Chapter 5 Practice for Greedy Algorithms

Yonghui Wu

School of Computer Science, Fudan University

yhwu@fudan.edu.cn

Wechat: 13817360465

Greedy algorithms

- Greedy algorithms are used to solve optimization problems through a sequence of steps.
- At each step greedy algorithms make the locally optimal choice in order to find a globally optimal solution.
- For some problems, greedy algorithms can yield a globally optimal solution; but for some problems, such as the traveling salesman problem (TSP), they can't.

Chapter 5 Practice for Greedy Algorithms

- 5.1 Practices for Greedy Algorithms
- 5.2 Greedy-Choices Based on Sorted Data
- 5.3 Greedy Algorithms Used with Other Methods to Solve P-Problems

5.1 Practices for Greedy Algorithms

- Greedy algorithms are used to solve optimization problems through a sequence of steps, and make the choice that looks best at each step.

- Some famous greedy algorithms
  - Prim’s algorithm and Kruskal’s algorithm used to find a minimum spanning tree for a weighted undirected graph;
  - Dijkstra’s algorithm used to get single-source shortest paths between nodes in a graph;
  - Huffman coding;
  - …….

5.1.1 Pass-Muraille

- Source: ACM Tehran 2002 Preliminary
- IDs for Online Judges: POJ 1230， ZOJ 1375

- In modern day magic shows, passing through walls is very popular in which a magician performer passes through several walls in a predesigned stage show. The wall-passer (Pass-Muraille) has a limited wall-passing energy to pass through at most k walls in each wall-passing show. The walls are placed on a grid-like area. An example is shown in Figure 1, where the land is viewed from above. All the walls have unit widths, but different lengths. You may assume that no grid cell belongs to two or more walls. A spectator chooses a column of the grid. Our wall-passer starts from the upper side of the grid and walks along the entire column, passing through every wall in his way to get to the lower side of the grid. If he faces more than k walls when he tries to walk along a column, he would fail presenting a good show.

- For example, in the wall configuration shown in Figure 1, a wall-passer with $k = 3$ can pass from the upper side to the lower side choosing any column except column 6.

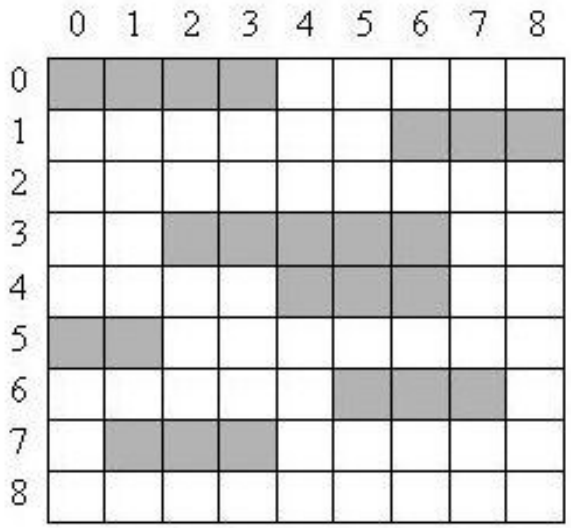


Figure 1. Shaded cells represent the walls.

- Given a wall-passer with a given energy and a show stage, we want to remove the minimum number of walls from the stage so that our performer can pass through all the walls at any column chosen by spectators.

- **Input**
- The first line of the input file contains a single integer t ($1 \leq t \leq 10$), the number of test cases, followed by the input data for each test case. The first line of each test case contains two integers n ($1 \leq n \leq 100$), the number of walls, and k ($0 \leq k \leq 100$), the maximum number of walls that the wall-passer can pass through, respectively. After the first line, there are n lines each containing two (x, y) pairs representing coordinates of the two endpoints of a wall. Coordinates are non-negative integers less than or equal to 100. The upper-left of the grid is assumed to have coordinates (0, 0). The second sample test case below corresponds to the land given in Figure 1.
- **Output**
- There should be one line per test case containing an integer number which is the minimum number of walls to be removed such that the wall-passer can pass through walls starting from any column on the upper side.

Analysis

- All columns are scanned from left to right.
- Removing the minimum number of walls from the stage must guarantee removing the minimum number of walls in scanned columns.

- The optimal solution to the problem consists of its optimal solutions to subproblems.
- The key to the problem is its greedy-choice.

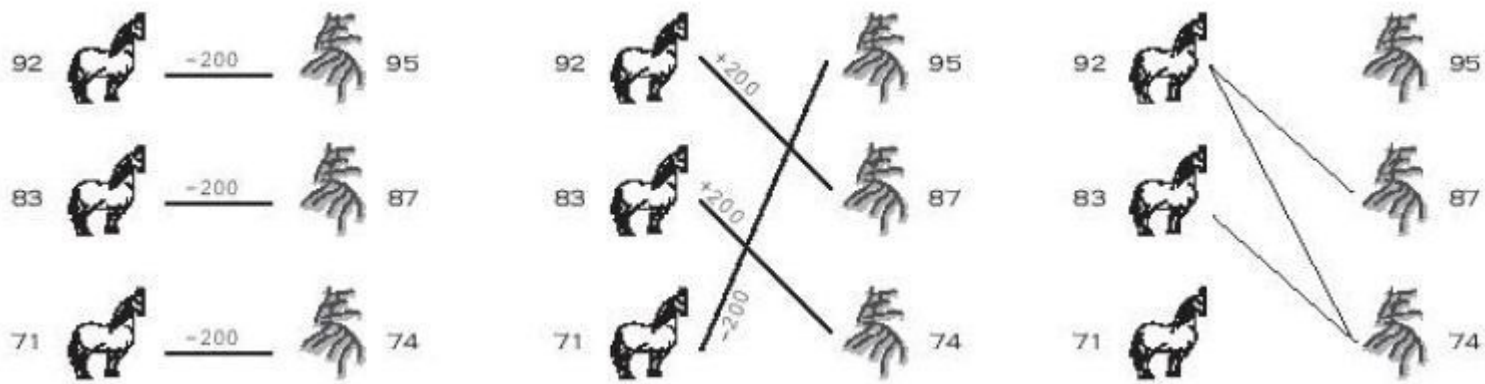
- Suppose there are D walls in the current column.
  - If $D \leq K$, we needn't remove any wall;
  - If $D > K$, $D-K$ walls must be removed.

- The greedy-choice:
  - For walls in the current column, the longest *D-K* walls in unscanned columns are removed.
- The greedy-choice removes minimum number of walls.

5.1.2 Tian Ji -- The Horse Racing

- Source: ACM Shanghai 2004
- IDs for Online Judges: POJ 2287， ZOJ 2397， UVA 3266

- Here is a famous story in Chinese history.
- That was about 2300 years ago. General Tian Ji was a high official in the country Qi. He likes to play horse racing with the king and others.
- Both of Tian and the king have three horses in different classes, namely, regular, plus, and super. The rule is to have three rounds in a match; each of the horses must be used in one round. The winner of a single round takes two hundred silver dollars from the loser.
- Being the most powerful man in the country, the king has so nice horses that in each class his horse is better than Tian's. As a result, each time the king takes six hundred silver dollars from Tian.
- Tian Ji was not happy about that, until he met Sun Bin, one of the most famous generals in Chinese history. Using a little trick due to Sun, Tian Ji brought home two hundred silver dollars and such a grace in the next match.
- It was a rather simple trick. Using his regular class horse race against the super class from the king, they will certainly lose that round. But then his plus beat the king's regular, and his super beat the king's plus. What a simple trick. And how do you think of Tian Ji, the high ranked official in China?


92
-200
95
83
-200
87
71
-200
74
92
+200
95
83
+200
87
71
-200
74
92
95
83
87
71
74

- Were Tian Ji lives in nowadays, he will certainly laugh at himself. Even more, were he sitting in the ACM contest right now, he may discover that the horse racing problem can be simply viewed as finding the maximum matching in a bipartite graph. Draw Tian's horses on one side, and the king's horses on the other. Whenever one of Tian's horses can beat one from the king, we draw an edge between them, meaning we wish to establish this pair. Then, the problem of winning as many rounds as possible is just to find the maximum matching in this graph. If there are ties, the problem becomes more complicated, he needs to assign weights 0, 1, or -1 to all the possible edges, and find a maximum weighted perfect matching...
- However, the horse racing problem is a very special case of bipartite matching. The graph is decided by the speed of the horses -- a vertex of higher speed always beat a vertex of lower speed. In this case, the weighted bipartite matching algorithm is a too advanced tool to deal with the problem.
- In this problem, you are asked to write a program to solve this special case of matching problem.

- **Input**
- The input consists of up to 50 test cases. Each case starts with a positive integer n ($n \leq 1000$) on the first line, which is the number of horses on each side. The next n integers on the second line are the speeds of Tian's horses. Then the next n integers on the third line are the speeds of the king's horses. The input ends with a line that has a single `0' after the last test case.
- **Output**
- For each input case, output a line containing a single number, which is the maximum money Tian Ji will get, in silver dollars.

Analysis

- The problem can be solved by several different methods.
- Maximum matching in a bipartite graph or dynamic programming can be used to solve the problem, but using greedy algorithm to solve the problem is simple and efficient.

- First, the speeds of Tian's horses and the speeds of the king's horses are sorted in ascending order respectively.
- Suppose
  - the sequence for speeds of Tian's current horses in ascending order is $A=a_1 \ldots a_n$;
  - the sequence for the speeds of the king's current horses are sorted in ascending order is $B=b_1 \ldots b_n$.

- Second, greedy-choices are as follow.
- 1. If Tian's current slowest horse is faster than the king's current slowest horse, that is, $a_1 > b_1$;
  - then Tian's current slowest horse races against the king's current slowest horse, that is, a_1 is compared with b_1.
- Because b_1 is less than any elements in A and the king's current slowest horse can be defeated by any Tian’s remainder horse, it is suitable that the king's current slowest horse is defeated by Tian's current slowest horse.

- 2. If Tian's current slowest horse is slower than the king's current slowest horse, that is, $a_1<b_1$;
  - then Tian's current slowest horse races against the king's current fastest horse, that is, a_1 is compared with b_n.
- Because a_1 is less than any elements in B and Tian's current slowest horse can be defeated by any king's remainder horse, it is suitable that Tian's current slowest horse is defeated by the king's current fastest horse.

- 3. If Tian's current fastest horse is faster than the king's current fastest horse, that is, $a_n > b_n$;
  - then Tian's current fastest horse races against the king's current fastest horse, that is, a_n is compared with b_n.
- Because a_n is larger than any elements in B and Tian's current fastest horse can defeat any king’s remainder horse, it is suitable that Tian's current fastest horse defeats the king's current fastest horse.

- 4. If Tian's current fastest horse is slower than the king's current fastest horse, that is, $a_n<b_n$;
  - then Tian's current slowest horse races against the king's current fastest horse, that is, a_1 is compared with b_n.
- Because b_n is larger than any elements in A and the king's current fastest horse can defeat any Tian’s remainder horse, it is suitable that the king's current fastest horse defeats Tian's current slowest horse.

- 5. If ($a_1==b_1$) and ($a_n>b_n$),
  - then it is suitable that Tian's current fastest horse races against the king's current fastest horse, that is, a_n is compared with b_n.

- 6. If (a_n==b_n),
  - then there exist an optimal solution that a_1 is compared with b_n.

- The above process repeats until the horse racing ends.
- Tian's current fastest or slowest horse races against the king's current fastest or slowest horse each time based on above greedy-choices.

- Optimal solutions to subproblems constitute the global optimal solution to the problem.

5.2 Greedy-Choices based on Sorted Data

- The key to a greedy algorithm is its greedy-choices.
- Sometimes the greedy-choices must be based on sorted data.
  - Data are sorted.
  - Greedy-choices are made based on sorted data.

5.2.1 Shoemaker's Problem

- **Source: Second Programming Contest of Alex Gevak， 2000**
- **ID for Online Judge: UVA 10026**

- Shoemaker has N jobs (orders from customers) which he must make. Shoemaker can work on only one job in each day. For each i_{th} job, it is known the integer T_i ($1 \le T_i \le 1000$), the time in days it takes the shoemaker to finish the job. For each day of delay before starting to work for the i_{th} job, shoemaker must pay a fine of S_i ($1 \le S_i \le 10000$) cents. Your task is to help the shoemaker, writing a programm to find the sequence of jobs with minimal total fine.

- **Input**
- The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.
- First line of input contains an integer N ($1 \leq N \leq 1000$). The next N lines each contain two numbers: the time and fine of each task in order.
- **Output**
- For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.
- You programm should print the sequence of jobs with minimal fine. Each job should be represented by its number in input. All integers should be placed on only one output line and separated by one space. If multiple solutions are possible, print the first lexicographically.

Analysis

- "For each day of delay before starting to work for the i_{th} job, shoemaker must pay a fine of S_i cents" means "For each day of delay after starting to work for the i_{th} job, shoemaker must pay a fine of S_i/T_i cents".
- S_i/T_i is the measurement of influence for the i_{th} job, $1 \le i \le n$.
- Therefore, in order to pay minimal fine, the job whose measurement of influence is higher must be finished earlier.

- The greedy algorithm is as follow.
- The metric for jobs is their measurement of influence. The *n* jobs are sorted using their measurement of influence as the first key (in ascending order), and numbers of jobs as the second key (in descending order).
- The sorted sequence is the sequence of jobs with minimal fine.

5.2.4 Radar Installation

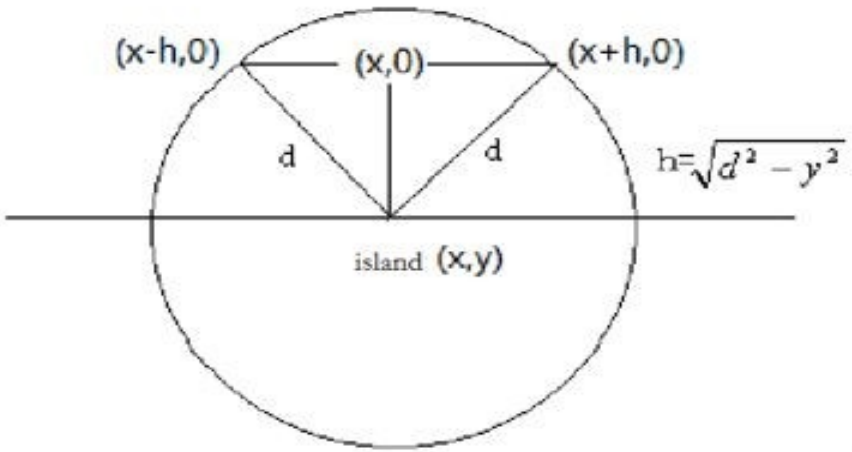
- Source: ACM Beijing 2002
- IDs for Online Judge: POJ 1328， ZOJ 1360， UVA 2519

- Assume the coasting is an infinite straight line. Land is in one side of coasting, sea in the other. Each small island is a point locating in the sea side. And any radar installation, locating on the coasting, can only cover *d* distance, so an island in the sea can be covered by a radius installation, if the distance between them is at most *d*.
- We use Cartesian coordinate system, defining the coasting is the x-axis. The sea side is above x-axis, and the land side below. Given the position of each island in the sea, and given the distance of the coverage of the radar installation, your task is to write a program to find the minimal number of radar installations to cover all the islands. Note that the position of an island is represented by its x-y coordinates.

- **Input**
- The input consists of several test cases. The first line of each case contains two integers n ($1 \le n \le 1000$) and d, where n is the number of islands in the sea and d is the distance of coverage of the radar installation. This is followed by n lines each containing two integers representing the coordinate of the position of each island. Then a blank line follows to separate the cases. The input is terminated by a line containing pair of zeros.
- **Output**
- For each test case output one line consisting of the test case number followed by the minimal number of radar installations needed. "-1" installation means no solution for that case.

Analysis

Each small island is represented as a segment on the coasting. If a rader locates on the segment, the island can be covered by the radar. Suppose the Cartesian coordinate for the island is (x, y). If a rader locates on the coasting from $(x-h, 0)$ to $(x+h, 0)$, where $h = \sqrt{d^2 - y^2}$, the island can be covered. Therefor the island is represented as a segment from $(x-h, 0)$ to $(x+h, 0)$. It can be showed as Figure.



- Suppose there are *n* islands.
- *n* islands are represented as *n* segments.
- Right endpoints are as the first key(in ascending order), left endpoints are as second key (in ascending order), and the *n* segments are sorted.
- All sorted segments are scanned one by one.
  - If the current segment isn't covered by a radar, a radar locates at the right endpoint for the segment.