```
::::::::::::::
src/Driver.java
::::::::::::::
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * Brian Chesko
 * DAA Programming Assignment 2
 * Fully complete and tested, 2019/02/27
 */
public class Driver {

    public static void main(String[] args) throws IOException {
        System.out.println("Enter matrix size");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in
));
        int n = Integer.parseInt(reader.readLine().trim());
        System.out.println(n);
        short[][] matrix = new short[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = (short) Integer.parseInt(reader.readLine().trim());
                System.out.printf("%d\t", matrix[i][j]);
            }
            System.out.println();
        }

        //JobEmployeeMatrix inputMatrix = new JobEmployeeMatrix(matrix);
        Solver solver = new Solver(matrix);

        int[] sol = solver.solve();
        System.out.printf("Number of job assignments explored: %d\n", solver.getEx
ploredSize());
        System.out.println("Best job assignment is:");
        for (int i = 0; i < sol.length; i++) {
            System.out.printf("Person %d assigned job %d\n", i, sol[i]);
        }
        System.out.printf("Best job assignment cost: %d\n", solver.getSolutionProd
uctivity());
    }
}
::::::::::::::
src/Solver.java
::::::::::::::
import java.util.Arrays;

/**
 * Brian Chesko
 * DAA Programming Assignment 2
 * Fully complete and tested, 2019/03/04
 */
public class Solver {
    private short[][] matrix;
    private int size;
    private int highestProductivity;
    private long solutionsExplored;
    private int[] bestArrangement;

    public Solver(short[][] jobEmployeeMatrix) {
        this.matrix = jobEmployeeMatrix;
        this.size = matrix.length;
        this.highestProductivity = 0;
        this.solutionsExplored = 0;
        this.bestArrangement = null;
    }

    /**
     * Reinitialize the solver to work with the specified matrix.
     * @param jobEmployeeMatrix The new matrix to solve.
     */
    public void setJobEmployeeMatrix(short[][] jobEmployeeMatrix) {
        this.matrix = jobEmployeeMatrix;
        this.size = matrix.length;
        this.highestProductivity = 0;
        this.solutionsExplored = 0;
        this.bestArrangement = null;
    }

    /**
     * Finds the most productive set of jobs assignments given the job
     * employee matrix.
     * @return an array containing the most productive arrangement, such that
     * array[i] is the best job for employee i.
     */
    public int[] solve() {
        // Already solved for this matrix, return previous solution.
        if (bestArrangement != null)
            return bestArrangement;

        bestArrangement = new int[size];
        int[] arrangement = new int[size];
        int[] partialProductivities = new int[size];
        int[] maxSubtreeProductivities = new int[size];
        boolean[] columnUsed = new boolean[size];
        // Ensure prefill arrangement to all unused
        for (int i = 0; i < size; i++) {
            arrangement[i] = -1;
            // Do preprocessing to determine largest productivity of each
            // size subtree.  This lets us prune subtree searches that cannot
            // possibly beat a current max partial assignment.
            int largestVal = Integer.MIN_VALUE;
            for (int val : matrix[i]) {
                if (val > largestVal) {
                    largestVal = val;
                }
            }
            for (int j = 0; j <= i; j++) {
                maxSubtreeProductivities[j] += largestVal;
            }

        }

        int emp = 0;
        int job;

        while (emp < size && emp >= 0) {
            int prevJob = arrangement[emp];
            if (prevJob != -1) {
                // Don't check the same index twice for the same employee,
                // move to the next job.
                job = prevJob + 1;
```

```
                // Reset variables tracking the previous setup.
                columnUsed[prevJob] = false;
                arrangement[emp] = -1;
            } else {
                // Haven't seen this employee before (for this subtree)
                // so start from beginning of job search.
                job = 0;
            }
            boolean foundCol = false;

            // If the partial solution + the largest combination after this is les
s
            // than the highest seen, we can't possibly beat it. Skip that subtree
.
            int prodSoFar = emp == 0 ? 0 : partialProductivities[emp - 1];
            if (prodSoFar + maxSubtreeProductivities[emp] > highestProductivity) {
                while (job < size && !foundCol) {
                    if (!columnUsed[job]) {
                        foundCol = true;
                        columnUsed[job] = true;
                        arrangement[emp] = job;
                        partialProductivities[emp] = prodSoFar + matrix[emp][job];
                    } else {
                        job++;
                    }
                }
            }

            // We ALWAYS backtrack after the emp == size - 1 iteration.
            // We also ALWAYS backtrack after job == size - 1, but ONLY
            // after visiting children trees (if necessary).
            if (emp == size - 1) {
                this.solutionsExplored++;
                int partialProductivity = partialProductivities[emp];
                if (foundCol) {
                    // Save new best solution, if needed
                    if (partialProductivity > highestProductivity) {
                        highestProductivity = partialProductivity;
                        for (int i = 0; i < size; i++) {
                            bestArrangement[i] = arrangement[i];
                        }
                    }
                    // Reset tracking variables for this setup so we can search mo
re
                    columnUsed[arrangement[emp]] = false;
                    arrangement[emp] = -1;
                }

                // Reset tracking variables for this position
                emp--;
            } else if (!foundCol) {
                // Not the last employee, but still need to backtrack.
                emp--;
            } else {
                emp++;
            }
        }

        return bestArrangement;
    }

    /**
     * @return the best job assignment for the current matrix.
     */
    public int[] getBestArrangement() {
        return bestArrangement;
    }

    /**
     * @return the total number of solutions explored while finding the best
     * assignment
     */
    public long getExploredSize() {
        return solutionsExplored;
    }

    /**
     * @return the overall productivity of the best assignment
     */
    public int getSolutionProductivity() {
        return highestProductivity;
    }

}
```

```
::::::::::::::
testing/all_tests
::::::::::::::
::::::::::::::
output1
::::::::::::::
Enter matrix size
7
35      10      15      38      16      22      25
2       36      22      7       19      2       8
10      21      8       26      21      12      39
26      32      6       15      29      32      26
35      7       10      30      17      17      21
34      0       38      28      36      21      28
7       15      36      9       36      4       35
Number of job assignments explored: 31
Best job assignment is:
Person 0 assigned job 3
Person 1 assigned job 1
Person 2 assigned job 6
Person 3 assigned job 5
Person 4 assigned job 0
Person 5 assigned job 2
Person 6 assigned job 4
Best job assignment cost: 254
::::::::::::::
time1
::::::::::::::

real    0m0.088s
user    0m0.086s
sys     0m0.016s
::::::::::::::
output2
::::::::::::::
Enter matrix size
3
9       7       9
7       5       0
6       5       0
```

```
Number of job assignments explored: 6
Best job assignment is:
Person 0 assigned job 2
Person 1 assigned job 0
Person 2 assigned job 1
Best job assignment cost: 21
::::::::::::::
time2
::::::::::::::

real    0m0.111s
user    0m0.093s
sys     0m0.019s
::::::::::::::
output3
::::::::::::::
Enter matrix size
12
376     826     969     461     59      834     550     81      9       172     70
4       71
475     344     62      366     631     992     283     389     372     718     85
6       22
332     826     716     642     855     436     258     583     262     227     44
673
62      946     106     423     119     810     494     946     506     908     31
2       137
900     947     526     272     665     735     294     350     913     362     34
4       768
150     602     352     412     830     748     437     244     695     543     66
7       166
353     513     112     211     421     776     348     674     724     227     29
8       741
962     945     91      875     659     787     996     810     390     700     57
4       572
448     12      816     143     555     483     661     261     996     773     47
2       769
902     173     443     978     752     742     719     714     687     163     94
1       698
950     937     508     692     989     83      264     438     95      80      93
3       2
915     595     615     911     720     88      681     622     613     476     60
0       365
Number of job assignments explored: 4376
Best job assignment is:
Person 0 assigned job 2
Person 1 assigned job 5
Person 2 assigned job 1
Person 3 assigned job 7
Person 4 assigned job 8
Person 5 assigned job 4
Person 6 assigned job 11
Person 7 assigned job 6
Person 8 assigned job 9
Person 9 assigned job 3
Person 10 assigned job 10
Person 11 assigned job 0
Best job assignment cost: 10812
::::::::::::::
time3
::::::::::::::

real    0m0.139s
user    0m0.128s
sys     0m0.027s
::::::::::::::
output4
::::::::::::::
Enter matrix size
10
41      678     642     857     189     533     928     492     480     94
28      83      631     788     489     520     709     697     150     319
617     66      815     643     435     702     104     29      800     556
517     841     235     160     50      424     693     978     268     173
73      296     608     704     436     97      576     146     794     726
817     763     144     633     406     932     687     862     961     487
419     478     680     654     638     731     430     683     61      699
856     134     995     816     190     432     913     766     930     59
845     747     822     989     732     581     921     772     443     234
259     862     713     940     868     703     671     299     387     732
Number of job assignments explored: 1398
Best job assignment is:
Person 0 assigned job 6
Person 1 assigned job 3
Person 2 assigned job 8
Person 3 assigned job 7
Person 4 assigned job 9
Person 5 assigned job 5
Person 6 assigned job 4
Person 7 assigned job 2
Person 8 assigned job 0
Person 9 assigned job 1
Best job assignment cost: 8492
::::::::::::::
time4
::::::::::::::

real    0m0.094s
user    0m0.075s
sys     0m0.023s
::::::::::::::
output5
::::::::::::::
Enter matrix size
11
225     3057    9142    433     5757    7741    1672    9780    4533    1512    71
33
9896    1381    6868    8280    251     1772    7739    9789    3723    5267    44
90
293     428     6332    5247    4461    799     904     2309    8877    1129    17
18
8020    7914    7475    2113    5939    3607    2998    7451    741     2895    88
32
3961    7527    9083    5734    5266    5224    5809    6885    9715    2454    36
66
2399    7701    8127    3198    4957    6788    2076    2439    8506    6448    67
05
5982    4913    2644    5941    7911    95      6682    806     5279    6996    83
34
4363    2730    9952    5939    8539    3190    2006    993     6856    4406    50
46
4983    7604    6356    8124    6032    8795    6630    8832    5500    8964    37
45
4497    4906    1657    4592    7940    8815    9872    4936    3501    587     76
66
```

```
9806    6526    2557    2996    8533    9902    9852    9291    1301    1187    68
95
Number of job assignments explored: 528
Best job assignment is:
Person 0 assigned job 7
Person 1 assigned job 0
Person 2 assigned job 3
Person 3 assigned job 10
Person 4 assigned job 8
Person 5 assigned job 1
Person 6 assigned job 4
Person 7 assigned job 2
Person 8 assigned job 9
Person 9 assigned job 6
Person 10 assigned job 5
Best job assignment cost: 97772
::::::::::::::
time5
::::::::::::::

real    0m0.177s
user    0m0.174s
sys     0m0.024s
::::::::::::::
output6
::::::::::::::
Enter matrix size
13
667     909     127     345     483     390     981     230     76      392     94
3       206     804
842     131     111     258     866     851     966     322     71      849     46
7       221     573
949     226     451     389     228     470     299     708     816     134     98
797     365
526     541     660     732     698     503     863     809     113     729     12
431     51
83      280     870     305     205     819     883     8       208     463     47
9       859     171
295     994     621     444     711     147     337     723     879     35      57
8       94      845
691     823     209     122     226     293     754     96      950     960     91
5       833     968
476     296     799     335     820     446     681     441     242     392     94
1       580     116
820     615     694     267     812     386     90      22      860     317     31
5       615     765
617     575     681     450     895     509     98      47      844     270     49
3       526     712
736     918     5       316     386     825     931     81      444     96      81
9       535     118
679     204     785     294     969     402     221     2       204     117     51
1       654     164
708     925     9       234     989     97      504     994     413     243     17
1       697     324
Number of job assignments explored: 9854
Best job assignment is:
Person 0 assigned job 6
Person 1 assigned job 9
Person 2 assigned job 0
Person 3 assigned job 3
Person 4 assigned job 11
Person 5 assigned job 1
Person 6 assigned job 12
Person 7 assigned job 10
Person 8 assigned job 8
Person 9 assigned job 4
Person 10 assigned job 5
Person 11 assigned job 2
Person 12 assigned job 7
Best job assignment cost: 11632
::::::::::::::
time6
::::::::::::::

real    0m0.230s
user    0m0.241s
sys     0m0.026s
```