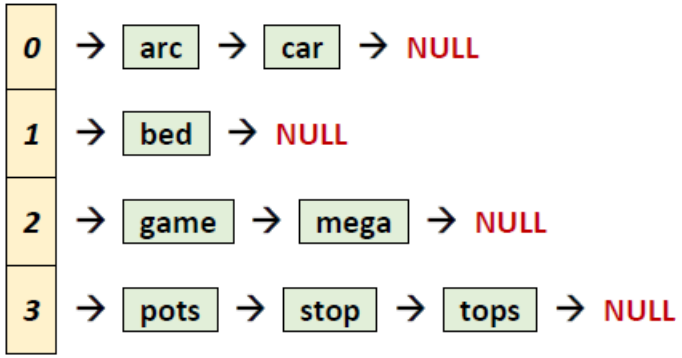
Step #2. LIST A should then be **sorted** (example):

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST A (array A)

Sorted 1-D array of words

Step #3. A new **LIST B** (**1-D array** of **singly linked lists**) should then be created directly from **LIST A** (i.e., the sorted 1-D array of words, step #2) as follows (example):



LIST B

1-D array of **singly linked lists**

We can have different ways of building LIST B from LIST A

One possible solution is shown in the next slides

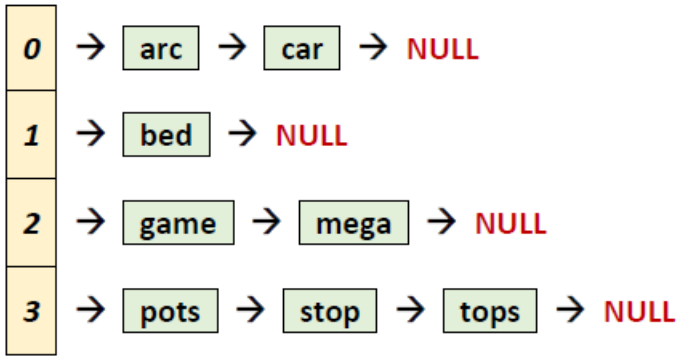
Step #2. LIST A should then be **sorted** (example):

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST A (array A)

Sorted 1-D array of words

Step #3. A new **LIST B** (**1-D array** of **singly linked lists**) should then be created directly from **LIST A** (i.e., the sorted 1-D array of words, step #2) as follows (example):



LIST B

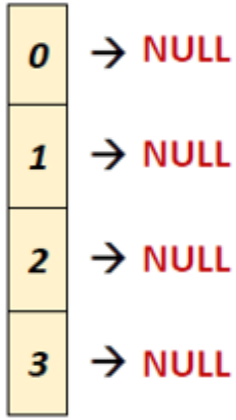
1-D array of **singly linked lists**

Declare **LIST B** as an array large enough to store the anagrams, e.g. LIST B with size 4

LIST A

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST B



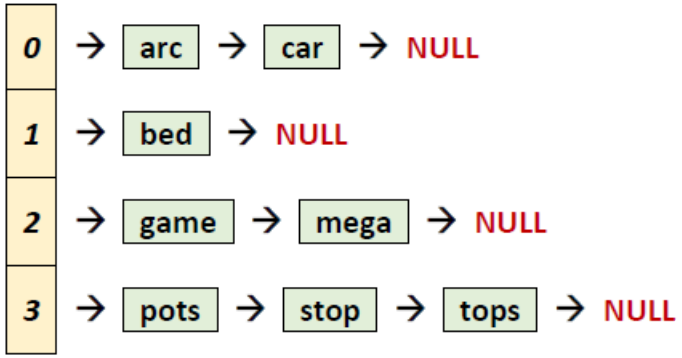
Step #2. LIST A should then be **sorted** (example):

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST A (array A)

Sorted 1-D array of words

Step #3. A new **LIST B** (**1-D array** of **singly linked lists**) should then be created directly from **LIST A** (i.e., the sorted 1-D array of words, step #2) as follows (example):



LIST B

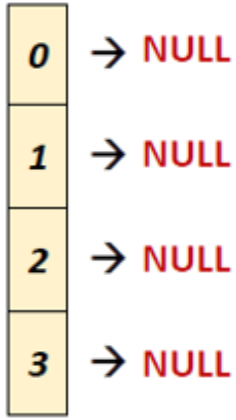
1-D array of **singly linked lists**

Declare **LIST B** as an array large enough to store the anagrams, e.g. LIST B with size 4

LIST A

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST B

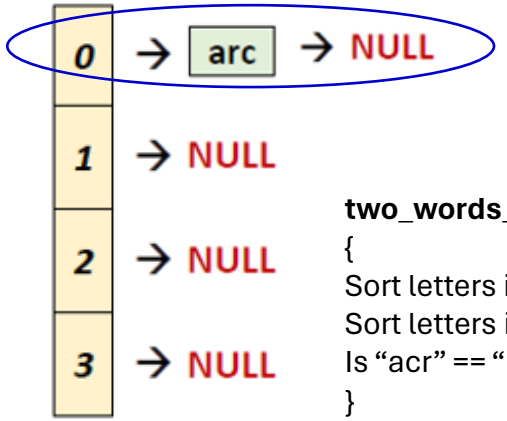


Now, let's traverse LIST A and start building LIST B
Start from LIST A[0], blue arrow; **Is "arc" in the linked list? NO, so insert LIST A[0], i.e., "arc" in the linked list at LIST B[0] and then progress to search for anagrams to "arc"**

LIST A

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

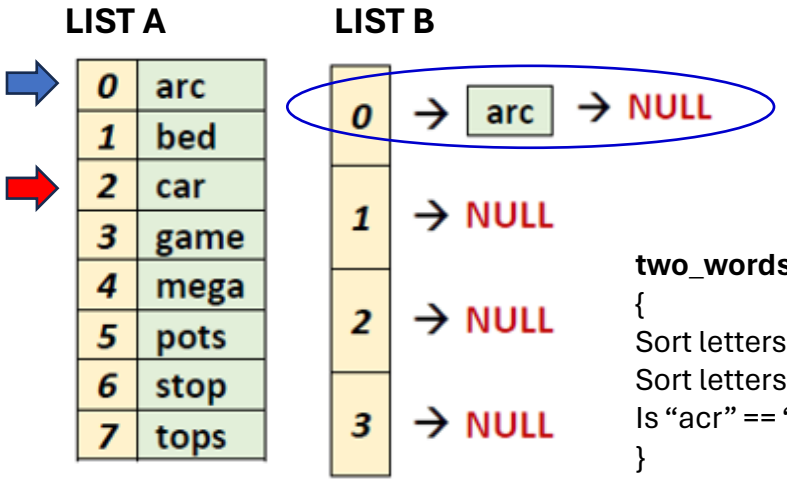
LIST B



Word 1 = arc

Word 2 = bed

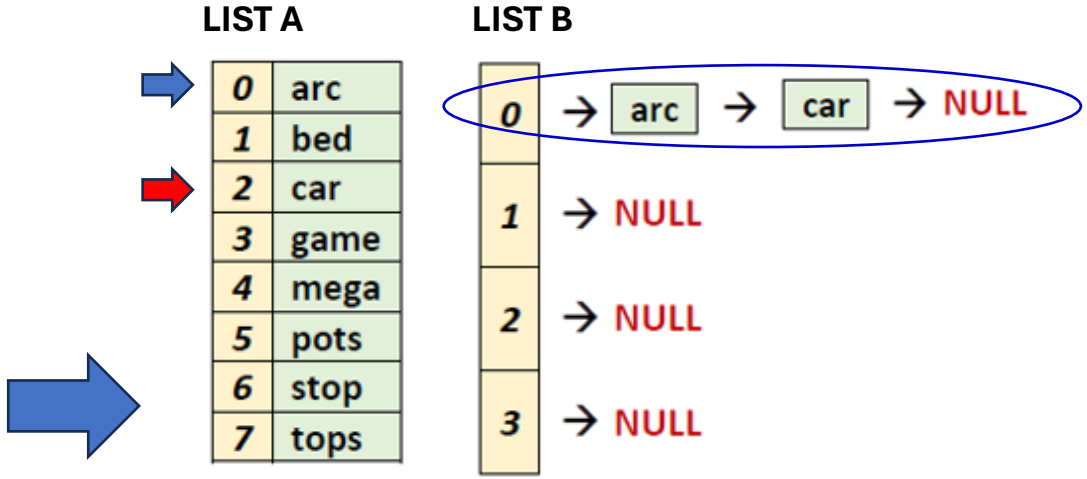
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "acr"
Sort letters in Word 2 → "bde"
Is "acr" == "bde" NO, so they are not anagrams
}



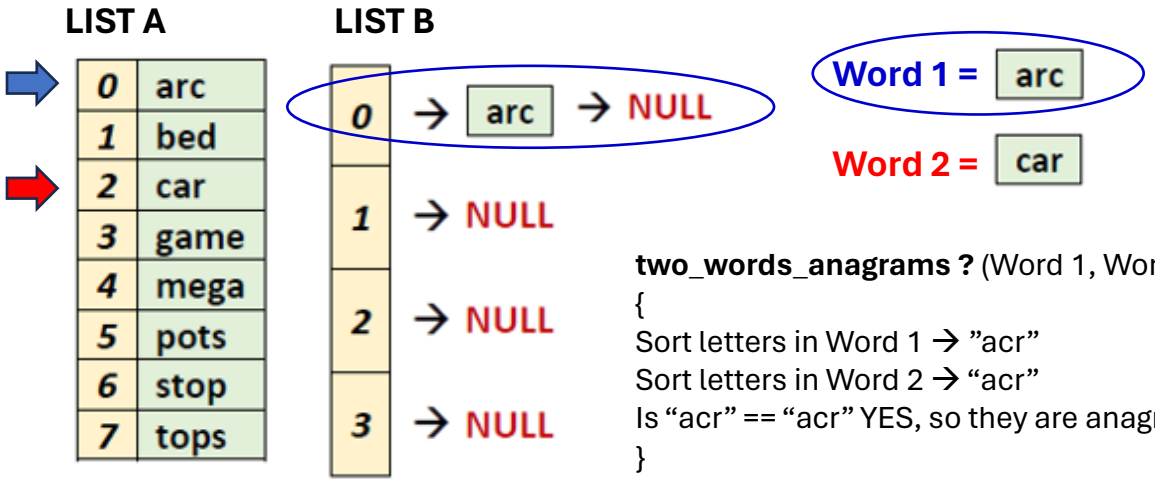
Word 1 = arc

Word 2 = car

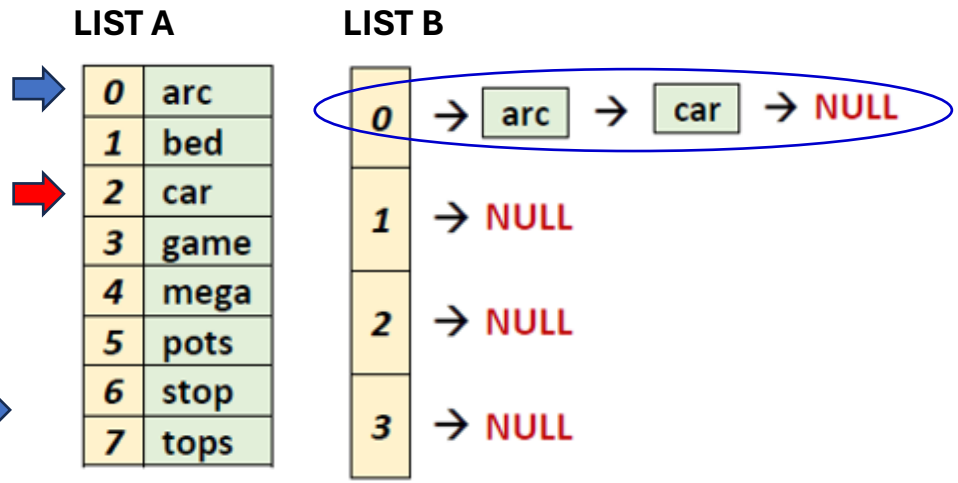
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “acr”
Is “acr” == “acr” YES, so they are anagrams
}



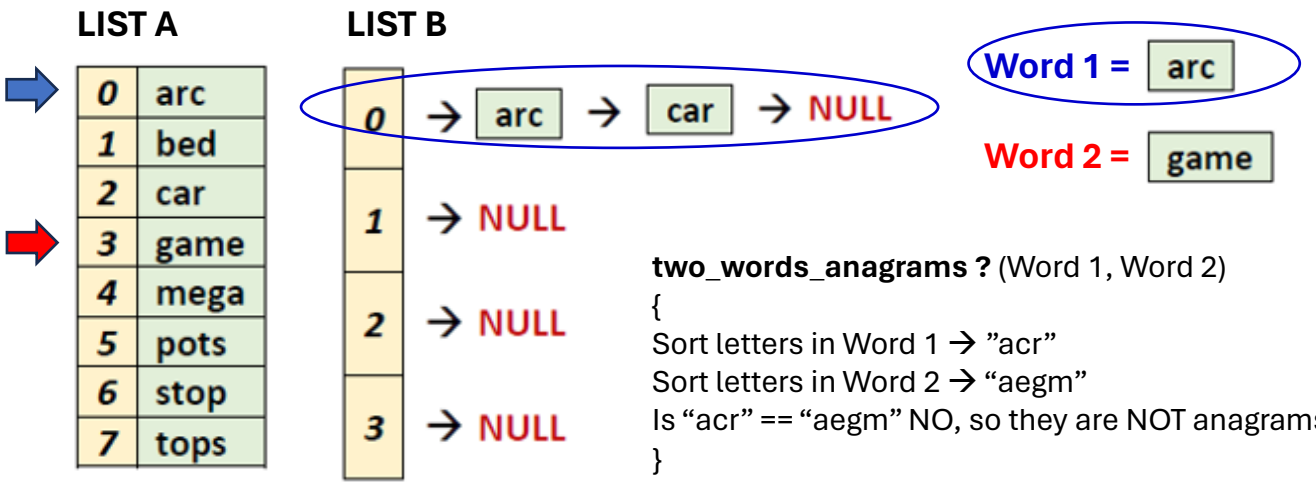
Add “car” to the linked list **after** “arc”



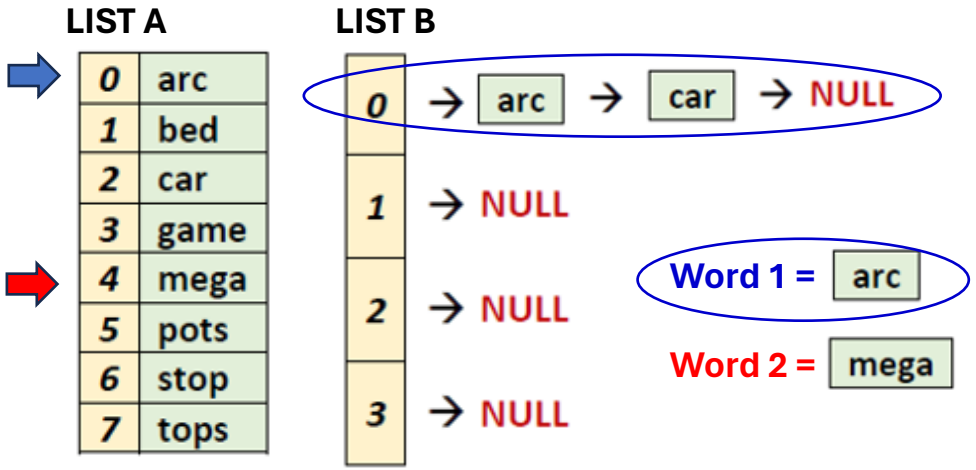
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “acr”
Is “acr” == “acr” YES, so they are anagrams
}



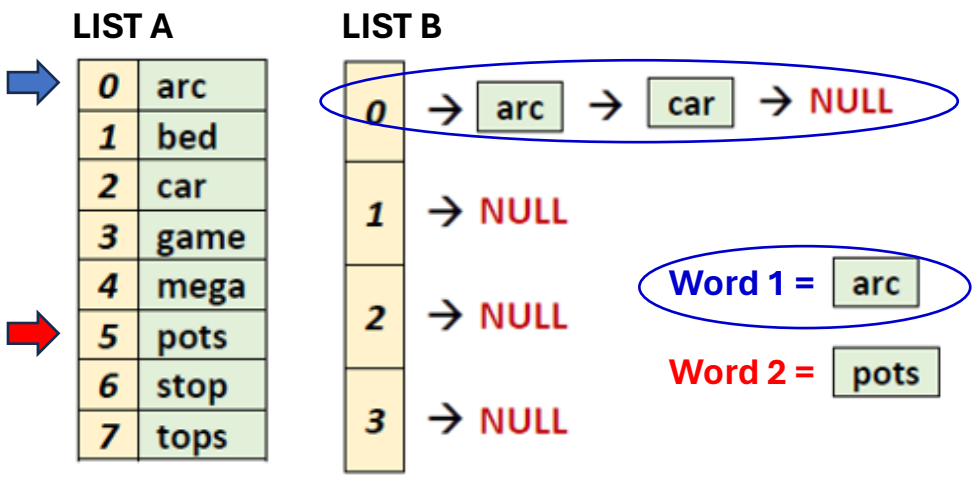
Add “car” to the linked list **after** “arc”



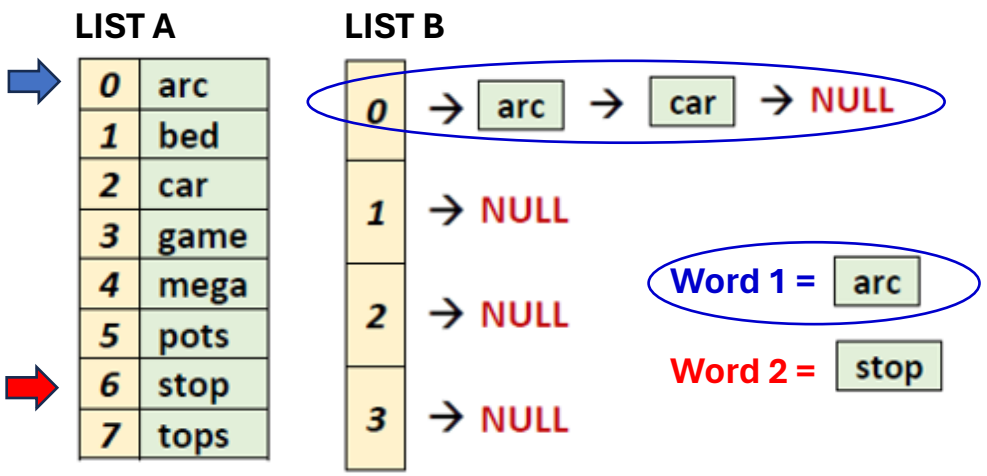
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “aegm”
Is “acr” == “aegm” NO, so they are NOT anagrams
}



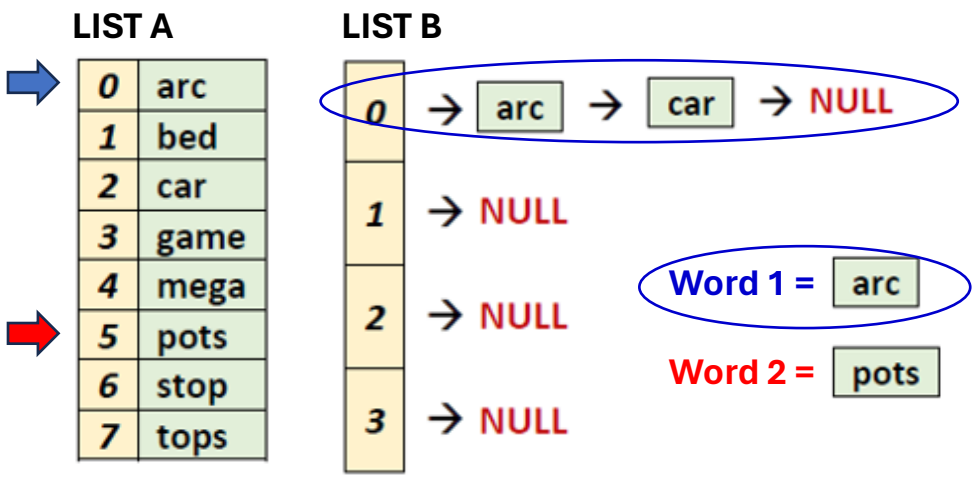
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “aegm”
Is “acr” == “aegm” NO, so they are NOT anagrams
}



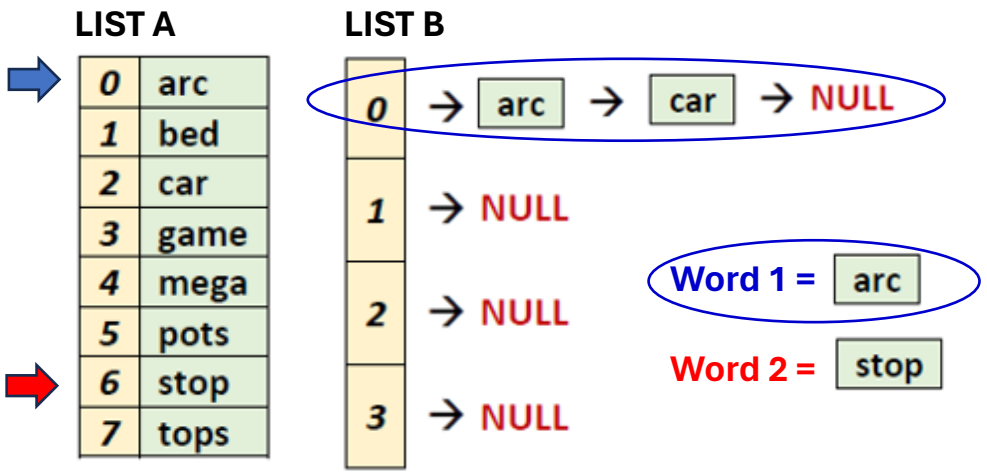
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “opst”
Is “acr” == “opst” NO, so they are NOT anagrams
}



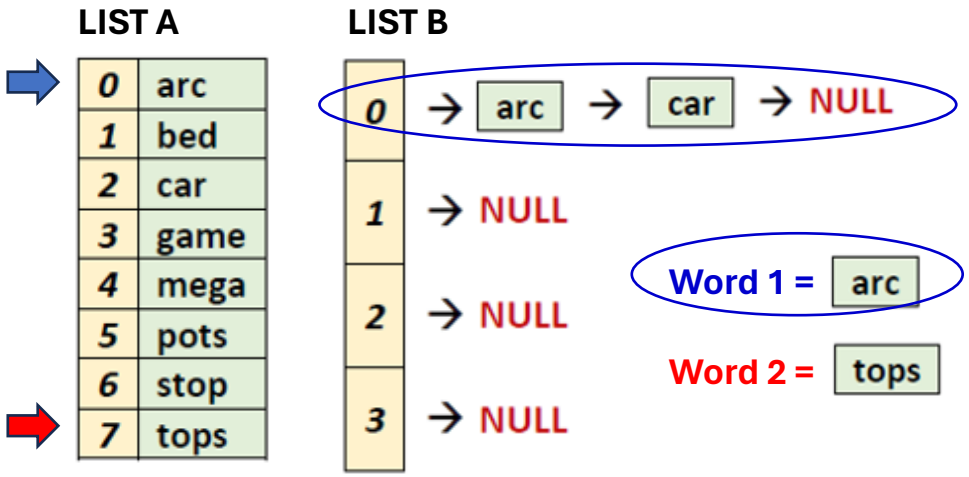
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “opst”
Is “acr” == “opst” NO, so they are NOT anagrams
}



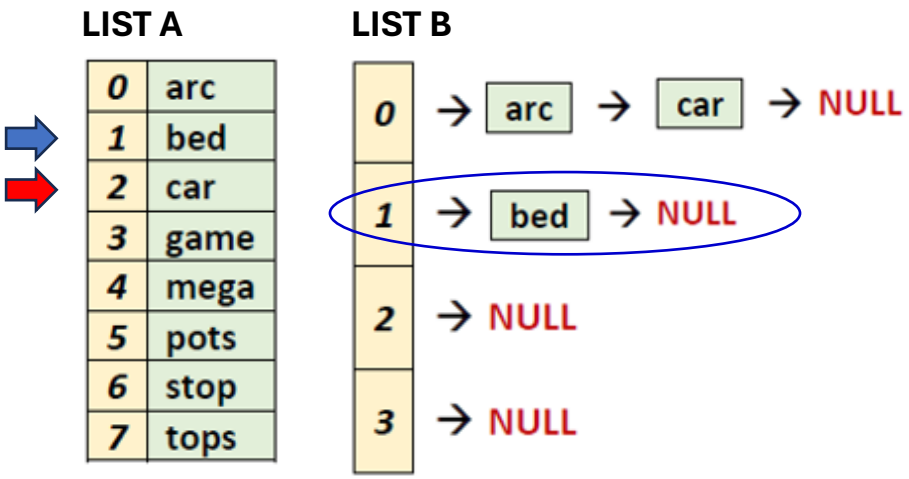
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “opst”
Is “acr” == “opst” NO, so they are NOT anagrams
}



two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “opst”
Is “acr” == “opst” NO, so they are NOT anagrams
}



two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → ”acr”
Sort letters in Word 2 → “opst”
Is “acr” == “opst” NO, so they are NOT anagrams
}

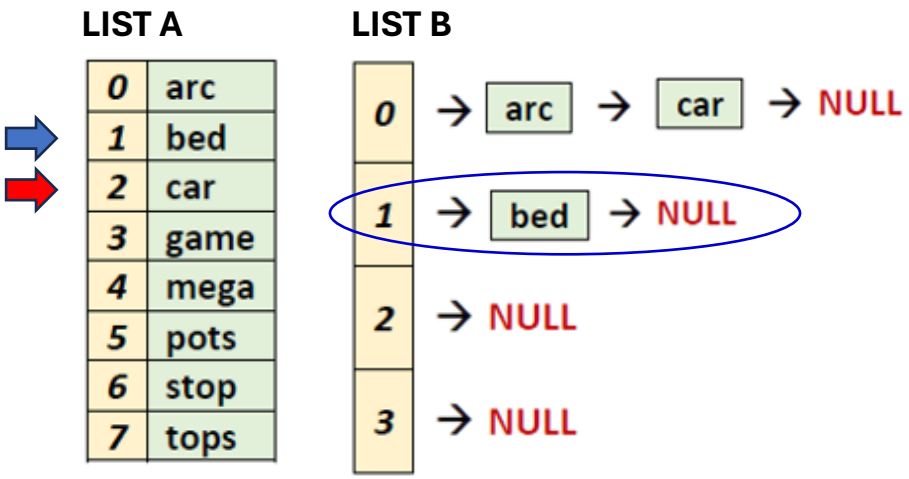


Now we are in LIST A[1], blue arrow; Is LIST A[1] = "bed" in the linked list? NO, so insert LIST A[1] = "bed" in the next entry LIST B[1] linked list, forming the <u>second group</u> of anagrams for "bed" and then start searching for anagrams to "bed"

Word 1 = bed

Word 2 = car

two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "acr"
Is "bde" == "acr" NO, so they are NOT anagrams
}

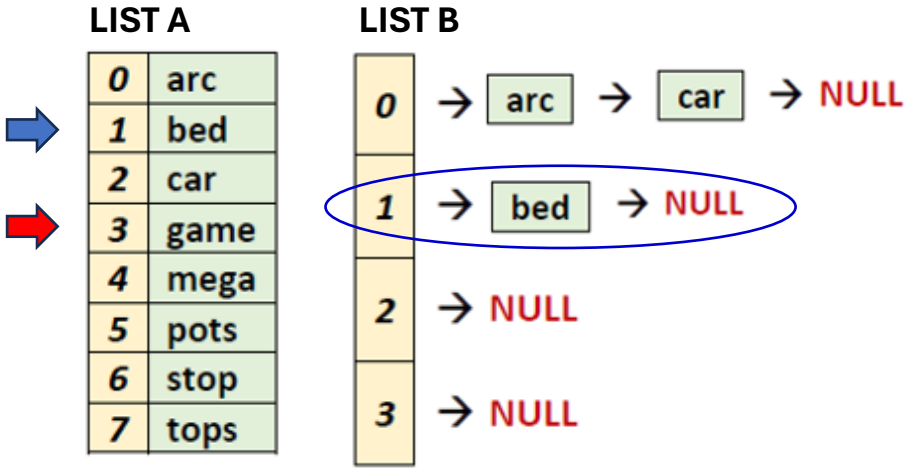


Now we are in LIST A[1], blue arrow; Is LIST A[1] = "bed" in the linked list? NO, so insert LIST A[1] = "bed" in the next entry LIST B[1] linked list, forming the <u>second group</u> of anagrams for "bed" and then start searching for anagrams to "bed"

Word 1 = bed

Word 2 = car

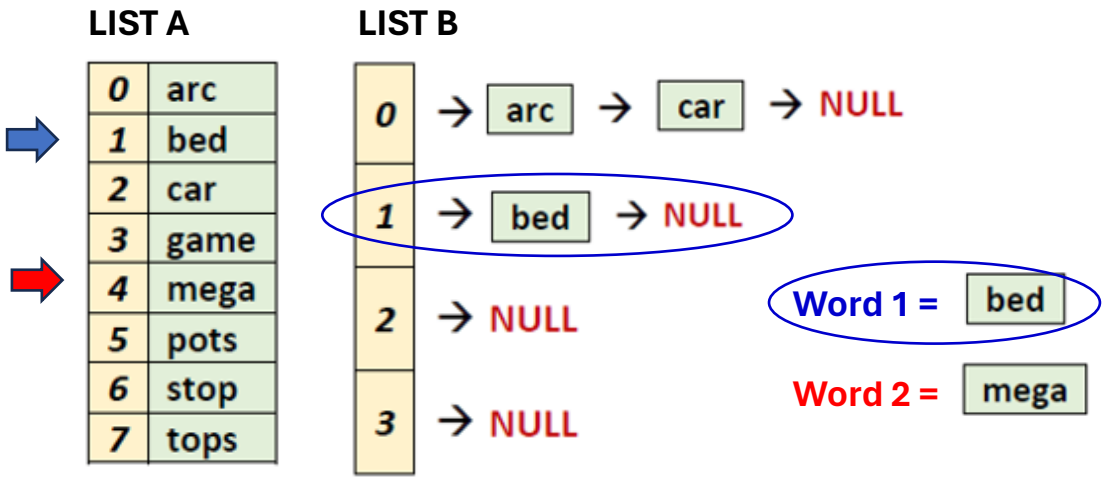
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "acr"
Is "bde" == "acr" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = game

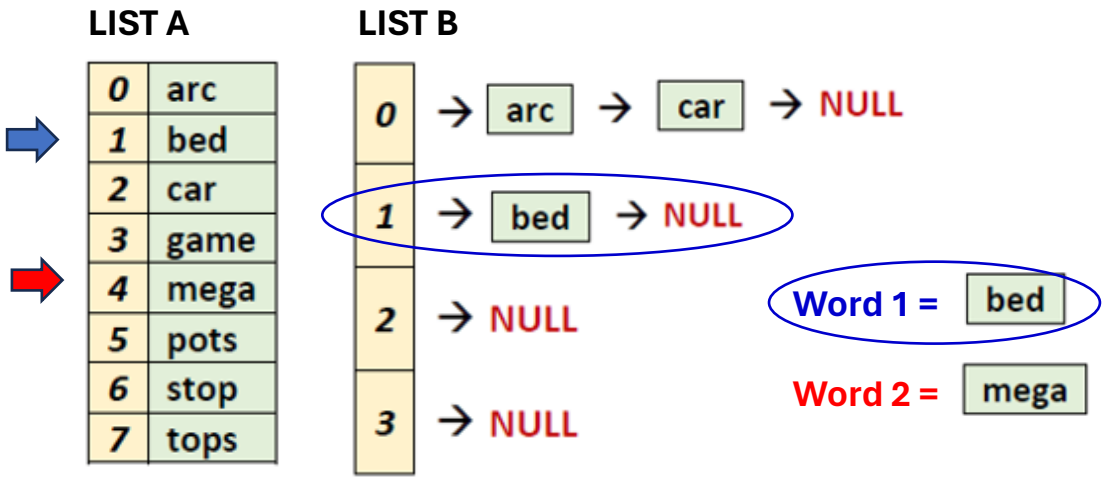
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "aegm"
Is "bde" == "aegm" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = mega

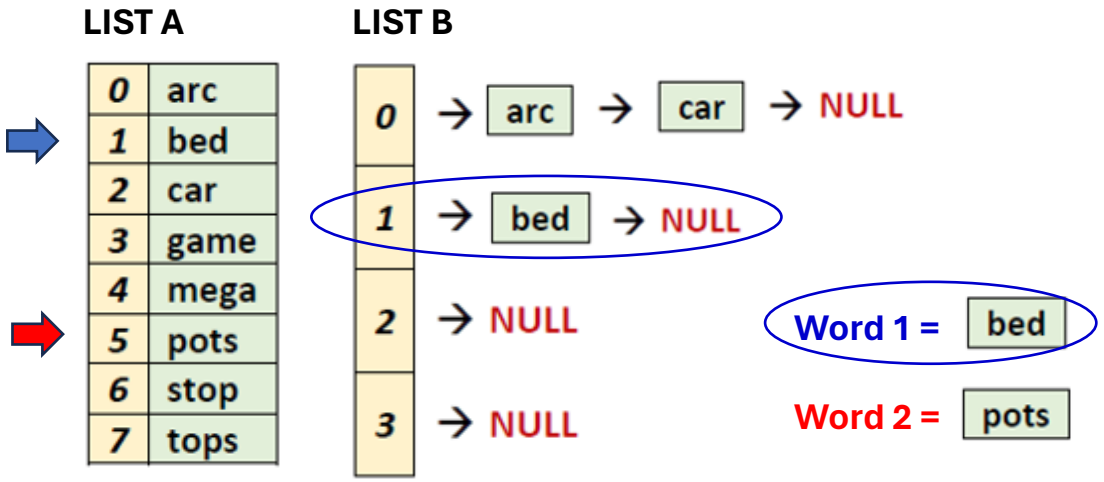
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "aegm"
Is "bde" == "aegm" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = mega

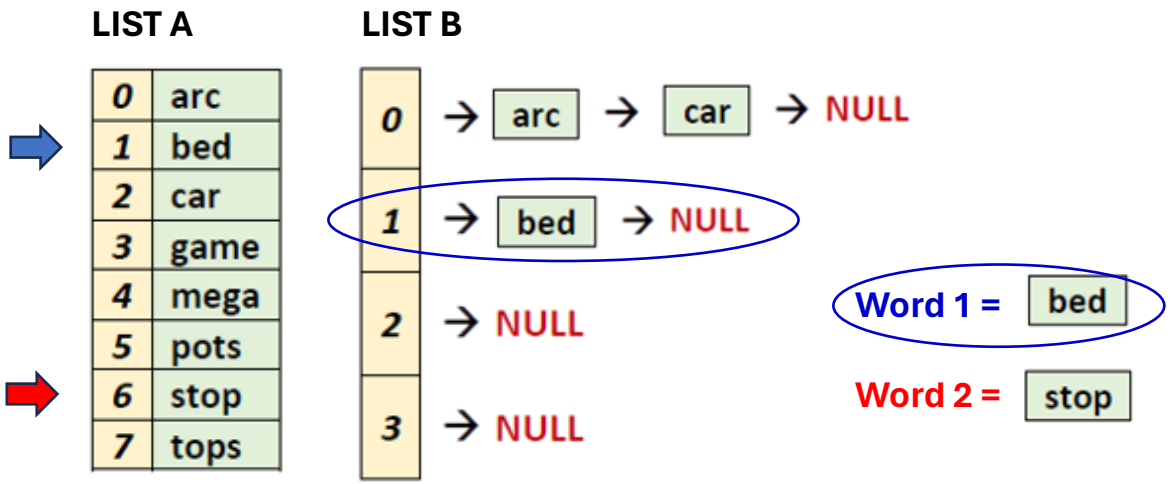
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "aegm"
Is "bde" == "aegm" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = pots

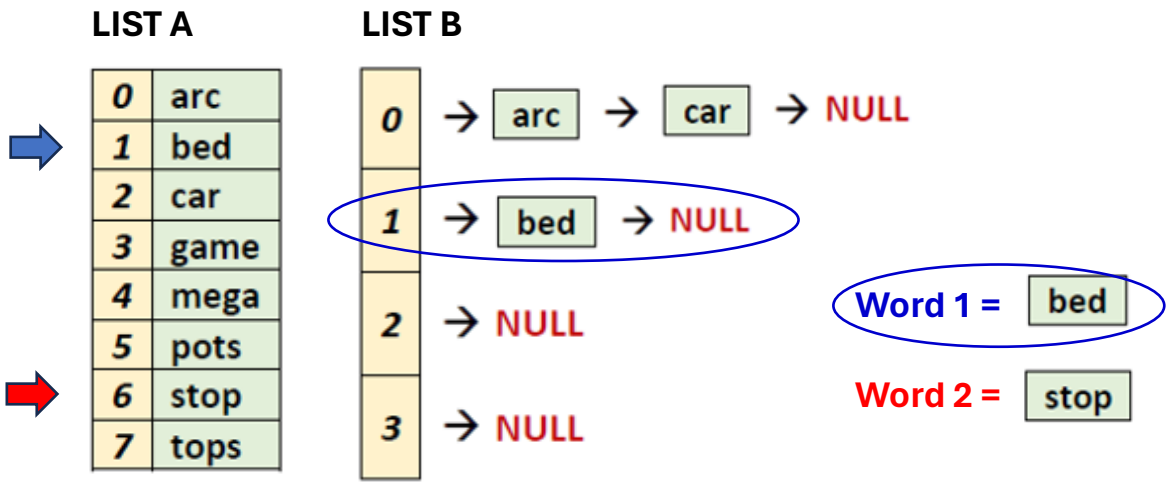
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "opst"
Is "bde" == "opst" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = stop

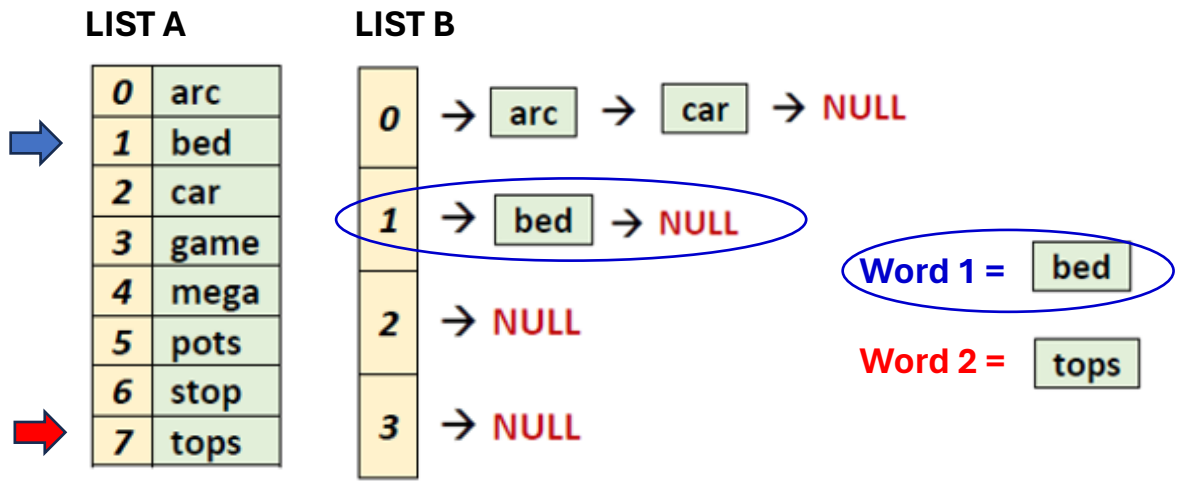
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "opst"
Is "bde" == "opst" NO, so they are NOT anagrams
}



Word 1 = bed

Word 2 = stop

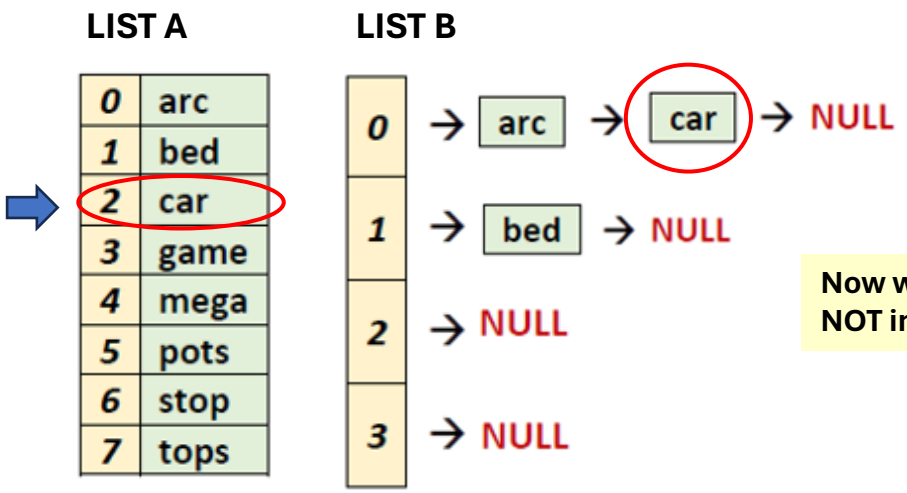
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "opst"
Is "bde" == "opst" NO, so they are NOT anagrams
}



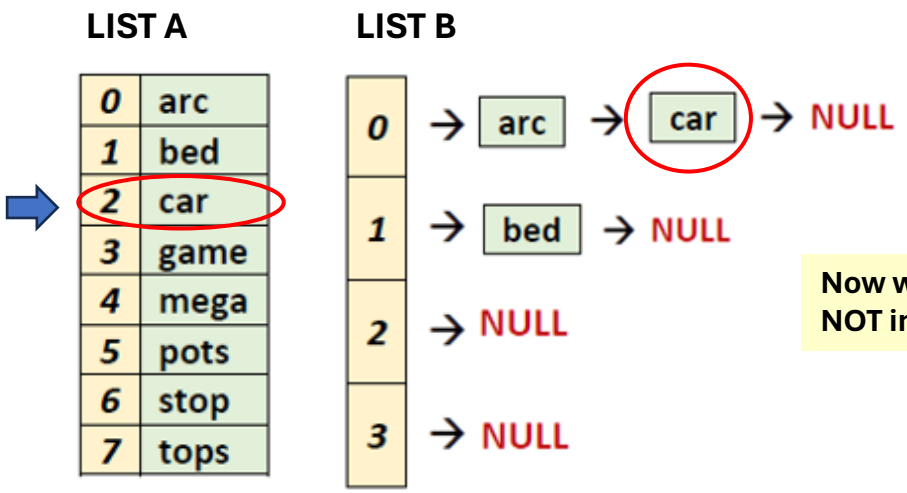
Word 1 = bed

Word 2 = tops

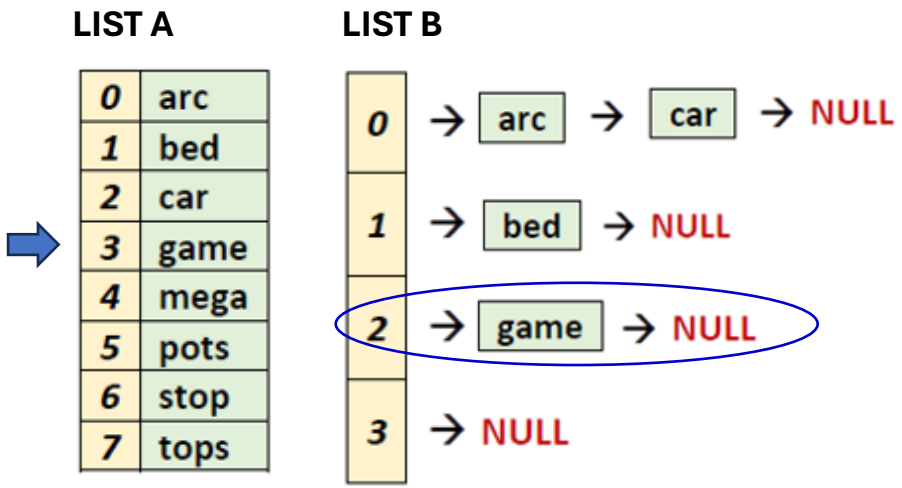
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "bde"
Sort letters in Word 2 → "opst"
Is "bde" == "opst" NO, so they are NOT anagrams
}



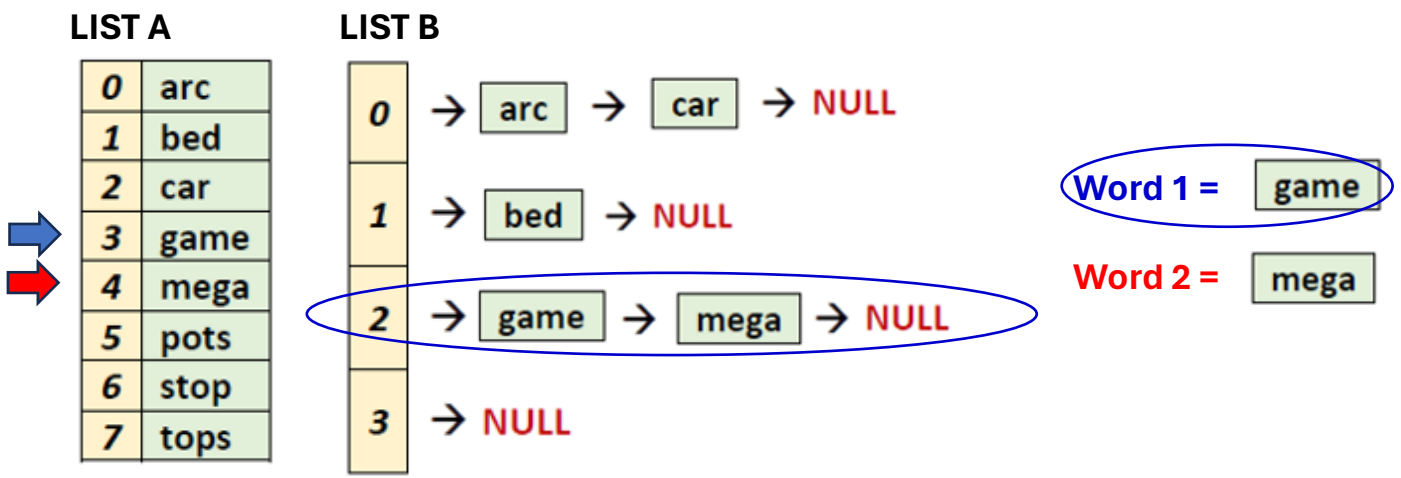
Now we are in LIST A[2], blue arrow; Is LIST A[2] = "car" in the linked list? YES, so DO NOT insert LIST A[2] = "car" in the linked list, and move to the next entry in LIST A



Now we are in LIST A[2], blue arrow; Is LIST A[2] = “car” in the linked list? YES, so DO NOT insert LIST A[2] = “car” in the linked list, and move to the next entry in LIST A



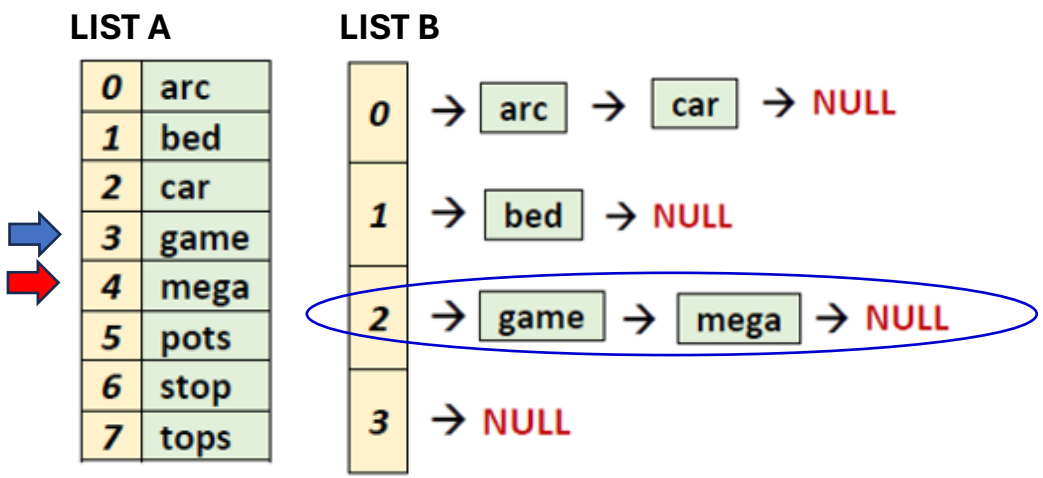
Now we are in LIST A[3], blue arrow; Is LIST A[3] = “game” in the linked list? NO, so insert LIST A[3] = “game” in the linked list at LIST B[2]



Word 1 = game

Word 2 = mega

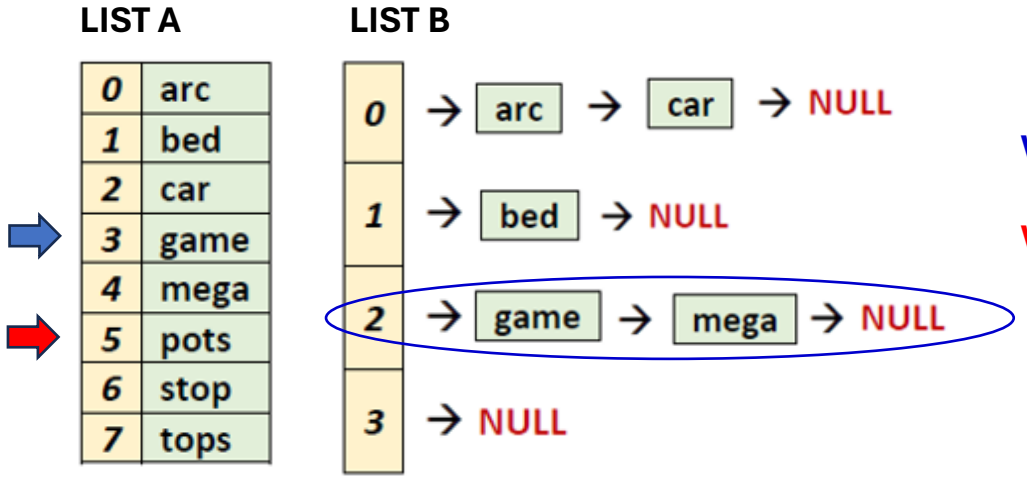
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "aegm" YES, so they are anagrams
}



Word 1 = game

Word 2 = mega

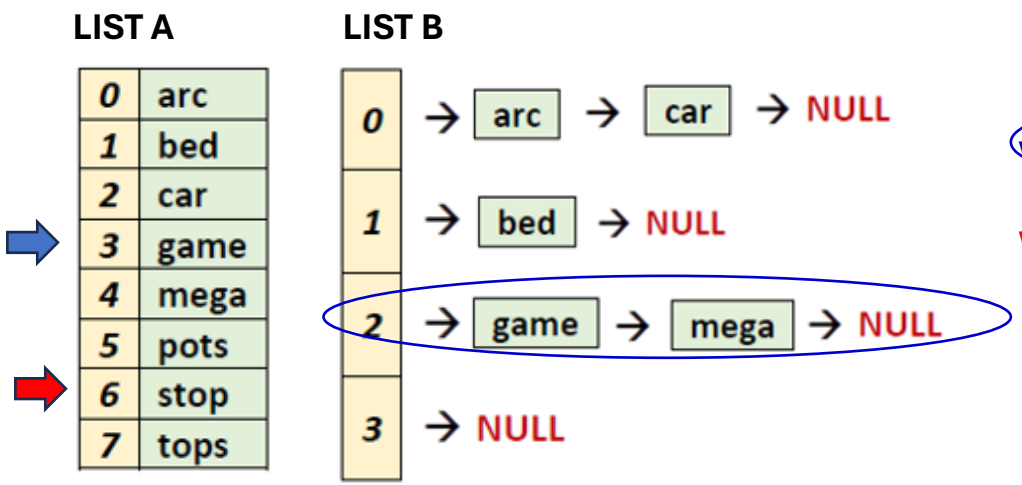
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "aegm" YES, so they are anagrams
}

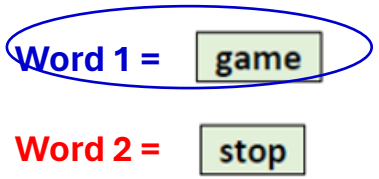


Word 1 = game

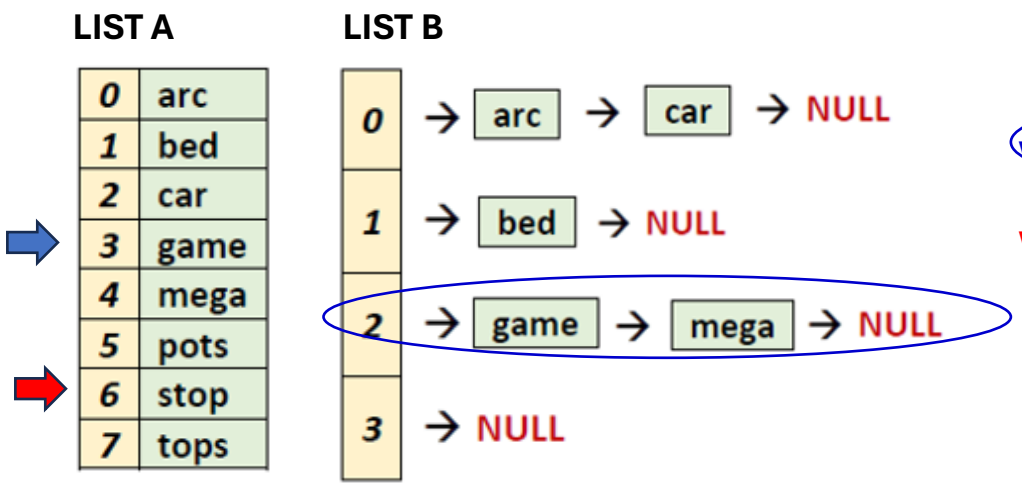
Word 2 = pots

two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "opst" NO, so they are not anagrams
}





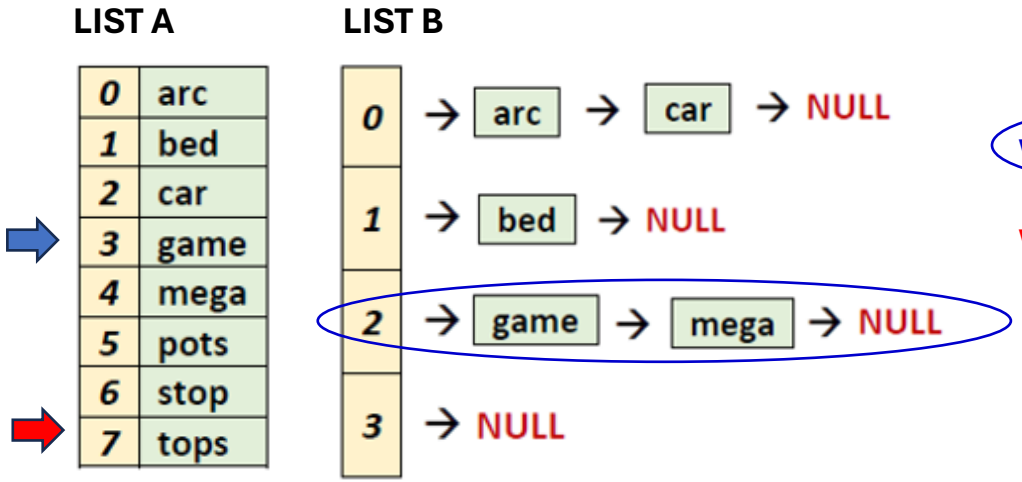
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "opst" NO, so they are not anagrams
}



Word 1 = game

Word 2 = stop

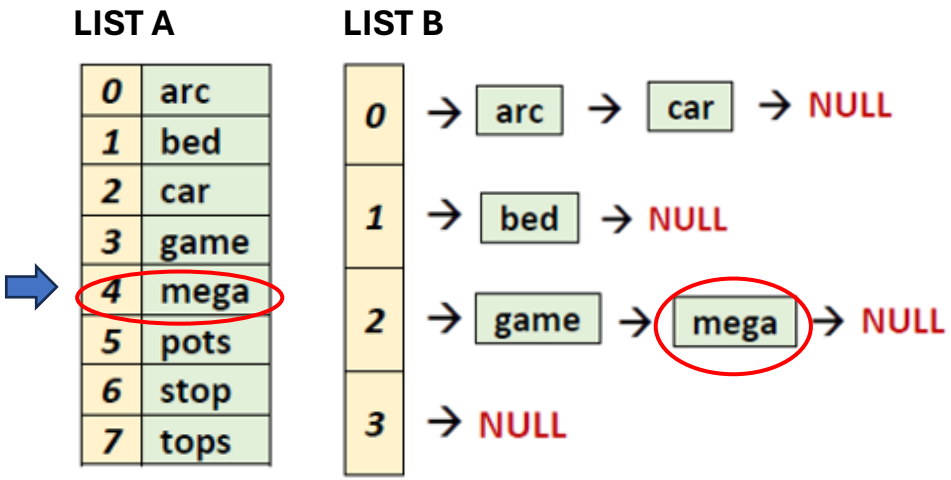
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "opst" NO, so they are not anagrams
}



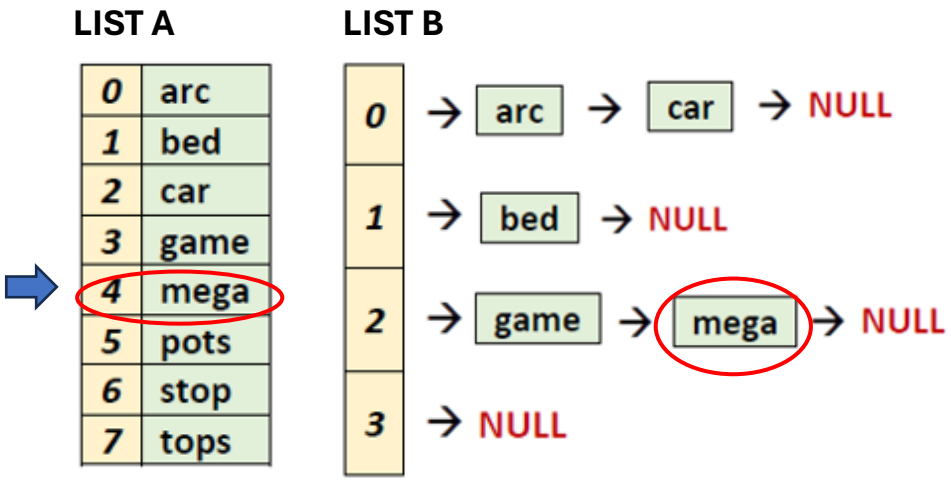
Word 1 = game

Word 2 = tops

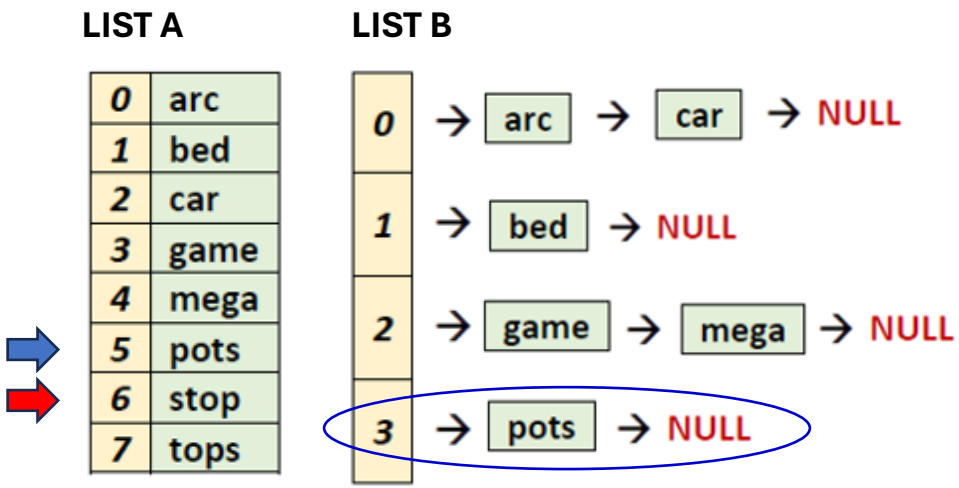
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "aegm"
Sort letters in Word 2 → "aegm"
Is "aegm" == "opst" NO, so they are not anagrams
}



Now we are in LIST A[4], blue arrow; Is LIST A[4] = "mega" in the linked list?
YES, so DO NOT insert LIST A[4] = "mega" in the linked list



Now we are in LIST A[4], blue arrow; Is LIST A[4] = "mega" in the linked list?
YES, so DO NOT insert LIST A[4] = "mega" in the linked list

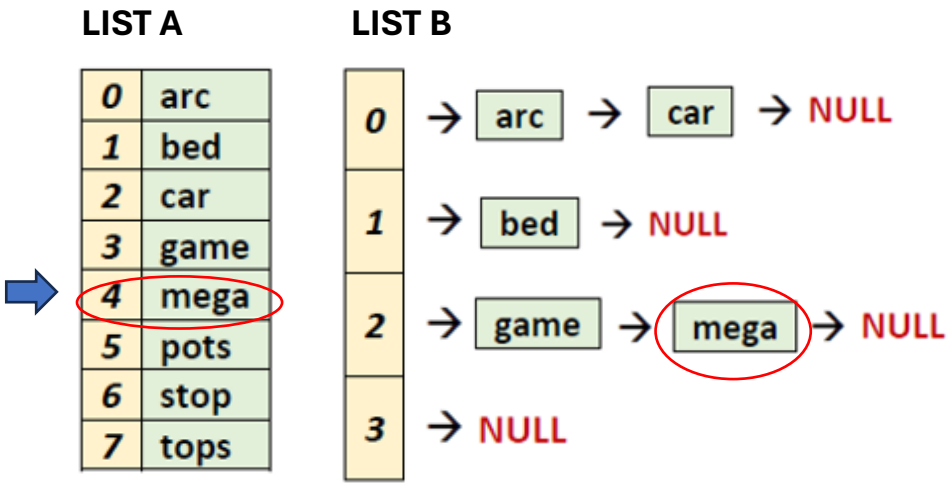


Now we are in LIST A[45], blue arrow; Is LIST A[5] = "pots" in the linked list?
NO, so insert LIST A[5] = "pots" in the linked list

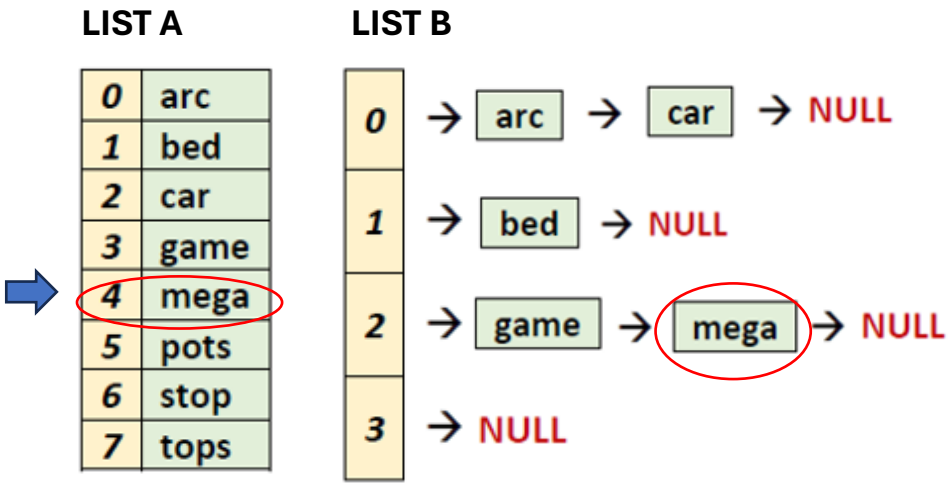
Word 1 = pots

Word 2 = stop

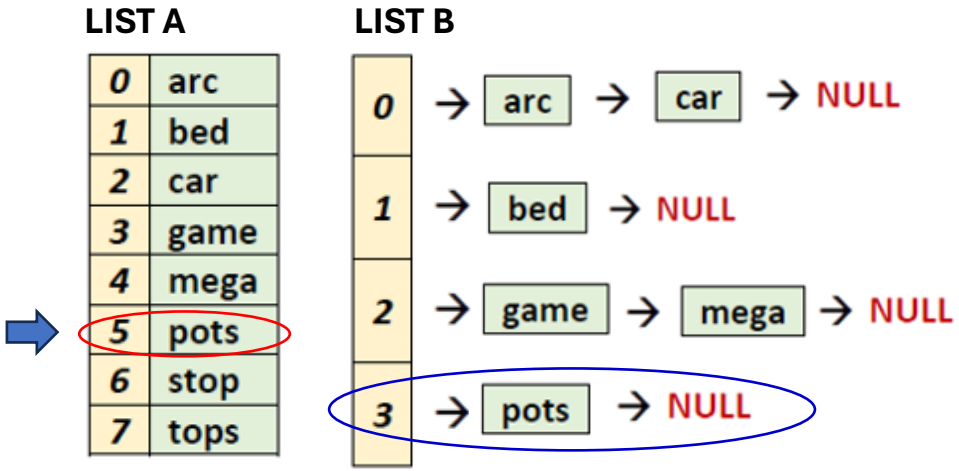
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "opst"
Sort letters in Word 2 → "opst"
Is "opst" == "opst" YES, so they are anagrams
}



Now we are in LIST A[4], blue arrow; Is LIST A[4] = "mega" in the linked list?
YES, so DO NOT insert LIST A[4] = "mega" in the linked list



Now we are in LIST A[4], blue arrow; Is LIST A[4] = "mega" in the linked list?
YES, so DO NOT insert LIST A[4] = "mega" in the linked list

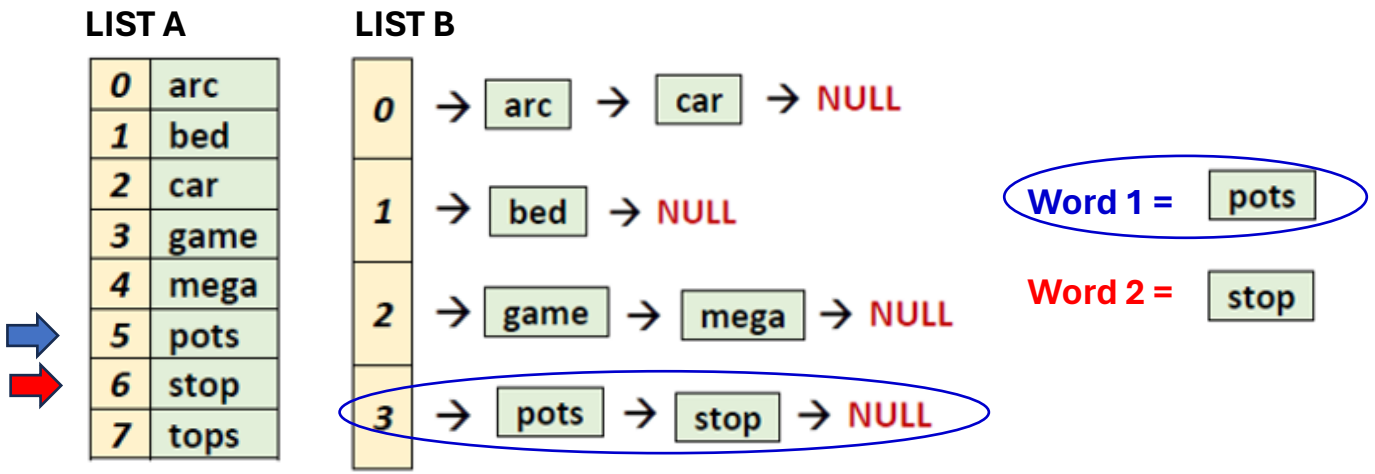


Now we are in LIST A[5], blue arrow; Is LIST A[5] = "pots" in the linked list?
NO, so insert LIST A[5] = "pots" in the linked list

Word 1 = pots

Word 2 = stop

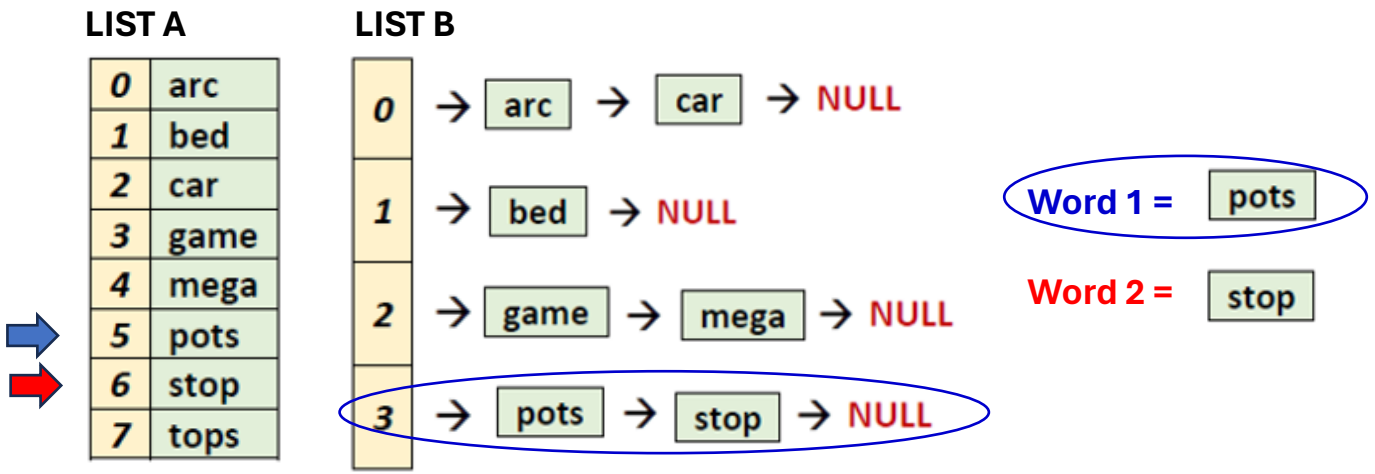
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "opst"
Sort letters in Word 2 → "opst"
Is "opst" == "opst" YES, so they are anagrams
}



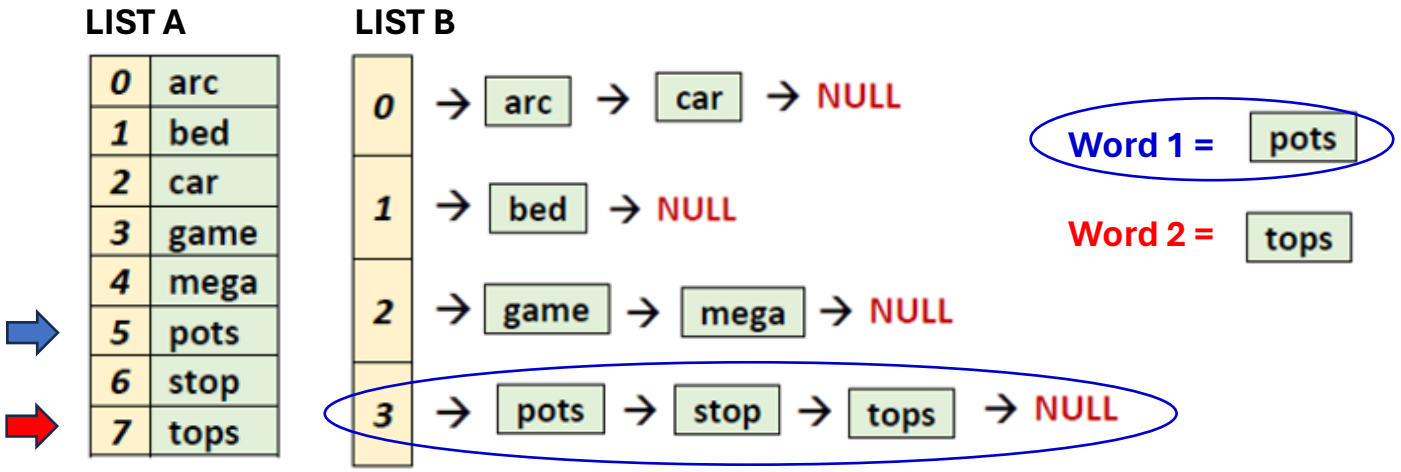
Word 1 = pots

Word 2 = stop

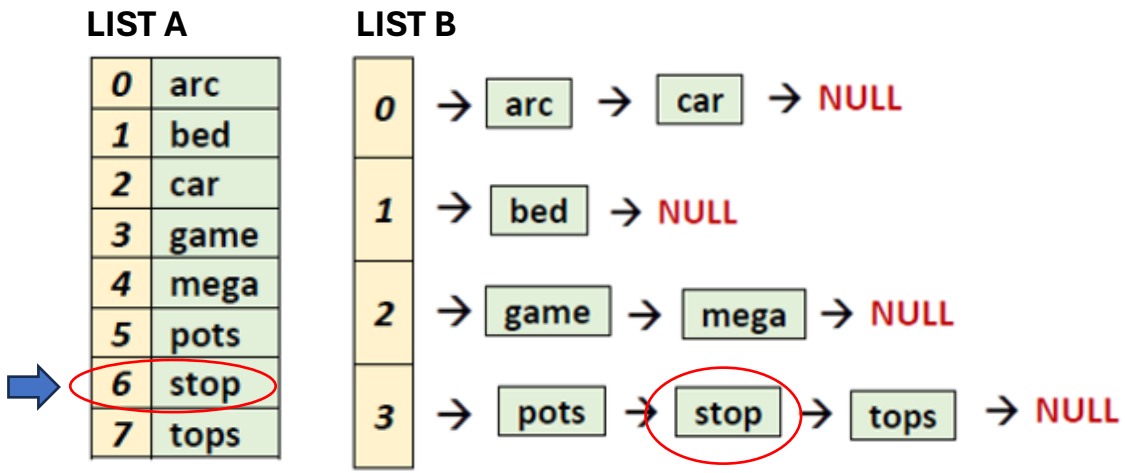
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 → "opst"
Sort letters in Word 2 → "opst"
Is "opst" == "opst" YES, so they are anagrams
}



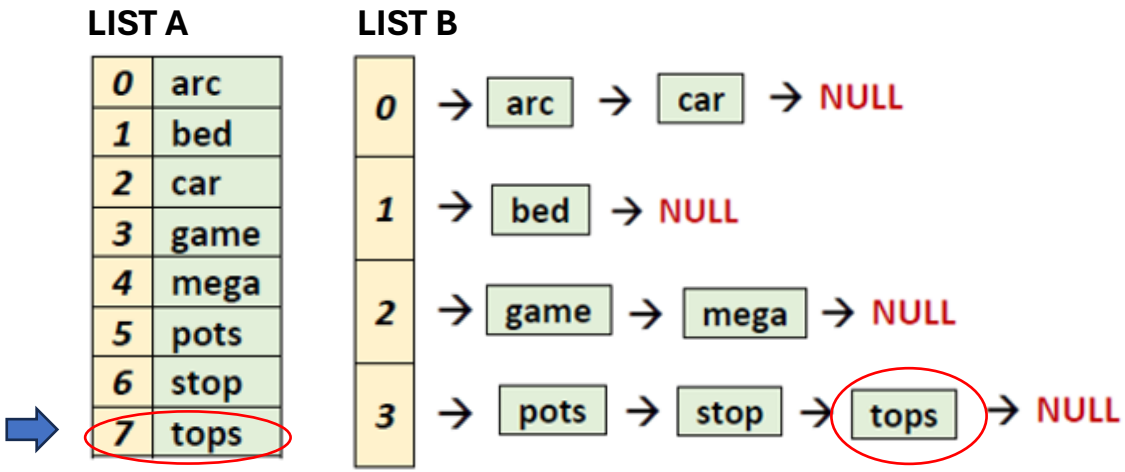
two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 →"opst"
Sort letters in Word 2 → "opst"
Is "opst" == "opst" YES, so they are anagrams
}



two_words_anagrams ? (Word 1, Word 2)
{
Sort letters in Word 1 →"opst"
Sort letters in Word 2 → "opst"
Is "opst" == "opst" YES, so they are anagrams
}



Now we are in LIST A[6], blue arrow; Is LIST A[6] = "stop" in the linked list?
YES, so DO NOT insert LIST A[6] = "stop" in the linked list



Now we are in LIST A[7], blue arrow; Is LIST A[7] = "tops" in the linked list?
YES, so DO NOT insert LIST A[7] = "tops" in the linked list

LIST A

0	arc
1	bed
2	car
3	game
4	mega
5	pots
6	stop
7	tops

LIST B

0 → arc → car → NULL

1 → bed → NULL

2 → game → mega → NULL

3 → pots → stop → tops → NULL

We are done! ✓

Step #3. A new **LIST B** (**1-D array** of **singly linked lists**) should then be created directly from **LIST A** (i.e., the sorted 1-D array of words, step #2) as follows (example):

0 → arc → car → NULL

1 → bed → NULL

2 → game → mega → NULL

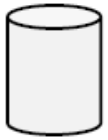
3 → pots → stop → tops → NULL

LIST B

1-D array of **singly linked lists**

Step #4. Traverse **LIST B** (**1-D array** of **singly linked lists**) to generate the required **output text file** format (refer to previous page).

arc car
bed
game mega
pots stop tops

⇨ 

Output text file

Refer to folder "PA-2--Examples-of-expected-outputs" for examples of how your final output text files should look like.