DAY 12:

SESSION 1:

Vendor Maximum Profit

ID:11132 Solved By 667 Users

A vendor has a shop and he wants to purchase some items for a maximum of **N** rupees. There are **K** items in a wholesale store that the vendor is going to buy. Each item has the cost price and profit that the seller can sell. The vendor can buy multiple items but not the same type. The program must accept the value of N and the cost price and profit of the K items as the input. The program must print the maximum profit that the vendor can earn by buying and selling the items as the output.

Boundary Condition(s):
1 <= N <= 1000
1 <= K <= 100
1 <= Cost price and profit of each item <= 1000

Input Format:
The first line contains N and K separated by a space.
The next K lines, each containing two integers representing the cost price and profit of an item.

Output Format:
The first line contains the maximum profit that the vendor can earn by buying and selling the items as per the given conditions.

Example Input/Output 1:
Input:
10 6
5 2
6 4
3 2
4 3
1 2
15 20

Output:
8

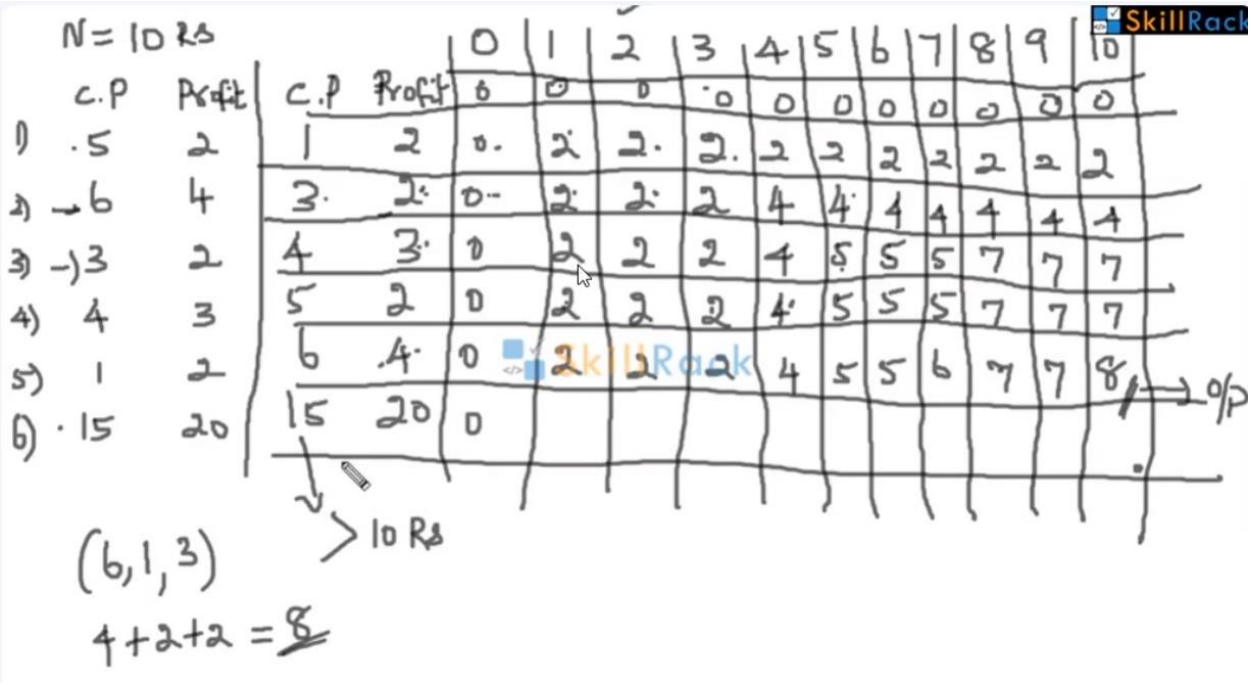
Explanation:
Here N = **10** and K = **6**.
The maximum profit that the vendor can earn by buying and selling the following 3 items is **8**.
6 4
1 2
3 2

Example Input/Output 2:
Input:
20 6
5 2
6 4
3 2
4 3
1 2
15 20

Output:
25

Max Execution Time Limit: 500 millisecs

Logic:



Code:

```
import java.util.*;

class Item implements Comparable<Item> {
    int costPrice, profit;

    @Override
    public int compareTo(Item other) {
        return this.costPrice - other.costPrice;
    }
}

public class vendorMaximumProfit {
    public static void main(String[] args) {
        // Your Code Here
        Scanner s = new Scanner(System.in);
        int N = s.nextInt();
        int K = s.nextInt();
        // int costPrice, profit;
        List<Item> items = new ArrayList<>();
        items.add(new Item()); // dummy item added to avoid index zero
        for (int index = 0; index < K; index++) {
            Item item = new Item();
```

```
            item.costPrice = s.nextInt();
            item.profit = s.nextInt();
            items.add(item);
        }
        Collections.sort(items);
        int matrix[][] = new int[K + 1][N + 1]; // profit matrix
        for (int item = 1; item <= K; item++) { // starting from index 1 as 0 is
the dummy item
            Item currentItem = items.get(item);
            if (currentItem.costPrice > N) { // optimisation - check the logic
photo 15>10 rs..
                System.out.println(matrix[item - 1][N]);
                return;
            }
            for (int amount = 1; amount <= N; amount++) {
                if (amount < currentItem.costPrice) {
                    matrix[item][amount] = matrix[item - 1][amount]; // previous
row value is retained
                } else {
                    int includedProfit = currentItem.profit + matrix[item -
1][amount - currentItem.costPrice];
                    int excludedProfit = matrix[item - 1][amount];
                    matrix[item][amount] = Math.max(includedProfit,
excludedProfit);
                }
            }
        }
        System.out.println(matrix[K][N]);
    }
}
```

Session 2:

Largest Square Sub Matrix with 1s

ID:11134 Solved By 712 Users

The program must accept an integer matrix of size **RxC** containing only **0s** and **1s** as the input. The program must print the size of the largest square sub matrix containing all the elements as 1 in the given matrix.
Note: There is always at least one square sub matrix containing all the elements as 1 in the given matrix.

Boundary Condition(s):
2 <= R, C <= 1000

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.

Output Format:
The lines containing the largest square sub matrix containing all the elements as 1 in the given matrix.

Example Input/Output 1:
Input:
7 5
1 1 1 0 1
1 1 0 1 0
0 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 0 0 0 0

Output:
4

Explanation:
In the given 7x5 matrix, the largest square sub matrix with 1s is highlighted below.
1 1 1 0 1
1 1 0 1 0
0 **1 1 1 1**
1 **1 1 1 1**
1 **1 1 1 1**
1 **1 1 1 1**
0 0 0 0 0
The size of the largest square sub matrix is 4.
Hence the output is 4

Example Input/Output 2:
Input:
7 6
1 1 0 1 1 1
1 1 0 1 1 1
0 0 0 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
1 1 0 0 1 1
1 1 0 0 1 1

Output:
3

Max Execution Time Limit: 500 millisecs

Code:

```
import java.util.*;
public class largestSquareSubMatrix {

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        int R=in.nextInt();
        int C=in.nextInt();
        int matrix[][] = new int[R][C];
        int parallelMatrix[][] = new int[R][C];
        for(int row=0;row<R;row++){
            for(int col=0;col<C;col++){
                matrix[row][col]=in.nextInt();
                if(row==0 || col==0){
                    parallelMatrix[row][col]=matrix[row][col];
                }
            }
        }
        //commented line alternate code for this problem
        // for(int row=0;row<R;row++){
        //     parallelMatrix[row][0]=matrix[row][0];
        // }

        // for(int col=0;col<C;col++){
        //     parallelMatrix[0][col]=matrix[0][col];
        // }
        int maxValue=0;

        for(int row=1;row<R;row++){
            for(int col=1;col<C;col++){
                if(matrix[row][col]!=0){
                    int left = parallelMatrix[row][col-1];  //checking for left
                    int top = parallelMatrix[row-1][col];   //checking for top
                    int topLeft = parallelMatrix[row-1][col-1]; //checking for
topLeft

                    //parallelMatrix[row][col]=matrix[row][col]+Math.min(Math.min
(parallelMatrix[row-1][col],parallelMatrix[row][col-1]),parallelMatrix[row-
1][col-1]);
                    //matrix[row][col]=1 will be one so it can be directly coded
as 1
```

```
                    parallelMatrix[row][col]=matrix[row][col]+Math.min(Math.min(t
op,topLeft),left);

                    if(parallelMatrix[row][col]>maxValue){
                        maxValue=parallelMatrix[row][col];
                    }
                }
            }
        }
        System.out.println(maxValue);
    }
}
```

Session 3:

Word Break

ID:11133 Solved By 666 Users

The program must accept two string values **S1** and **S2** as the input. The string S1 contains a list of words separated by a space. The string S2 contains the words from the string S1 without any space. The program must break the string S2 into the words as in S1 and print all the possible word breaks as shown in the Example Input/Output section.

Boundary Condition(s):
1 <= Length of S1, S2 <= 1000

Input Format:
The first line contains the string S1.
The second line contains the string S2.

Output Format:
The lines contain all the possible word breaks in S2 as shown in the Example Input/Output section.

Example Input/Output 1:
Input:
hot box hotbox
hotboxhotbox

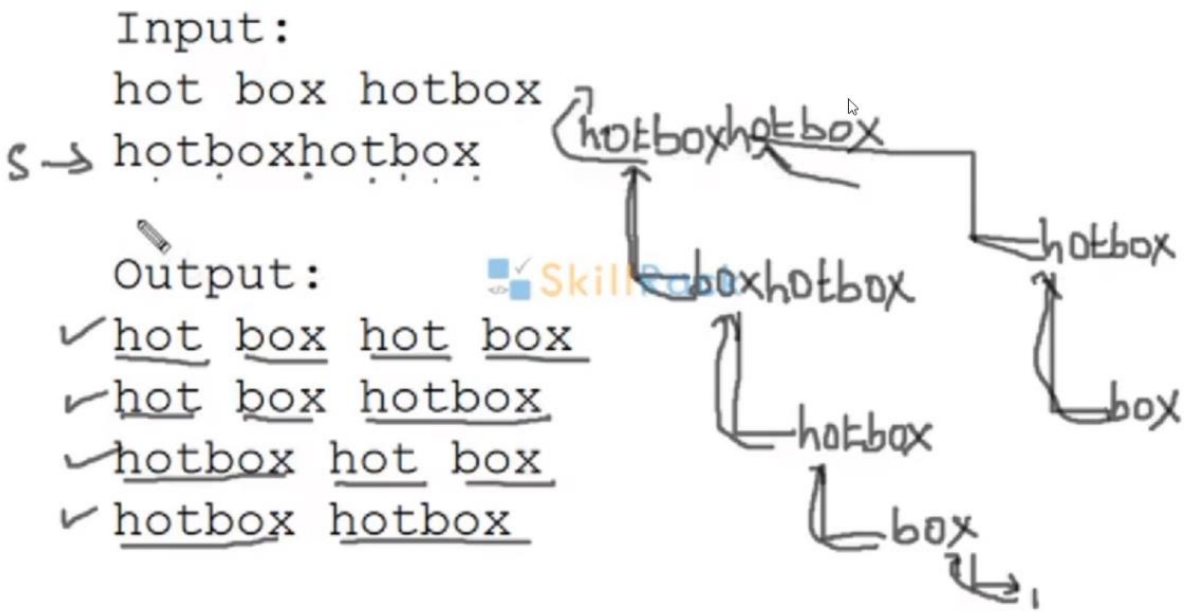
Output:
hot box hot box
hot box hotbox
hotbox hot box
hotbox hotbox

Example Input/Output 2:
Input:
t h i s th hi is thi his this
this

Output:
t h i s
t h is
t hi s
t his
th i s
th is
thi s
this

Max Execution Time Limit: 1000 millisecs

Logic:



Stimulation:

https://cscircles.cemc.uwaterloo.ca/java_visualize/#code=import+java.util.*%3B%0A%0Apublic+class+wordBreak+%7B%0A%0A++++public+static+void+main(String%5B%5D+args)+%7B%0A++++++++Scanner+in+%3D+new+Scanner(System.in)%3B%0A++++++++//+String+words%5B%5D+%3D+in.nextLine().split(%22%5C%5Cs%2B%22)%3B%0A++++++++String+words%5B%5D+%3D+%7B%22hot%22,%22box%22,%22hotbox%22%7D%3B%0A++++++++String+str+%3D+%22hotboxhotbox%22%3B%0A++++++++List%3CString%3E+wordsList+%3D+Arrays.asList(words)%3B%0A++++++++breakWord(wordsList,+str,+%22%22,+0)%3B%0A++++%7D%0A%0A++++public+static+void+breakWord(List%3CString%3E+wordsList,+String+str,+String+output,+int+start)+%7B%0A++++++++if+(start+%3D%3D+str.length())+%7B%0A++++++++++++System.out.println(output.trim())%3B%0A++++++++++++return%3B%0A++++++++%7D%0A++++++++for+(int+index+%3D+start%3B+index+%3C+str.length()%3B+index%2B%2B)+%7B%0A++++++++++++String+word+%3D+str.substring(start,+index+%2B+1)%3B+//in+substring+the+last+argument+is+exclusive%0A++++++++++if+(wordsList.contains(word))+%7B%0A++++++++++++++++breakWord(wordsList,+str,+output+%2B+word+%2B+%22+%22,+index+%2B+1)%3B%0A++++++++++++%7D%0A++++++++%7D%0A++++%7D%0A%7D%0A&mode=display&curInstr=228

Code:

```
import java.util.*;

public class wordBreak {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String words[] = in.nextLine().split("\\s+"); //splitting with spaces
        String str = in.nextLine();
        List<String> wordsList = Arrays.asList(words);
        breakWord(wordsList, str, "", 0);
    }

    public static void breakWord(List<String> wordsList, String str, String
output, int start) {
        if (start == str.length()) { //base condtion for recursion //to print the
string
            System.out.println(output.trim());
            return;
        }
        for (int index = start; index < str.length(); index++) {
            String word = str.substring(start, index + 1); // in substring the
last argument is exclusive
            if (wordsList.contains(word)) {
                breakWord(wordsList, str, output + word + " ", index + 1);
            }
        }
    }
}
```