Threefold Problem Sheet – Christmas Presents

This sheet presents three problems based on a common storyline. The three problems share several characteristics and as a result they appear quite similar in nature. But this does not necessarily mean that these problems can be solved using the same strategies and techniques. In fact, the whole idea of these threefold sheets is to highlight how possibly subtle differences and changes in a problem formulation can have a great impact on how the resulting problem can be solved. We believe that studying these sheets is an excellent practice of how to approach the modeling aspects of solving a problem.

Read through the problems and try to work out by yourself possible lines of modeling and then algorithmically solving them. Give reasons of why or why not certain approaches appear plausible. Ideally, substantiate your ideas by sketching corresponding proof ideas.

This sheet was handed out during the tutorial on December 12, 2018, and solved and discussed there as well.

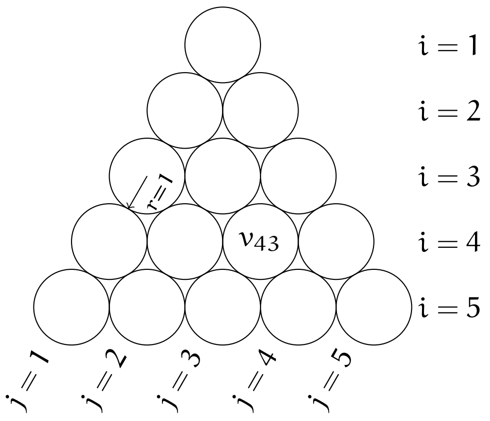
Main Story

Al Gore-Riethmer and his wife Ada prepared a special treat for Christmas this year. For each of their kids they have prepared a pile of presents. Each present is wrapped in a ball-shaped package, and the balls form three Christmas trees, one for each child. Of course, Al and Ada want their children to learn something on the way. Hence they have written an integer value on each of the balls. Your task is to help Alice, Bob and Carol to get the most out of their Christmas tree.

For simplicity, we consider only two-dimensional piles of presents. Each of the three piles has the following form: At the top (first layer), there is one ball. It rests on two balls (second layer), which in turn are put on top of three balls. This goes on all the way to the bottom (layer k), where we have k balls on the ground. Hence, overall there are

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

many balls in the Christmas tree.

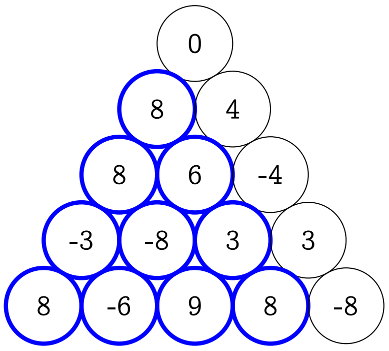


The jth ball in the ith level, denoted B_{ij}, is labeled with an integer value v_{ij}. Every ball in the Christmas tree has unit radius and touches all of its surrounding balls (up to six, depending on the position).

Exercise – *Alice's Accumulation*

Alice is the youngest child of Al and Ada and *really* likes presents. She is known for accumulating a lot of stuff, for example she is crazy about the newest generation of RaspberryPis. To teach Alice that stockpiling as much as you can is not always the best option, her parents have labeled each of the balls with an integer value ν_{ij}, to determine how much Alice will like this particular ball.

Problem On one hand, a package might contain a present (e.g., a RaspberryPi), in which case we have $\nu_{ij} > 0$. On the other hand, a package might also contain a chore (e.g., helping her parents code testsets for an algorithms course that they teach). In such a case, we have $\nu_{ij} \leqslant 0$.



Alice may freely pick any ball B_{xy} she wants. She then receives the *accumulation of all supporting balls*, including B_{xy}. Naturally, Alice wants to maximize the sum of all ν_{ij} in the resulting subpile. Help her to claim the *best-possible accumulation*.

Input The first line contains $1 \leqslant t \leqslant 30$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a line that contains k, the number of layers in Alice's tree ($2 \leqslant k \leqslant 1'000$).
- k lines follow. The ith of these lines contains the values of the i balls in layer i, given as a list of i integers $\nu_{i1}\ \nu_{i2}\ \ldots\ \nu_{ii}$, separated by a space (where $\nu_{ij} \in [-1'000, 1'000]$).

Output For each test case output one line containing the sum of all values in the best-possible accumulation. An accumulation consists of an arbitrary ball B_{xy} together with all supporting balls B_{ij}, where $x < i \leqslant k$ and $y \leqslant j \leqslant y + i - x$.

Points There are three groups of test sets which are worth 100 points in total.

1. For the first group of test sets, worth 20 points, you may assume that $k \leqslant 30$.
2. For the second group of test sets, worth 20 points, you may assume that $k \leqslant 100$.
3. For the third group of test sets, worth 60 points, there are no further assumptions.

Sample Input

```
3
3
0
6 3
-9 3 4
4
-2
6 3
-9 -3 -7
-5 -4 0 9
5
0
8 4
8 6 -4
-3 -8 3 3
8 -6 9 8 -8
```

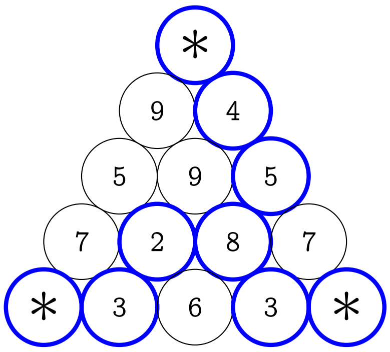
Sample Output

```
10
9
33
```

Exercise – *Bob's Burden*

Bob is the oldest of the three siblings and a freshman Computer Science student at PolyALG. Therefore his parents decided to give him only three presents this year, and they conveniently wrapped those in the corner packages B_{11}, B_{k1}, B_{kk} of Bob's Christmas tree. The other balls are empty, but they are necessary to support the tree construction. Depending on the strength and thickness of the cardboard it is made of, each ball has a particular weight. Luckily for Bob, Al and Ada wrote down on each ball its corresponding weight in grams.

Problem To guarantee structural soundness of Bob's present, he can not simply pick the three packages B_{11}, B_{k1}, B_{kk}. Instead he is required to choose a connected subset of balls in his Christmas tree, i.e. a subset of balls that connects the three apices of the tree.



For each ball B_{ij}, Bob is given its weight v_{ij}. Bob has to switch trains several times on his way home to his flat share. Naturally, he wants to minimize the sum of all weights v_{ij} in his subset of packages. Since Bob certainly picks his presents, we may assume they have weight $v_{11} = v_{k1} = v_{kk} = 0$, while all other balls have weight $v_{ij} > 0$. Help Bob carry the *minimum burden*.

Input The first line contains $1 \leqslant t \leqslant 30$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a line that contains k, the number of layers in Bob's tree ($2 \leqslant k \leqslant 800$).
- k lines follow. The ith of these lines contains the values of the i balls in layer i, given as a list of i integers $v_{i1}\ v_{i2}\ \dots\ v_{ii}$, separated by a space (where $v_{11} = v_{k1} = v_{kk} = 0$ and $v_{ij} \in [1, 1'000]$ otherwise).

Output For each test case output one line containing the smallest possible burden in grams. A burden is the total weight of a subset S of balls that connects the three apex packages. (S connects the three apex packages, if for each pair $B_a, B_b \in \{B_{11}, B_{k1}, B_{kk}\}$ there is a sequence of balls $B_0, B_1, \dots, B_j \in S$, such that $B_0 = B_a$, $B_j = B_b$ and for all i, $0 \leqslant i < j$, B_i touches B_{i+1}.)

Points There are two groups of test sets which are worth 100 points in total.

1. For the first group of test sets, worth 40 points, you may assume that $k \leqslant 40$.
2. For the second group of test sets, worth 60 points, there are no further assumptions.

Sample Input

```
3
3
0
8 6
0 2 0
4
0
9 8
5 3 5
0 6 7 0
5
0
9 4
5 9 5
7 2 8 7
0 3 6 3 0
```

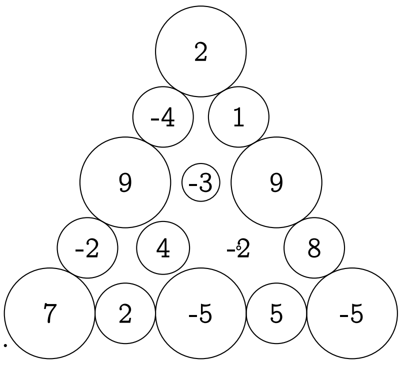
Sample Output

```
8
21
25
```

Exercise – *Carol's Configuration*

Carol is the most creative of the Gore-Riethmer family. She leaves no design untouched and works on it until it bears her hallmark. Al and Ada are already looking forward to what Carol will do to her Christmas tree. As with Alice's presents, her parents have labeled each of the balls with an integer value describing the liking that Carol takes towards this particular present.

Problem Carol decides to take all the presents her parents have put together for her, even the ones with negative value. This doesn't matter so much for Carol, because she has found a way to tune the presents according to her interests:



For each ball B_{ij}, Carol chooses a radius $0 \leqslant r_{ij} \leqslant 2$. Instead of the old value v_{ij}, the package turns into a present of value $r_{ij} \cdot v_{ij}$ — obviously, the larger the value v_{ij} is, the more Carol wants to enlarge the radius r_{ij}.

Nonetheless she takes care of some restrictions in order for the Christmas tree to keep its shape: (i) the center of each ball needs to stay at its original position, (ii) no two enlarged balls are allowed to intersect (balls are only allowed to touch, if at all), and (iii) the boundary needs to stay connected (formally, for all $i, j < k$ the ball B_{i1} is in touch with $B_{(i+1)1}$, B_{ii} is in touch with $B_{(i+1)(i+1)}$ and B_{kj} with $B_{k(j+1)}$). Help Carol to *configure her presents*.

Input The first line contains $1 \leqslant t \leqslant 30$, the number of testcases. Each of the t testcases is described as follows:

- It starts with a line that contains k, the number of layers in Carol's tree ($2 \leqslant k \leqslant 15$).
- k lines follow. The ith of these lines contains the values of the i balls in layer i, given as a list of i integers $v_{i1}\ v_{i2}\ \dots\ v_{ii}$, separated by a space (where $v_{ij} \in [-1'000, 1'000]$).

Output For each test case output one line containing the largest possible sum $\sum r_{ij}v_{ij}$ of a *valid* configuration, rounded down to the nearest integer.
A configuration is valid, if for all $i, j < k$ the ball B_{i1} is in touch with $B_{(i+1)1}$, the ball B_{ii} is in touch with $B_{(i+1)(i+1)}$, and B_{kj} is in touch with $B_{k(j+1)}$.

Points There are two groups of test sets which are worth 100 points in total.

1. For the first group of test sets, worth 30 points, you may assume that all interior balls have a negative weight (i.e. for all $1 < j < i < k$ we have $v_{ij} < 0$).
2. For the second group of test sets, worth 70 points, there are no further assumptions.

Sample Input

```
3
3
-4
3 -1
6 8 2
4
-5
8 3
0 9 4
-1 9 8 4
5
2
-4 1
9 -3 9
-2 4 -2 8
7 2 -5 5 -5
```

Sample Output

```
14
39
34
```

Briefly sketch your approach for each of the three exercises.

Exercise 1 – *Alice's Accumulation*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting
☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree
☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows
☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 2 – *Bob's Burden*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting
☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree
☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows
☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning:

Exercise 3 – *Carol's Configuration*

Method(s): ☐ Brute force ☐ Greedy ☐ Backtracking ☐ Dynamic programming ☐ Sorting
☐ Sliding window ☐ Binary search ☐ Graph components ☐ Minimum spanning tree
☐ BFS/DFS ☐ Shortest paths (Dijkstra) ☐ Maximum matching ☐ Network flows
☐ MinCost flows ☐ Delaunay triangulation ☐ Minimum enclosing circle ☐ LP ☐ QP

Time Complexity:

Reasoning: