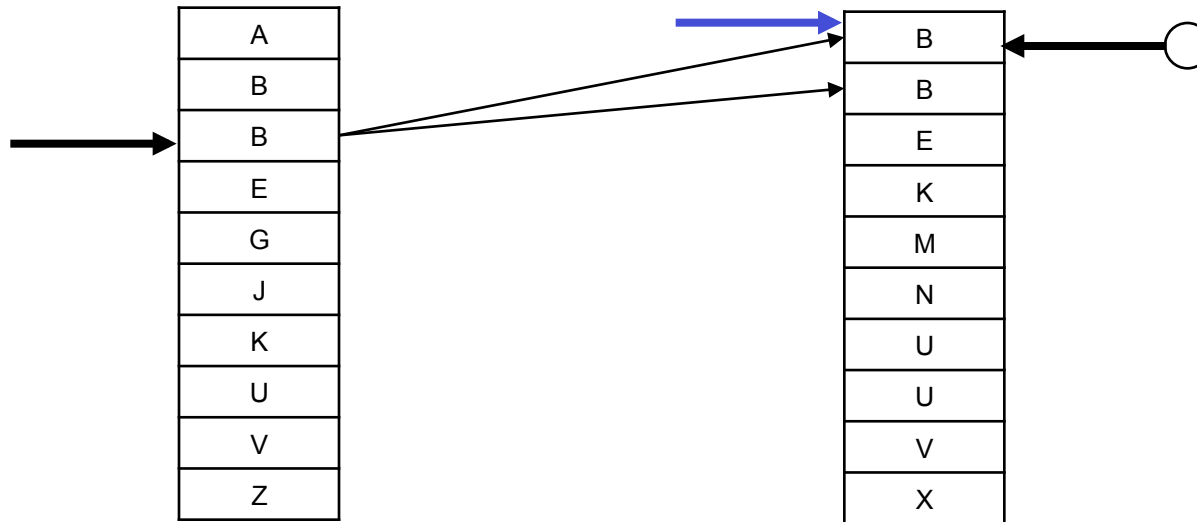
	B	← ○
	B	
	E	
	K	
	M	
	N	
	U	
	U	
	V	
	X	




**Match**



**Double Balayage  
(boucles imbriquées)  
De R sur S  
égalité**

B

**Double Balayage  
(boucles imbriquées)  
De R sur S  
égalité**



## External Loop

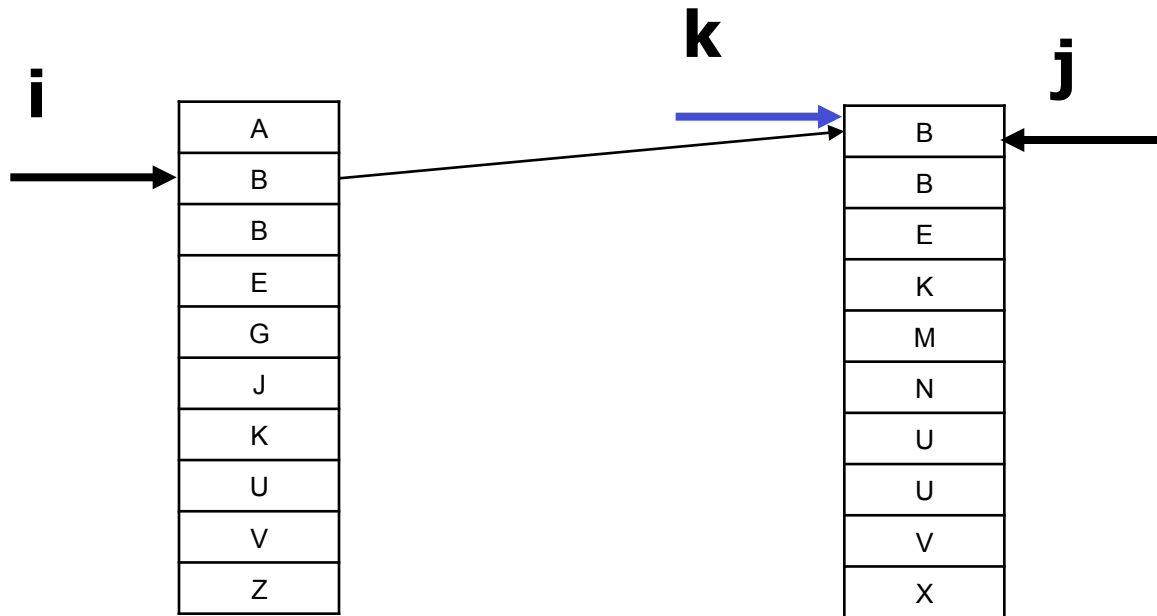
R	
B	
B	

## Internal Loop

S	
B	
B	

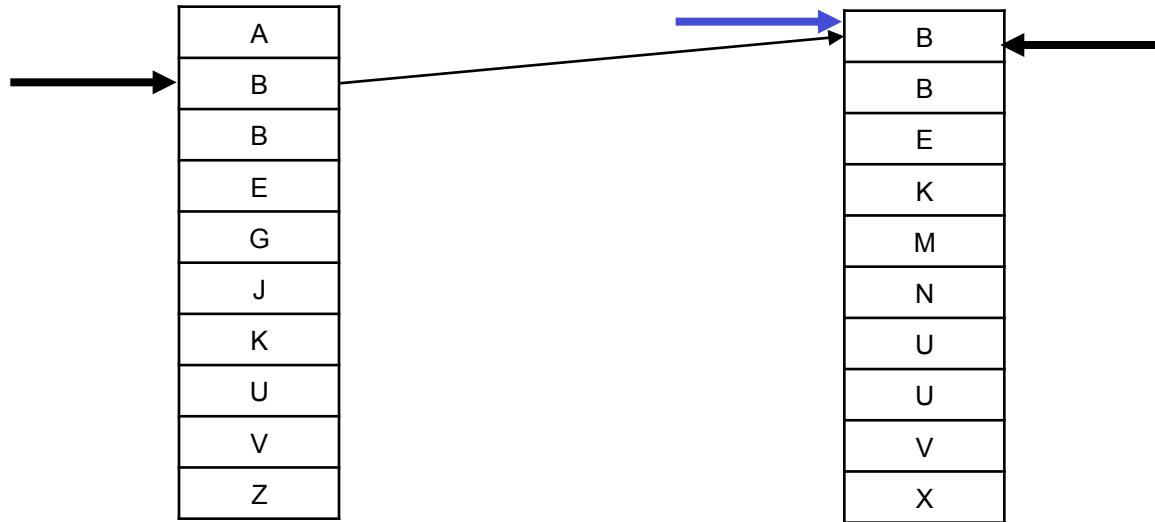
**Nested Loop sur les « duplicates »**





**Match**

# Inner loop

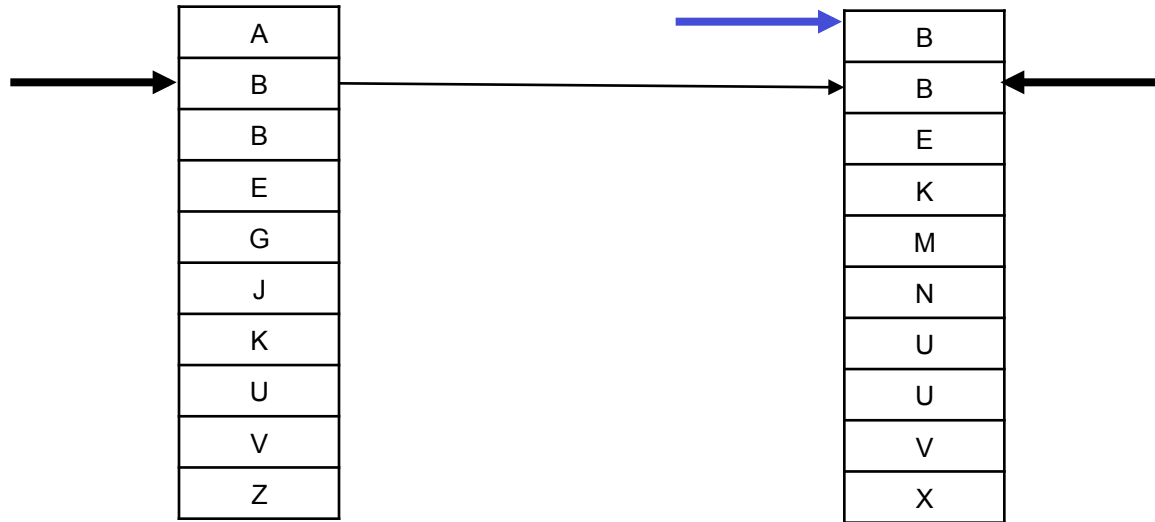


## Match

B



# Inner loop



B
B

**Match**

	A
→	B
	B
	E
	G
	J
	K
	U
	V
	Z

→	B	
	B	
	E	←
	K	
	M	
	N	
	U	
	U	
	V	
	X	

B
B

**E > B**



A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X

B
B

**E > B**



# Outer loop

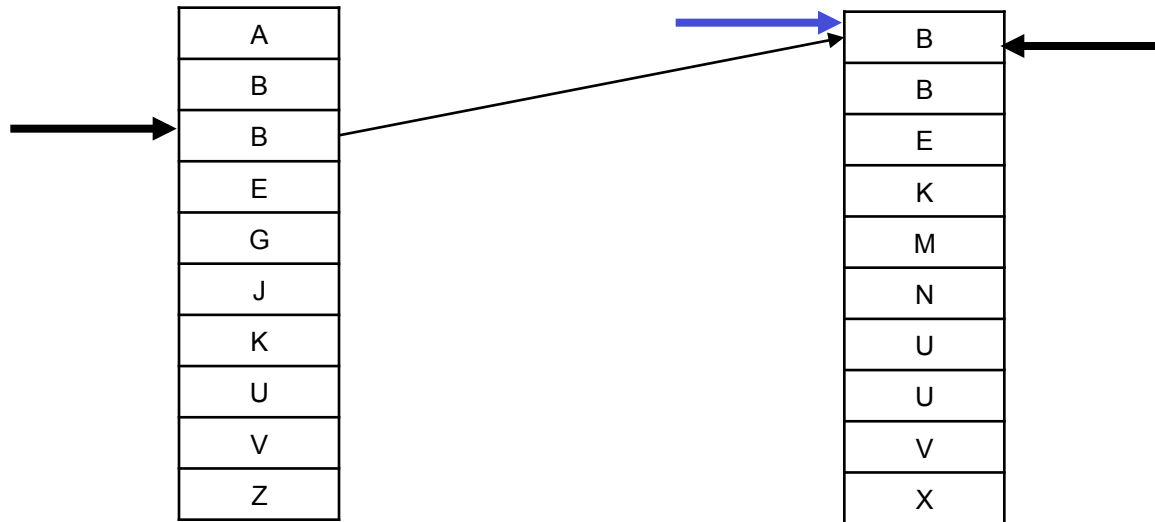
	A
	B
→	B
	E
	G
	J
	K
	U
	V
	Z

→	B	
	B	
	E	←
	K	
	M	
	N	
	U	
	U	
	V	
	X	

B
B

**E > B**

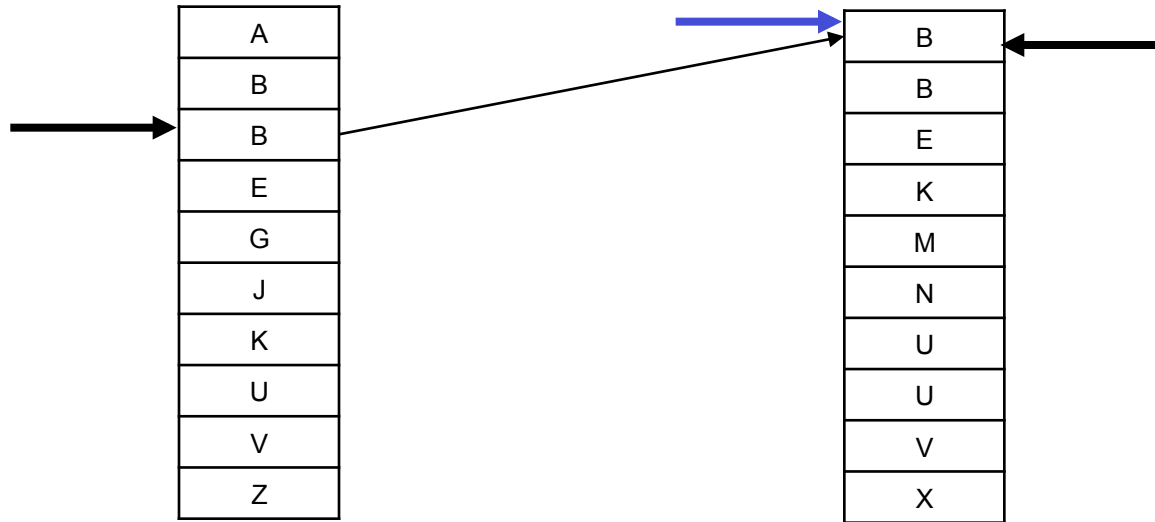




B
B
B

**Match**

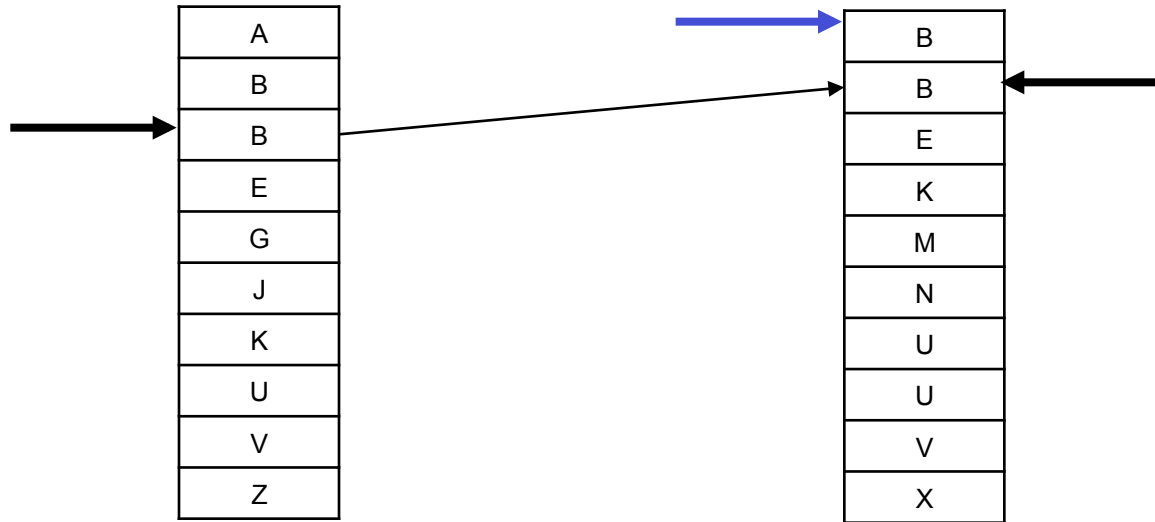
# Inner loop



## Match

B
B
B

# Inner loop



B
B
B
B

**Match**



A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X

B
B
B
B

**B < E**





A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B

**B < E**



A
B
B
E
G
J
K
U
V
Z



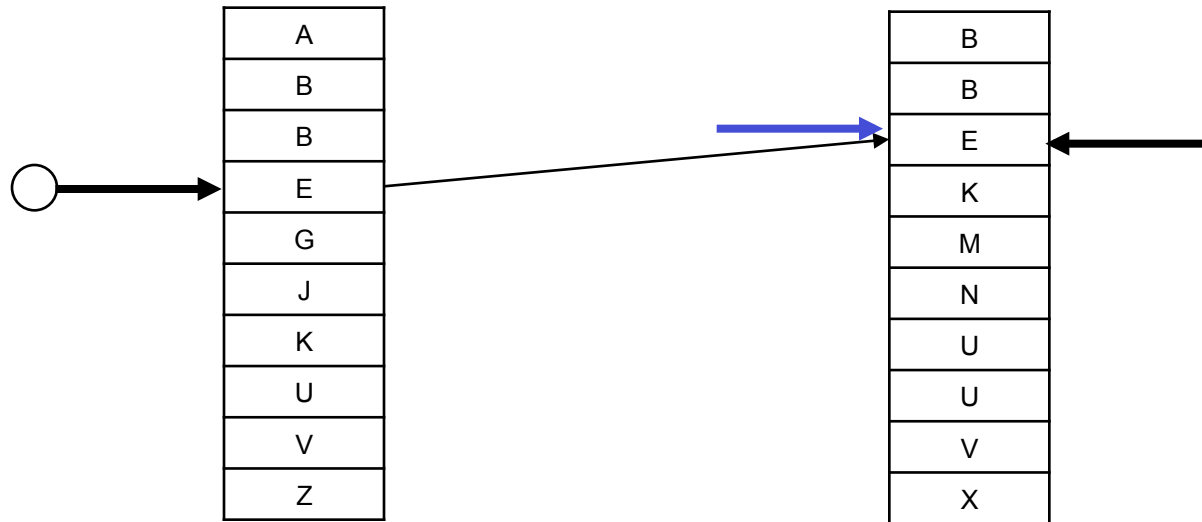
B
B
E
K
M
N
U
U
V
X



B
B
B
B


**Match**






B
B
B
B
E


**Match**



A
B
B
E
G
J
K
U
V
Z




B
B
E
K
M
N
U
U
V
X




B
B
B
B
E

**K > E**





A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X

B
B
B
B
E

**G < K**



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B
E



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B
E

**J < K**



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X

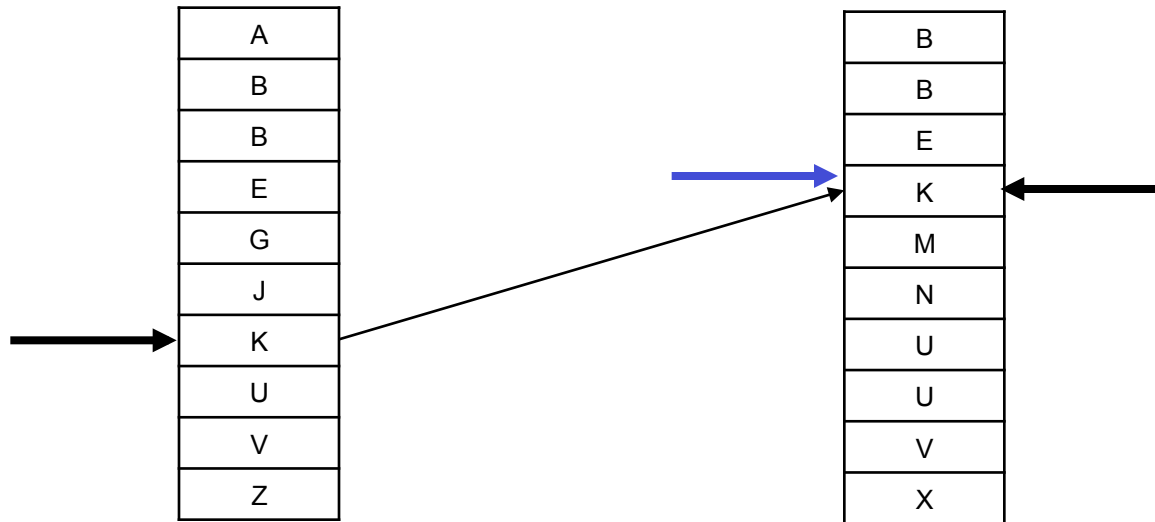


B
B
B
B
E

**Match**







B
B
B
B
E
K

Match



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B
E
K

**K < M**



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B
E
K

**K < M**



A
B
B
E
G
J
K
U
V
Z




B
B
E
K
M
N
U
U
V
X




B
B
B
B
E
K

**U > M**






A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X

B
B
B
B
E
K





A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X



B
B
B
B
E
K

**U > N**



A
B
B
E
G
J
K
U
V
Z



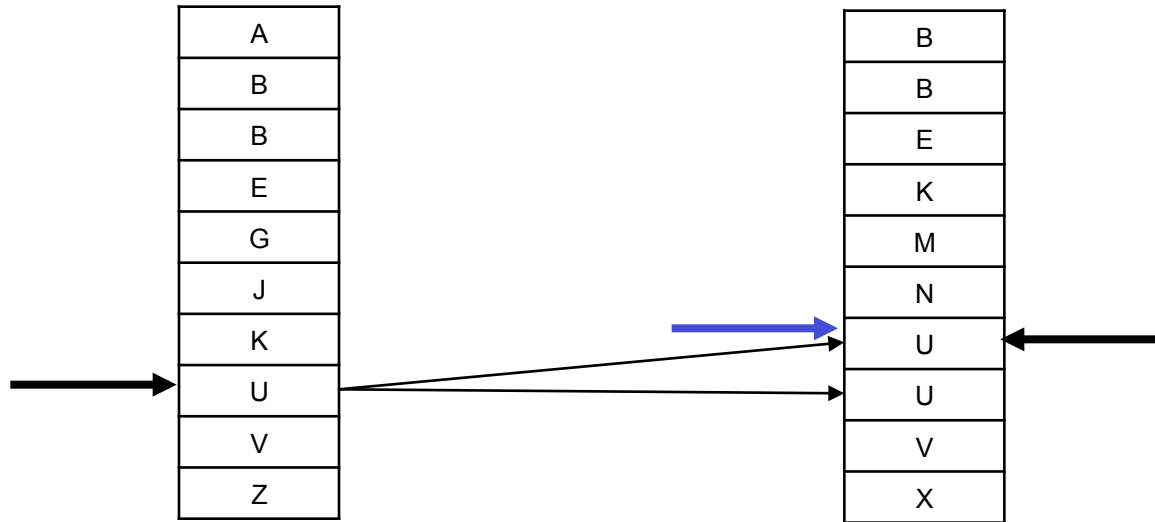
B
B
E
K
M
N
U
U
V
X



B
B
B
B
E
K

**Match**

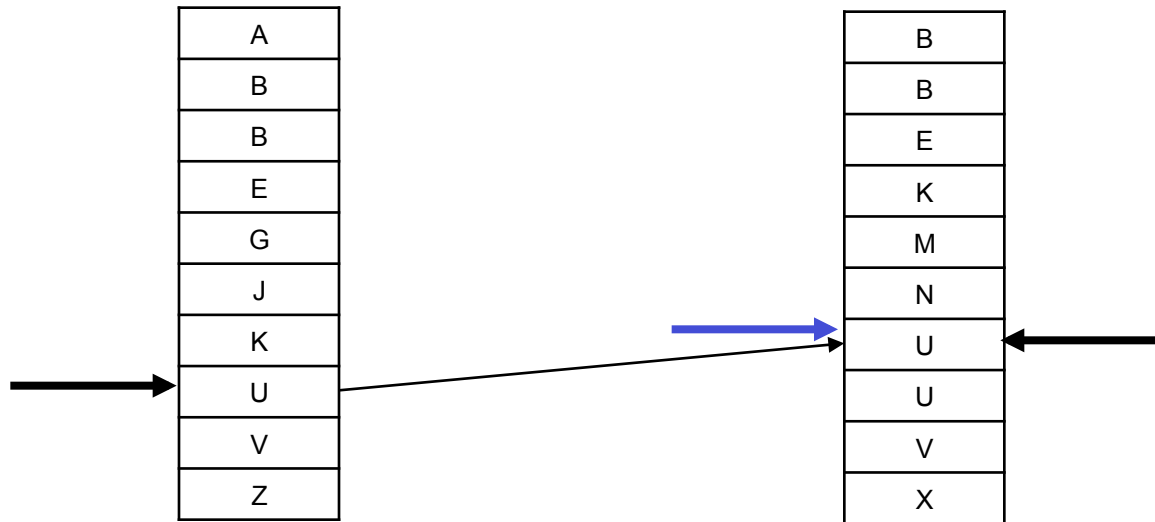




B
B
B
B
E
K

**Match**

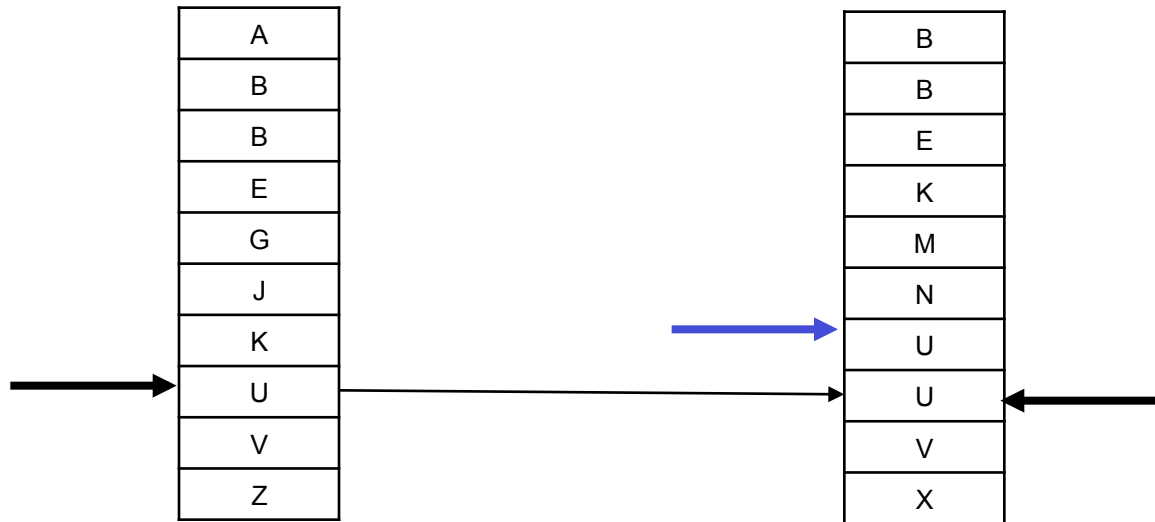




B
B
B
B
E
K
U

**Match**





B
B
B
B
E
K
U
U

**Match**



A
B
B
E
G
J
K
U
V
Z



B
B
E
K
M
N
U
U
V
X



B
B
B
B
E
K
U
U

**U < V**





	A
	B
	B
	E
	G
	J
	K
→	U
	V
	Z

	B
	B
	E
	K
	M
	N
	U
	U
←○	V
	X

B
B
B
B
E
K
U
U

**U < V**

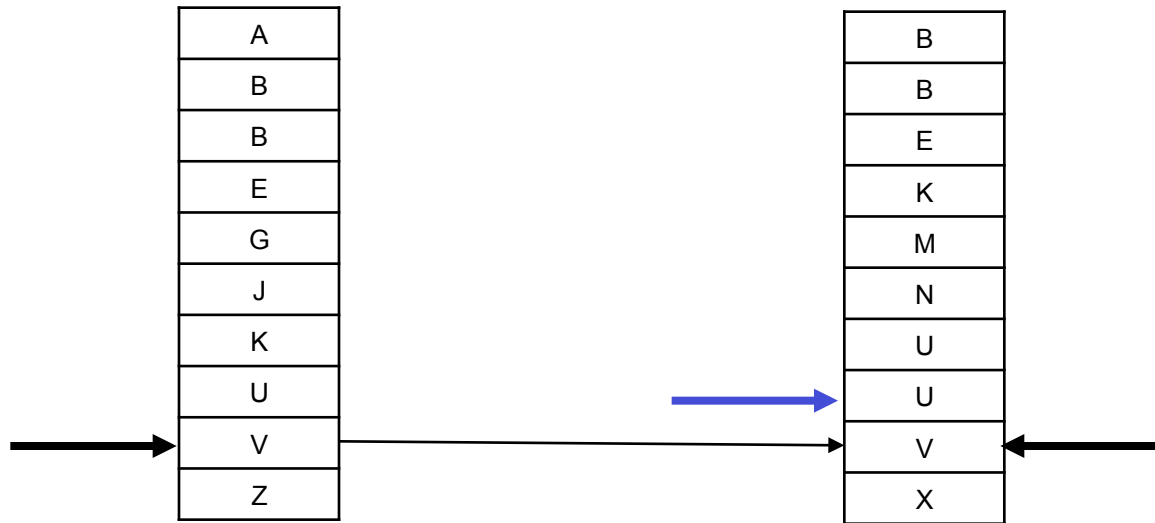


	A
	B
	B
	E
	G
	J
	K
	U
→	V
	Z

	B
	B
	E
	K
	M
	N
	U
	U
←○	V
	X

B
B
B
B
E
K
U
U

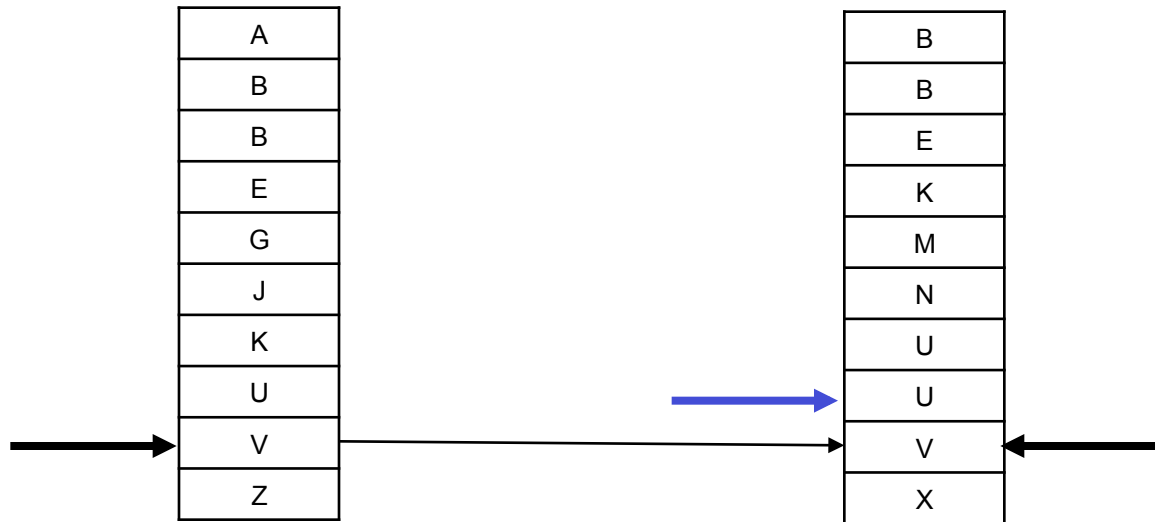
**Match**



B
B
B
B
E
K
U
U

**Match**

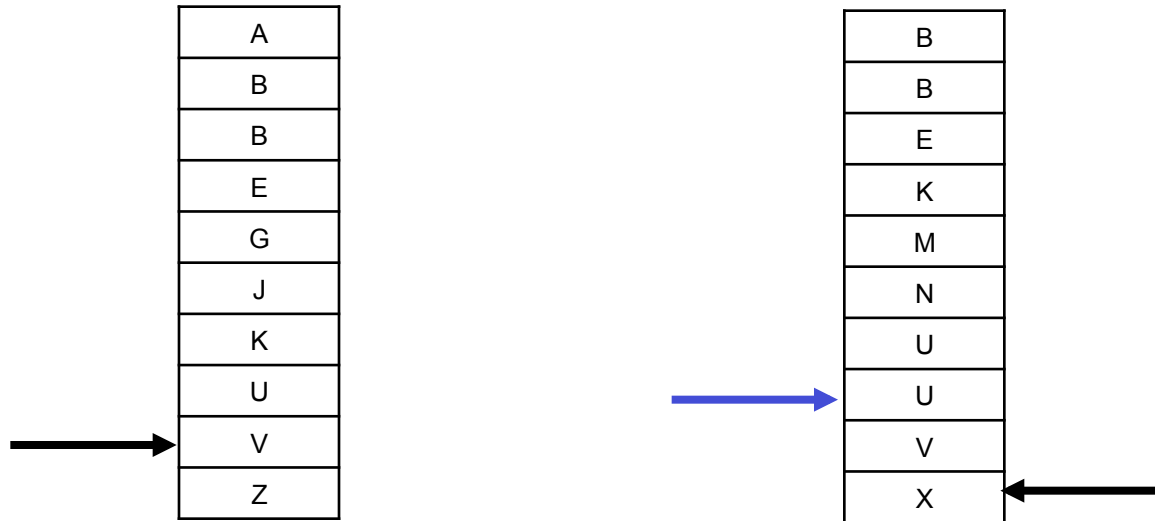




B
B
B
B
E
K
U
U
V

**Match**





**V < X**







	A
	B
	B
	E
	G
	J
	K
	U
→	V
	Z

	B
	B
	E
	K
	M
	N
	U
	U
	V
←○	X

B
B
B
B
E
K
U
U
V

**V < X**



A
B
B
E
G
J
K
U
V
Z

B
B
E
K
M
N
U
U
V
X

B
B
B
B
E
K
U
U
V

**Z > X**



	A
	B
	B
	E
	G
	J
	K
	U
	V
	Z

	B
	B
	E
	K
	M
	N
	U
	U
	V
	X

B
B
B
B
E
K
U
U
V

**Fin**

```
public static Vector<String> join ( Vector<String> R, Vector<String> S){
```

```
    int i=0;  
    int j=0;  
    int k=0;
```

```
    Vector<String> RS = new Vector<String>();
```

```
    Collections.sort(R);  
    Collections.sort(S);
```

```
    while( i < R.size() && j < S.size() ){
```

```
        if( R.elementAt(i).compareTo(S.elementAt(j)) == 0 ){
```

```
            // if R(i) = S(j)
```

```
            k = j;
```

```
            while( (i < R.size()) &&  
                (R.elementAt(i).compareTo(S.elementAt(j)) == 0) ){
```

```
                //      while R(i) = S(j)
```

```
                while((j < S.size()) && (  
                    R.elementAt(i).compareTo(S.elementAt(j)) == 0)){
```

```
                    //      while R(i) = S(j) then
```

```
                        RS.add(R.elementAt(i));
```

```
                        j++;
```

```
                    }
```

```
                    i++;  
                    j = k;
```

```
                }
```

```
            }
```



```
        else if( R.elementAt(i).compareTo(S.elementAt(j)) > 0 )
```

```
            //      if R(i) > S(j) then j=j+1  
            j++;
```

```
        else if( R.elementAt(i).compareTo(S.elementAt(j)) < 0 )
```

```
            //      else R(i) < S(j) then i=i+1  
            i++;
```

```
        }  
        return RS;  
    }
```

Java Version  
with vectors

