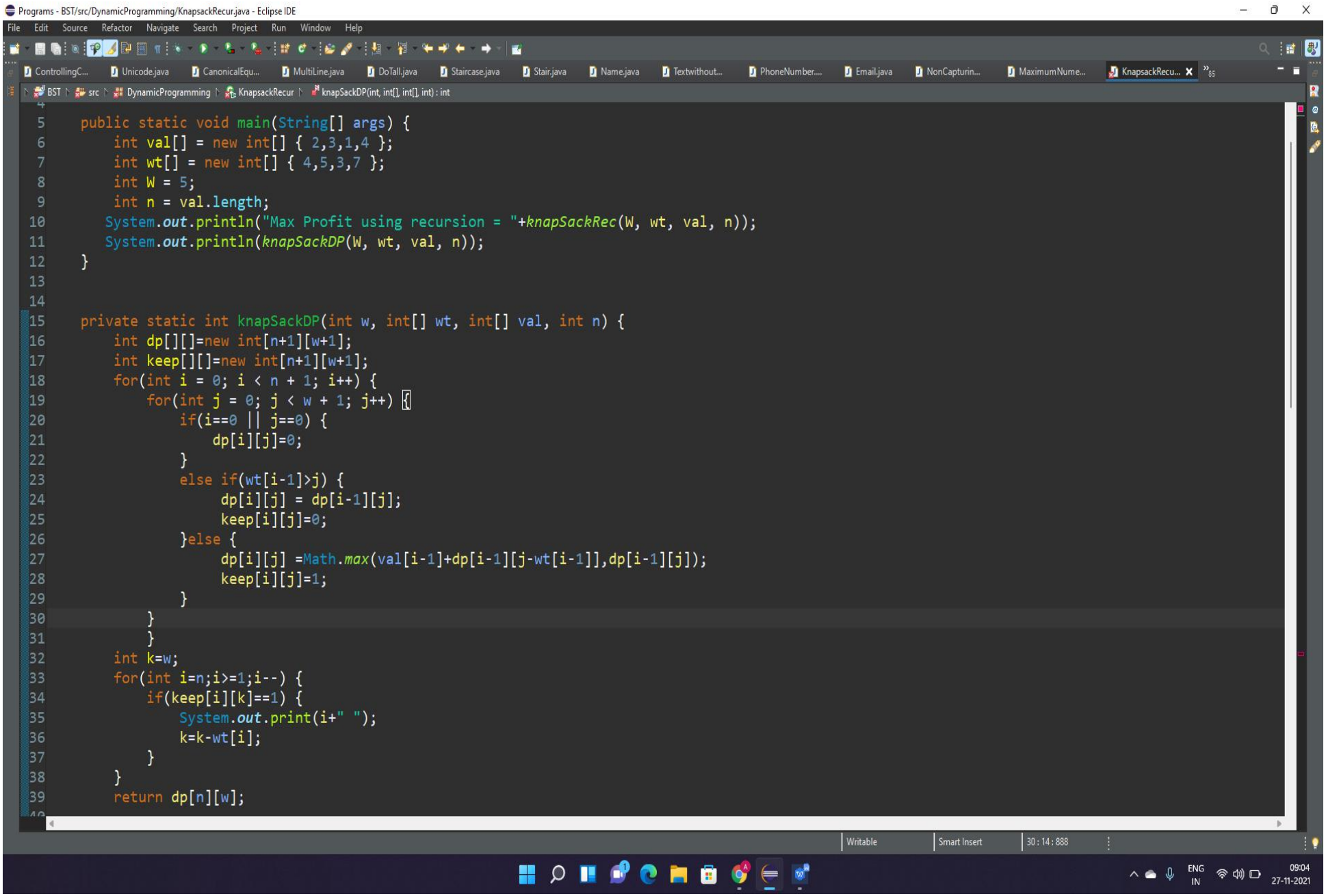
Programs - BST/src/DynamicProgramming/KnapsackRecur.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
ControllingC... Unicode.java CanonicalEqu... MultiLine.java DoTall.java Staircase.java Stair.java Name.java Textwithout... PhoneNumber.... Email.java NonCapturin... MaximumNume... KnapsackRecu...
BST src DynamicProgramming KnapsackRecur knapSackDP(int, int[], int[], int) : int
5 public static void main(String[] args) {
6 int val[] = new int[] { 2,3,1,4 };
7 int wt[] = new int[] { 4,5,3,7 };
8 int W = 5;
9 int n = val.length;
10 System.out.println("Max Profit using recursion = "+knapSackRec(W, wt, val, n));
11 System.out.println(knapSackDP(W, wt, val, n));
12 }
13
14
15 private static int knapSackDP(int w, int[] wt, int[] val, int n) {
16 int dp[][]=new int[n+1][w+1];
17 int keep[][]=new int[n+1][w+1];
18 for(int i = 0; i < n + 1; i++) {
19 for(int j = 0; j < w + 1; j++) {
20 if(i==0 || j==0) {
21 dp[i][j]=0;
22 }
23 else if(wt[i-1]>j) {
24 dp[i][j] = dp[i-1][j];
25 keep[i][j]=0;
26 }else {
27 dp[i][j] =Math.max(val[i-1]+dp[i-1][j-wt[i-1]],dp[i-1][j]);
28 keep[i][j]=1;
29 }
30 }
31 }
32 int k=w;
33 for(int i=n;i>=1;i--) {
34 if(keep[i][k]==1) {
35 System.out.print(i+" ");
36 k=k-wt[i];
37 }
38 }
39 return dp[n][w];
Writable Smart Insert 30 : 14 : 888
ENG IN 09:04 27-11-2021