## Pair with given product | Set 1 (Find if any pair exists)

Given an array and a number x, find if there is a pair with product equal to x.

Examples :

```
Input : arr[] = {10, 20, 9, 40};
        int x = 400;
Output : Yes

Input : arr[] = {10, 20, 9, 40};
        int x = 190;
Output : No

Input : arr[] = {-10, 20, 9, -40};
        int x = 400;
Output : Yes

Input : arr[] = {-10, 20, 9, 40};
        int x = -400;
Output : Yes

Input : arr[] = {0, 20, 9, 40};
        int x = 0;
Output : Yes
```

**Naive approach ( $O(n^2)$ )** is to run two loops to consider all possible pairs. For every pair, check if product is equal to x or not.

```
// A simple C++ program to find if there is a pair
// with given product.
#include<bits/stdc++.h>
using namespace std;

// Returns true if there is a pair in arr[0..n-1]
// with product equal to x.
bool isProduct(int arr[], int n, int x)
{
    // Consider all possible pairs and check for
    // every pair.
    for (int i=0; i<n-1; i++)
       for (int j=i+1; i<n; i++)
          if (arr[i] * arr[j] == x)
              return true;

    return false;
}

// Driver code
int main()
{
    int arr[] = {10, 20, 9, 40};
    int x = 400;
    int n = sizeof(arr)/sizeof(arr[0]);
    isProduct(arr, n, x)? cout << "Yes\n"
                        : cout << "No\n";
    x = 190;
    isProduct(arr, n, x)? cout << "Yes\n"
                        : cout << "No\n";
    return 0;
}
```

Output :

```
Yes
No
```

**Efficient Solution ( O(n) ):** We can improve time complexity to O(n) using hashing. Below are steps.

1. Create an empty hash table
2. Traverse array elments and do following for every element arr[i].
    - If arr[i] is 0 and x is also 0, return true, else ignore arr[i].
    - If x % arr[i] is 0 and x/arr[i] exists in table, return true.
    - Insert arr[i] into the hash table.
3. Return false

Below is C++ implementation of above idea.

```
// C++ program to find if there is a pair
// with given product.
#include<bits/stdc++.h>
using namespace std;

// Returns true if there is a pair in arr[0..n-1]
// with product equal to x.
bool isProduct(int arr[], int n, int x)
{
    if (n < 2)
        return false;

    // Create an empty set and insert first
    // element into it
    unordered_set<int> s;

    // Traverse remaining elements
    for (int i=0; i<n; i++)
    {
        // 0 case must be handles explicitly as
        // x % 0 is undefined behaviour in C++
        if (arr[i] == 0)
        {
           if (x == 0)
               return true;
           else
               continue;
        }

        // x/arr[i] exists in hash, then we
        // found a pair
        if (x%arr[i] == 0)
        {
            if (s.find(x/arr[i]) != s.end())
               return true;

            // Insert arr[i]
            s.insert(arr[i]);
        }
    }
    return false;
}

// Driver code
int main()
{
    int arr[] = {10, 20, 9, 40};
    int x = 400;

    int n = sizeof(arr)/sizeof(arr[0]);
    isProduct(arr, n, x)? cout << "Yes\n"
                        : cout << "No\n";

    x = 190;
    isProduct(arr, n, x)? cout << "Yes\n"
                         : cout << "No\n";

    return 0;
}
```

Output :

```
Yes
No
```

In the next set, we will be discussing approach to print all pairs with product equal to 0.