2009 the 34th ACM Collegiate Programming Contest Asia Regional

Hsinchu Site

October 31, 2009

- **Problems:** There are 9 problems in this set.
- **Problem Input:** The inputs to the problems are through the input files. The file names are given in the following table. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- **Problem Output:** All output should be directed to standard output (screen output).
- **Time Limit:** The judges will run the submitted program with certain time limit as shown in the following table.

	Problem Name	Input File	Time Limit
Problem A	Interstar Transport	pa.in	1 second
Problem B	Clues	pb.in	3 seconds
Problem C	Inventory	pc.in	10 seconds
Problem D	Vaccination Centers	pd.in	2 seconds
Problem E	Schedule Pairs of Jobs	pe.in	30 seconds
Problem F	A Constrained Queen Game	pf.in	2 seconds
Problem G	Location for a Power Generator	pg.in	10 second
Problem H	Movie Promotion	ph.in	15 seconds
Problem I	Magic Rope	pi.in	5 seconds

Problem A
Interstar Transport

Input File: *pa.in*
Time Limit: 1 second

Problem Description

By 2100, space travel will be a reality for the Milky Way (Solar Galaxy) residents. The Interstar Transport Travel agency operates scheduled direct space transports (flights) between some of the most popular planet destinations in the Milky Way. The cost of these scheduled direct transports are predetermined in Galaro (Galaxy Currency unit) and are published in many different languages. For travel to planets that is not on the schedule, there are slower, yet free, space flights from the closest planet that is on the direct transport list. To help travelers plan their itinerary, the Interstar Transport wants to offer a mobile application that can find the best traveling option, based on the total cost of the direct transports on the itinerary. Given the starting and destination planets on the itinerary, find the sequence of direct transports that has the lowest total traveling cost. Output all the planets in sequence that one must pass through on this best route. If two or more routes exist with the same cost, then the route that goes through the least number of intermediate planets is considered a better route. There will always exist a unique best route for any of the given test cases.

Technical Specification

1. The number of planets on the direct transport list is at most s, $1 \leq s \leq 26$. The planets are labeled using capital letters of the English alphabets, i.e., "A", "B", "C", ..., "Z", in no particular order.

2. The Interstar Transport operates at most p, $1 \leq p \leq 200$, direct transports between planets. There is at most one (could be none) direct transport between any two distinct planets.

3. The cost of any transport is given as a natural number less than or equal to 100 Galaros.

Input File Format

The first line of the input file contains two integers, s and p, separated by a space. The next p lines each contains two letters, e_i and e_j, followed by a natural number, d_{ij}, indicating that there exists a direct transport between planets e_i and e_j with a cost of d_{ij}. The next line contains an integer $n \leq 20$, indicating the number of queries to follow. For each of the next n lines, each line contains two letters e_k and e_m, indicating a user is looking for a best (lowest cost) way to get from planet e_k to planet e_m.

Output Format

For each of the n queries in the input, output on one line the best route to take, in the sequence of starting planet, the intermediate planets in sequence along the route and the destination planet; all separated by one blank space.

Sample Input

```
5 7
A B 1
B C 2
C D 3
D E 2
E A 1
A D 3
A C 4
3
B D
A D
E C
```

Sample Output for the Sample Input

```
B A D
A D
E A B C
```

Problem B
Clues

Input File: *pb.in*
Time Limit: 3 seconds

Problem Description

Dr. Wolf is an employee of Academic Cipher Machinery (ACM). His work is to secure a plenty of electronic documents by using a cipher machine invented by ACM. Given a document, the cipher machine is capable of making it encrypted whenever an arbitrary prime number, called the *key prime*, is provided. In mathematics, a prime number is a positive integer which has exactly two distinct positive divisors: 1 and itself. Of course, an encrypted document can be decrypted if the corresponding key prime is available.

As a consequence, Dr. Wolf picked many key primes for those documents to be secured. This seems to be an easy task. However, Dr. Wolf found that it is very difficult for him to remember so many key primes. Therefore, he decided to write down some information of the key primes in a notebook. To ensure safety, only a *clue* is recorded for each key prime. Let k_0 be a key prime. Dr. Wolf produces a clue C for k_0 according to the following strategy. Initially, C is an empty sequence. In Step 1, select an integer $r \geq 1$ as well as $r-1$ prime numbers $k_1, k_2, \ldots, k_{r-1}$ that are not larger than k_0, and then include r into C. In Step 2, for each k_i, $0 \leq i \leq r-1$, either include k_i into C, or partition k_i into smaller positive integers (adding up to exactly k_i) and then include the smaller integers into C. Finally, in Step 3, rearrange the integers in C non-decreasingly. For example, a clue for $k_0 = 13$ is made as follows. In Step 1, select $r = 4$ and $(k_1, k_2, k_3) = (5, 5, 7)$, and include 4 into C. In Step 2, partition $k_0 = 13$ into (2, 4, 7), partition $k_1 = 5$ into (1, 4), partition $k_2 = 5$ into (2, 3), and partition $k_3 = 7$ into (1, 1, 5). After Step 2, we have $C = (4, 2, 4, 7, 1, 4, 2, 3, 1, 1, 5)$. Finally, in Step 3, we obtain a clue $C = (1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 7)$.

Dr. Wolf would use clues to recover the original key primes. Unfortunately, Dr. Wolf found that there is a drawback in his strategy: the key prime that can be inferred from a given clue may not be unique! For example, consider the clue $C = (1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 7)$. We may conclude $k_0 = 13$ by letting $r = 4$ and $(k_0, k_1, k_2, k_3) = (2+4+7, 1+4, 2+3, 1+1+5) = (13, 5, 5, 7)$. However, we may also conclude $k_0 = 17$ by letting $r = 4$ and $(k_0, k_1, k_2, k_3) = (2+4+7+4, 1+2, 3, 1+1+5) = (17, 3, 3, 7)$, or conclude $k_0 = 29$ by letting $r = 2$ and $(k_0, k_1) = (2+3+4+4+4+5+7, 1+1+1) = (29, 3)$. To overcome this drawback, Dr. Wolf calls a clue of a key prime k_0 *good* if the largest key prime that can be inferred from it is k_0. In the above example, $C = (1, 1, 1, 2, 2, 3, 4, 4, 4, 5, 7)$ is good if $k_0 = 29$. In order to produce good clues, Dr. Wolf needs a program that computes the largest key prime that can be inferred from a given clue. Therefore, Dr. Wolf seeks for your help.

Technical Specification

1. The number of integers in a clue, n: $3 \leq n \leq 14$.
2. Each integer in a clue ranges from 1 to 10000.

Input File Format

There are at most 25 test cases. Each test case describes a clue $C = (c_1, c_2, \ldots, c_n)$ in two lines. The first line contains the integer n, where $3 \leq n \leq 14$. The second line contains the n integers $c_1, c_2, \ldots, c_n$, where $1 \leq c_1 \leq c_2 \leq \ldots \leq c_n \leq 10000$. The last test case is followed by a line containing a number -1.

Output Format

For each test case, print a line containing the test case number (beginning with 1) followed by the largest key prime that can be inferred from the clue C. If no key prime can be inferred, print "not a valid clue". (Dr. Wolf may make mistakes in Step 2, during the production of a clue.) Use the format of the sample output.

Sample Input

```
6
1 1 2 4 5 8
11
1 1 1 2 2 3 4 4 4 5 7
3
1 8 9
4
4 5 5 5
6
1 3 4 5 6 8
-1
```

Sample Output for the Sample Input

```
Case 1: 17
Case 2: 29
Case 3: 17
Case 4: not a valid clue
Case 5: not a valid clue
```

Problem C
Inventory

Input file: *pc.in*
Time limit: 10 seconds

Problem Description

A factory produces products as follows. There are N days in the coming production season. Each morning the factory orders materials. At noon the materials will arrive and shipped into a warehouse. In the afternoon the factory produces products using the material in the warehouse. There is only one kind of material and one kind of product, and one unit of materials can make one unit of product. The factory needs to produce exactly d_i products on the i-th day of the production season, where i is between 1 and N. Also if the factory does not use all the materials in the warehouse, it can use them later, but the material can only be stored in the warehouse for up to 2 days, after that they cannot be used to make products. That is, if a material was shipped to the factory on the i-th day, it can be used only on the i-th and $i + 1$-th day.

We want to order materials for this factory with minimum cost. Let p_i be the price of material on the i-th day and n_i be the maximum number of units of material one can order on the i-th day. Both p_i and n_i are changing daily. Note that we may want to order more materials when they are cheap, but keep in mind that we can order at most n_i units on the i-th day, and we must use the materials within two days after we order them. Fortunately, the warehouse is large enough to store *all* materials for the entire production season, so we do not need to worry about its capacity. Instead we only need to determine the amount of materials to order for each day, so that we can produce d_i products on the i-th day, and the entire material cost is minimized.

Technical Specification

1. The number of cases is a positive integer no more than 100.
2. The number of days N is a positive integer no more than 1000.
3. $1 \leq p_i, d_i \leq 100$, $1 \leq n_i \leq 10000$ for every i.

Input Format

The first line of the input file contains an integer indicating the number of test cases to follow. The first line of the test case has the number of days (N) in the production season. Each of the next N lines has the price of material per unit (p_i), the maximum number of units of material (n_i) the factory can order, and the amount of materials (d_i) ordered for that day.

Output Format

For each test case, output the minimum material cost to produce all products. It is assumed that there exists a solution.

Sample Input

```
2
3
59 5 1
96 9 1
67 6 11
3
26 3 1
76 9 11
56 6 2
```

Sample Output for the Sample Input

```
1000
874
```

Problem D
Vaccination Centers

Input file: *pd.in*
Time limit: 2 seconds

Problem Description

To prepare for the flu pandemic, country E wants to set up some vaccination centers. There are n regions in the country, numbered $0, 1, 2, ..., n-1$. The costs associated with the vaccination plan are the set up cost and the administration cost. The set up cost, namely c_i, is the expense to build a vaccination center at region i, for $i = 0$ to $n-1$. The administration cost mainly consists of the transportation cost for people to travel to nearby regions if no center established in their own region. It is certainly a difficult task to find out the set up costs and administration costs. Fortunately, the transportation bureau and the center for disease control have those numbers. Their investigation determined that the administration cost is a function of the distance from one region to the region where the closest vaccination center resides. More specifically, if there is no vaccination center in region i and region j is the nearest region of i with a vaccination center, the administration cost of region i would be the length of the shortest path from region i to region j. It is clear that the administration cost for a region with a vaccination center will be zero. Since E is famous for its efficiency, their transportation system connects all the regions. However, the structure of the road system is a tree. That is, there is only one route from one region to another. Now, the authority has decided that the number of vaccination centers must be smaller or equal to p. Once a set of regions V has been decided as the vaccination centers, the total set up cost will be the sum of the set up cost for each chosen region; the total administration cost will be the summation of the administration costs of all regions, and the total cost will be the sum of the total set up cost and the total administration cost. You are to write a program to select at most p vaccination centers so that the total cost is minimized.

Technical Specification

1. $3 \leq n \leq 30$.
2. $0 < p \leq 10$.
3. Let $link(i, j)$ be the distance between region i and region j that are directly connected with each other. $0 < link(i, j) \leq 10000$. $0 < c_i \leq 50000$. $c_i, link(i, j) \in \mathbb{N}$.

Input File Format

There are five cases in the input file. The first line of each test case contains two integers n, p separated by a space; the following line contains n numbers corresponding to $c_0, c_1, c_2, \cdots, c_{n-1}$ (the set up cost for the n regions numbered from 0 to $n-1$). Each of the next $n-1$ lines contains three numbers separated by a space, the first two numbers of which indicate the identification numbers of two regions that are directly connected with

each other, and the third number is the distance between these two regions. For example, "0 1 3" denotes $link(0, 1) = 3$.

Output Format

Output the total cost for each test case.

Sample Input 1

```
4 3
47 42 15 33
0 1 9
0 2 10
1 3 1
3 2
19 12 38
0 1 1
0 2 10
5 1
11 45 29 46 27
0 1 5
0 2 6
0 3 3
1 4 3
3 2
13 19 44
0 1 5
1 2 5
4 2
36 41 11 2
0 1 3
0 2 9
1 3 6
```

Sample Output for the Sample Input 1

```
59
24
33
28
28
```

Sample Input 2

```
2 1
```

```
30911 26788
0 1 1714
4 2
38688 34803 3330 22445
0 1 6437
0 2 3051
1 3 9544
3 1
24862 17211 29621
0 1 4550
1 2 9766
2 1
36346 28049
0 1 7991
3 2
45185 3150 10078
0 1 8665
0 2 8943
```

Sample Output for the Sample Input 2

```
28502
34901
31527
36040
21893
```

(This page is left blank intentionally.)

Problem E
Schedule Pairs of Jobs

Input File: *pe.in*
Time Limit: 30 second

Problem Description

In a factory, there are n pairs of jobs, (p_i, q_i), $i = 1, 2, \ldots, n$, to be scheduled. Each job, p_i or q_i, needs 1 unit of time to process. All the jobs p_i, $i = 1, 2, \ldots, n$, must be scheduled before of all the jobs q_i, $i = 1, 2, \ldots, n$. The order among the jobs p_i, $i = 1, 2, \ldots, n$, as well as the order among the jobs q_i, $i = 1, 2, \ldots, n$, is not important. However, it is required that the time between p_i and q_i, measured from the starting time of p_i to the starting time of q_i, should be at most d_i, for $i = 1, 2, \ldots, n$.

Given a sequence of n positive integers $d_1, d_2, \ldots, d_n$, we want to know whether these n pairs of jobs can be scheduled in the time interval $[0, 2n]$ or not. We say that the problem is *solvable* if the n pairs of jobs can be scheduled in a time interval of length $2n$ units, in such a way that the time between p_i and q_i is at most d_i, for $i = 1, 2, \ldots, n$.

For example, for $n = 3$, the sequence $1, 3, 5$ is solvable, since we can schedule these 3 pairs of jobs as follows:

p_3	p_2	p_1	q_1	q_2	q_3

The sequence $3, 3, 4, 6$ is also solvable, since we can schedule the jobs in the following way:

p_3	p_4	p_2	p_1	q_3	q_2	q_1	q_4

In this problem, you are going to design a computer program to schedule pairs of jobs with the above constraints.

Technical Specification

Assume that $n < 16$, and each $d_i < 2^{31}$. For simplicity, assume that $d_1 \leq d_2 \leq \cdots \leq d_n$, $\sum_{i=1}^{k} d_i \geq k^2$ for $1 \leq k < n$, and $\sum_{i=1}^{n} d_i = n^2$. Note that, in this case, if the problem is solvable then the time between each pair of jobs (p_i, q_i) is exactly d_i.

If the solution is not unique, try to schedule the jobs so that the job q_i with smaller index is finished as early as possible. For example, let the input requirements be `3 3 4 6`. Then print out the solution `p4 p1 p2 p3 q1 q2 q4 q3`.

Input File Format

Input file contains a set of test cases. Each test case contains a positive integer n, followed by n integers d_i, $1 \leq i \leq n$. The last test case is followed by a line containing only one integer 0.

Output Format

Print the the job in ascending order of their starting time. Print one line for each test case and for readability print a space before each "p" and "q". If the pairs of jobs cannot be scheduled, then print the message "`no solution`" in that line.

Sample Input

```
3
1 3 5
4
3 3 4 6
6
4 4 4 8 8 8
0
```

Sample Output for the Sample Input

```
 p3 p2 p1 q1 q2 q3
 p4 p1 p2 p3 q1 q2 q4 q3
no solution
```

Problem F
A Constrained Queen Game

Input File: *pf.in*
Time Limit: 2 seconds

Problem Description

Fig. 1 shows a well-known 8-queen problem solution in an 8×8 chess board. The problem is to place 8 queens in a chess board so that none of them is able to capture any other using the standard chess queen's moves.

Now, the chess board size can be $N \times N$ and you need to place N queens in the game. Besides, each square in the chess board has a score. The score is a positive integer (at most 5000) and is provided as part of the test data.

Let the score on square (i, j) be $score(i, j)$. The scores provided for a chess board follow a rule:

$$score(i, j) < score(s, t) \qquad if\ (s > i\ and\ t \geq j)\ or\ (s \geq i\ and\ t > j)$$

According to this rule, the scores on the board always increase toward right, bottom and right-bottom corner. A solution of N queens now has a total score, which is the sum of scores of N queens. Given an $N \times N$ chess board, please find the maximum total score that can be produced by a valid solution.

Technical Specification

1. $8 \leq N \leq 16$

Figure 1: A solution to the 8-queen problem.

Input File Format

The first line of the input file contains an integer indicating the number of test cases (at most 10 cases) to follow. Each test case begins a number N which is the number of queens. Follwing N is $N \times N$ scores for all the squares. These scores are listed in row major. For example, in an 8×8 queen board, the first eight scores are for squares indexed $(1,1),(1,2),(1,3)\ldots(1,8)$.

Output Format

For each test case, please output the maximum total score that can be produced by a valid solution.

Sample Input

```
1
8
1   2   3   4   5    6    7   8
2   4   6   8  10   12   14  16
3   6   9  12  15   18   21  27
4   8  12  16  20   24   28  32
5  10  15  20  25   30   35  40
6  12  18  24  30   36   42  48
7  14  21  28  35   42  500 550
8  16  24  32  40   48  550 999
```

Sample Output for the Sample Input

```
1097
```

Problem G
Location for a Power Generator

Input File: *pg.in*
Time Limit: 10 second

Problem Description

Happyland is a small island in Pacific Ocean. There are n families dwelling in the island. Each family owns a house and the address is specified by the coordinate (x, y), x and y are not necessary to be integers. One day, Happyland is struck by a typhoon and the electrical power system is destroyed.

The mayor of Happyland has a mobile nuclear power generator that can provide electricity to all of the house in the island. The power generator can be placed at any place in the Happyland. Each family has a power cord of length l. By connecting the power cord to the generator, the family can have electricity. Thus, even the power generator can supply electricity to all of the n families but the limitation is the length of the power cord. For example, if the distance between two houses is greater than $2l$, the two houses can not have electricity simultaneously.

The Mayor wants to find a good place for the power generator. He thinks the best spot should be a place where if the power generator is placed, the number of houses received electricity is maximized. Please help the mayor to find out the places for the generator.

There are different regions for the power generator to support the maximum number of houses. A region, R, in the Happyland is a k-region, if a power generator locating in R can serve exactly k houses. The power generator locating in the k-region can serve the maximum number of houses when k is maximized. Let $k_{\max}$ be the largest k. There are cases the number of $k_{\max}$-regions is not unique. Please design the code to compute $k_{\max}$ and the number of $k_{\max}$-regions.

Technical Specification

1. There are at most 1100 houses in Happyland.
2. The coordinates for each house and the length of the cord are real number. In the input file, there are 3 digits below decimal point.
3. Please use double precision to hold real number in your code.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. There is a blank line between two consecutive cases. Each set of data starts with a real number, l, which is the length of the power cord. Following the real number is an integer, n, that stands for the number of houses. Then there are n pairs of real numbers to

specify the addresses of the houses.

Output Format

For each test case, output two integers. The first integer is $k_{\max}$. The second integer is the number of $k_{\max}$-regions.

Sample Input 1

```
1
2.1
3
0.0 0.0
4.0 0.0
2.0 3.5
```

Sample Output for the Sample Input 1

```
2 3
```

Sample Input 2

```
2
2.5
3
0.0 0.0
4.0 0.0
2.0 3.5

2.0
3
0.0 0.0
4.0 0.0
2.0 3.5
```

Sample Output for the Sample Input 2

```
3 1
2 1
```

Problem H
Movie Promotion

Input File: *ph.in*
Time Limit: 15 seconds

Problem Description

All Cool Movie (ACM) is a company that runs a website with many movie DVDs for online rental service. A user who subscribes to the service not only can rent DVDs by a few online clicks, but also can rate the movies that she has seen using ranks of `one-star`, `two-star`, `three-star`, `four-star`, or `five-star`. `One-star` means that the user absolutely hates the movie, and `five-star` means that the user likes it very much. ACM's success is based on a simple but robust model that predicts the users' preferences on unseen movies. In particular, the company uses the ratings that were gathered to predict the users' preferences on unseen movies.

ACM's model is as follows. Assume that there are U users and M movies in the system, and user i rates movie j as rank $r_{i,j}$. ACM tries to decompose the rank into two factors: the user factor u_i and the movie factor m_j. In other words, it seeks to find a real value u_i and a real value m_j such that $\hat{r}_{i,j} = u_i + m_j \approx r_{i,j}$. For calibration, there is assumed to be a pseudo user 0 with $u_0 = 0$ that rates every movie as rank 3 (i.e., $r_{0,j} = 3$ for all $1 \le j \le M$). There is also a pseudo movie 0 with $m_0 = 0$ that gets rank 3 from every user (i.e., $r_{i,0} = 3$ for all $1 \le i \le U$).

After gathering N tuples of $(i, j, r_{i,j})$ pairs from the users, ACM then determines the optimal values of $(u_1, \cdots, u_i, \cdots, u_U)$ and $(m_1, \cdots, m_j, \cdots, m_M)$ by minimizing the sum of squared error $(r_{i,j} - \hat{r}_{i,j})^2$ over all the N tuples, the ranks from pseudo user 0 and the ranks for pseudo movie 0.

Because of the success in business, ACM decides to run a promotion as a thank-you gesture to its users. During the promotion period, each user would be freely awarded with one movie—no more, no less—that she/he has not rated (which is taken to mean that she/he has not seen the movie). Of course, ACM doesn't have many DVDs for each movie, and thus can only give the same movie to at most two different users. Assume that user k gets movie $g(k)$ during the promotion, her satisfaction is assumed to be $\left(\lfloor \hat{r}_{k,g(k)} \rfloor\right)^2$, where $\lfloor \hat{r}_{k,g(k)} \rfloor$ is the largest integer smaller than or equal to $\hat{r}_{k,g(k)}$. ACM's goal during the promotion is then maximizing the sum of the satisfaction from all the users under the constraints above. Given the N tuples that were gathered on ACM's website, please determine whether ACM's promotion plan is feasible. If so, please compute the maximum overall satisfaction $\sum_{k=1}^{U} \left(\lfloor \hat{r}_{k,g(k)} \rfloor\right)^2$ that can be achieved.

Technical Specification

1. The number of users U is a positive integer no more than 256.
2. The number of movies M is a positive integer no more than 256.
3. The actual users are indexed by $i \in \{1, 2, \cdots, U\}$.

4. The actual movies are indexed by $j \in \{1, 2, \cdots, M\}$.
5. The ratings $r_{i,j}$ are integers in $\{1, 2, 3, 4, 5\}$.
6. Each actual user would rate at least one actual movie; each actual movie would be rated by at least one actual user.
7. For any given actual movie, each actual user would rate it no more than once.

Input Format

The first line of the input file contains an integer indicating the number of test cases to follow. The first line of each test case contains three integers N U M separated by spaces. Each of the following N lines would contain three integers i j $r_{i,j}$ separated by spaces.

Output Format

For each test case, if there is a feasible plan, output the maximum overall satisfaction that can be achieved by the promotion in a line. Otherwise, output `no solution`.

Sample Input

```
2
2 2 2
1 2 1
2 1 5
3 2 2
1 2 1
2 1 5
2 2 4
```

Sample Output for the Sample Input

```
32
no solution
```

Problem I
Magic Rope

Input File: *pi.in*
Time Limit: 5 seconds

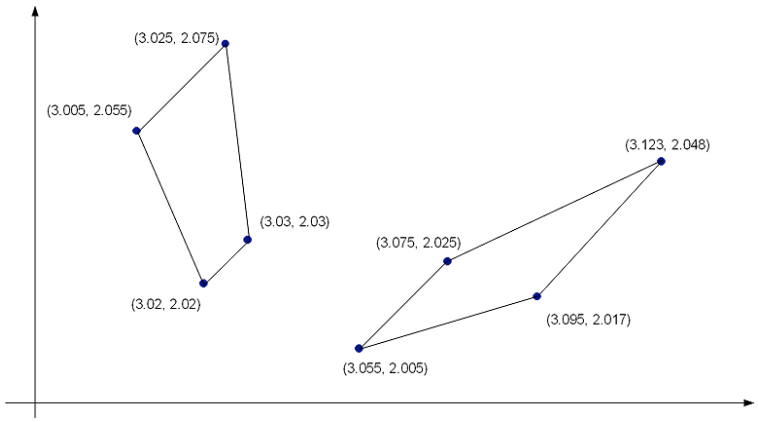
Problem Description

A young magician, David Jr., just learned an *Ancient Cord Magic* (ACM) that he can teleport a number of objects at a time to anywhere as he wishes. However, since David Jr. is not proficient in ACM yet, he still needs to use the tool, called *Magic Rope*, to surround the objects, and all the objects enclosed by the magic rope will be teleported together in an attempt. David Jr. can teleport four objects at most at a time.

David Jr. is liked by everyone because of his special skill of ACM. For example, he is very helpful when his friends are moving in/out the student dorm every year. However, David Jr. sets some rules as follows.

1. His friends need to pay him the moving cost, which is measured by the length of the *Magic Rope* used in teleporting.
2. To save his friends' money, David Jr. will always move four objects at a time. Moreover, he will always use as least length of the magic rope as possible to surround the four objects (i.e., forming a quadrangle, which might be convex or concave, with the minimum circumference) in each teleporting.
3. If there are two groups of objects (four objects in each group) that can be surrounded by the least length of the magic rope, David Jr. will teleport the group that has the smallest X-coordinate value among the *distinct* objects of the two groups, i.e., the objects that belong to exact one (but not both) of the two groups.
4. Every time after David Jr. finishes one round of teleporting, he will make the decision of the next four objects to be teleported, based on the X and Y coordinate values of the rest objects.

For example, there are eight objects on the plane, and their coordinates are (3.02, 2.02), (3.03, 2.03), (3.075, 2.025), (3.055, 2.005), (3.025, 2.075), (3.005, 2.055), (3.095, 2.017) and (3.123, 2.048) respectively. In the first round, David Jr. has two candidate groups of objects to teleport: the first group is (3.02, 2.02), (3.03, 2.03), (3.075, 2.025), and (3.055, 2.005); and the second group is (3.02, 2.02), (3.03, 2.03), (3.025, 2.075), and (3.005, 2.055). Both groups can be surrounding by a magic rope of the cord length 0.1258. However, since there are only four distinct objects in the two groups, i.e., (3.075, 2.025), (3.055, 2.005), (3.025, 2.075), and (3.005, 2.055), and the second group has the object (3.005, 2.055) which has the smallest X-coordinate value among the four distinct objects, David Jr. will teleport the second group in the first round. Then, he will teleport the remaining objects in the second round (cord length is 0.1650). The total length of the magic rope used is $0.1258 + 0.1650 = 0.2908$.



Technical Specification

1. There are $4N$ objects. N is an integer, and $1 \leq N \leq 50$.
2. The X, Y coordinates of the i-th object is (X_i, Y_i). X_i and Y_i are positive real numbers; moreover, $0 < X_i \leq 1,000$ and $0 < Y_i \leq 1,000$.
3. For any two distinct objects A and B, we know that $X_A \neq X_B$ and $Y_A \neq Y_B$.
4. There are no three objects collinear on the plane.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. For each test case, the first line contains a positive integer N, representing the number of rounds required to teleport the input objects. In the following $4N$ lines, the i-th line contains two positive real numbers, X_i and Y_i, which are separated by one single space and represent the X and Y coordinate values of the i-th object respectively (the values of X_i and Y_i are round off to four digits after decimal).

Output Format

Please output one number in one line for each test case. The number represents the length of the magic rope required to teleport the input objects. The value of the output number should be round off to four digits after decimal, with error within $\pm 10^{-4}$.

Sample Input 1

```
1
2
2.0000 2.0000
5.0000 4.0000
6.0000 3.0000
7.0000 7.0000
12.0000 5.0000
14.0000 8.0000
16.0000 6.0000
17.0000 10.0000
```

Sample Output for the Sample Input 1

```
30.9146
```

Sample Input 2

```
1
2
3.0200 2.0200
3.0300 2.0300
3.0750 2.0250
3.0550 2.0050
3.0250 2.0750
3.0050 2.0550
3.0950 2.0170
3.1230 2.0480
```

Sample Output for the Sample Input 2

```
0.2908
```