Programming Assignment 3: Greedy Algorithms

Revision: November 11, 2019

Introduction

In this programming assignment, you will be practicing implementing greedy solutions. As usual, in some problems you just need to implement an algorithm covered in the lectures, while for some others your goal will be to first design an algorithm and then to implement it. Thus, you will be practicing designing an algorithm, proving that it is correct, and implementing it.

Recall that starting from this programming assignment, the grader will show you only the first few tests.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. Apply greedy strategy to solve various computational problems. This will usually require you to design an algorithm that repeatedly makes the most profitable move to construct a solution. You will then need to show that the moves of your algorithm are safe, meaning that they are consistent with at least one optimal solution.
2. Design and implement an efficient greedy algorithm for the following problems:
   (a) changing money with a minimum number of coins;
   (b) maximizing the total value of a loot;
   (c) minimizing the number of tank refills;
   (d) maximizing revenue in online ad placement;
   (e) minimizing work while collecting signatures;
   (f) maximizing the number of prize places in a competition;
   (g) finally, maximizing your salary!

Passing Criteria: 3 out of 7

Passing this programming assignment requires passing at least 3 out of 7 programming challenges from this assignment. In turn, passing a programming challenge requires implementing a solution that passes all the tests for this problem in the grader and does so under the time and memory limits specified in the problem statement.

Contents

1 Money Change

Problem Introduction

In this problem, you will design and implement an elementary greedy algorithm used by cashiers all over the world millions of times per day.

Problem Description

Task. The goal in this problem is to find the minimum number of coins needed to change the input value (an integer) into coins with denominations 1, 5, and 10.

Input Format. The input consists of a single integer m.

Constraints. $1 \leq m \leq 10^3$.

Output Format. Output the minimum number of coins with denominations 1, 5, 10 that changes m.

Sample 1.

Input:

```
2
```

Output:

```
2
```

$2 = 1 + 1$.

Sample 2.

Input:

```
28
```

Output:

```
6
```

$28 = 10 + 10 + 5 + 1 + 1 + 1$.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

2 Maximum Value of the Loot

Problem Introduction

A thief finds much more loot than his bag can fit. Help him to find the most valuable combination of items assuming that any fraction of a loot item can be put into his bag.

Problem Description

Task. The goal of this code problem is to implement an algorithm for the fractional knapsack problem.

Input Format. The first line of the input contains the number n of items and the capacity W of a knapsack. The next n lines define the values and weights of the items. The i-th line contains integers v_i and w_i—the value and the weight of i-th item, respectively.

Constraints. $1 \le n \le 10^3$, $0 \le W \le 2 \cdot 10^6$; $0 \le v_i \le 2 \cdot 10^6$, $0 < w_i \le 2 \cdot 10^6$ for all $1 \le i \le n$. All the numbers are integers.

Output Format. Output the maximal value of fractions of items that fit into the knapsack. The absolute value of the difference between the answer of your program and the optimal value should be at most 10^{-3}. To ensure this, output your answer with at least four digits after the decimal point (otherwise your answer, while being computed correctly, can turn out to be wrong because of rounding issues).

Sample 1.

Input:

```
3 50
60 20
100 50
120 30
```

Output:

```
180.0000
```

To achieve the value 180, we take the first item and the third item into the bag.

Sample 2.

Input:

```
1 10
500 30
```

Output:

```
166.6667
```

Here, we just take one third of the only available item.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

3 Car Fueling

Problem Introduction

You are going to travel to another city that is located d miles away from your home city. Your car can travel at most m miles on a full tank and you start with a full tank. Along your way, there are gas stations at distances $stop_1, stop_2, \ldots, stop_n$ from your home city. What is the minimum number of refills needed?

Problem Description

Input Format. The first line contains an integer d. The second line contains an integer m. The third line specifies an integer n. Finally, the last line contains integers $stop_1, stop_2, \ldots, stop_n$.

Output Format. Assuming that the distance between the cities is d miles, a car can travel at most m miles on a full tank, and there are gas stations at distances $stop_1, stop_2, \ldots, stop_n$ along the way, output the minimum number of refills needed. Assume that the car starts with a full tank. If it is not possible to reach the destination, output -1.

Constraints. $1 \leq d \leq 10^5$. $1 \leq m \leq 400$. $1 \leq n \leq 300$. $0 < stop_1 < stop_2 < \cdots < stop_n < d$.

Sample 1.

Input:

```
950
400
4
200 375 550 750
```

Output:

```
2
```

The distance between the cities is 950, the car can travel at most 400 miles on a full tank. It suffices to make two refills: at points 375 and 750. This is the minimum number of refills as with a single refill one would only be able to travel at most 800 miles.

Sample 2.

Input:

```
10
3
4
1 2 5 9
```

Output:

```
-1
```

One cannot reach the gas station at point 9 as the previous gas station is too far away.

Sample 3.

Input:

```
200
250
2
100
150
```

Output:

```
0
```

There is no need to refill the tank as the car starts with a full tank and can travel for 250 miles whereas the distance to the destination point is 200 miles.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

4 Maximum Advertisement Revenue

Problem Introduction

You have n ads to place on a popular Internet page. For each ad, you know how much is the advertiser willing to pay for one click on this ad. You have set up n slots on your page and estimated the expected number of clicks per day for each slot. Now, your goal is to distribute the ads among the slots to maximize the total revenue.

Problem Description

Task. Given two sequences $a_1, a_2, \ldots, a_n$ (a_i is the profit per click of the i-th ad) and $b_1, b_2, \ldots, b_n$ (b_i is the average number of clicks per day of the i-th slot), we need to partition them into n pairs (a_i, b_j) such that the sum of their products is maximized.

Input Format. The first line contains an integer n, the second one contains a sequence of integers $a_1, a_2, \ldots, a_n$, the third one contains a sequence of integers $b_1, b_2, \ldots, b_n$.

Constraints. $1 \le n \le 10^3$; $-10^5 \le a_i, b_i \le 10^5$ for all $1 \le i \le n$.

Output Format. Output the maximum value of $\sum_{i=1}^{n} a_i c_i$, where $c_1, c_2, \ldots, c_n$ is a permutation of $b_1, b_2, \ldots, b_n$.

Sample 1.

Input:

```
1
23
39
```

Output:

```
897
```

$897 = 23 \cdot 39$.

Sample 2.

Input:

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

Output:

```
23
```

$23 = 3 \cdot 4 + 1 \cdot 1 + (-5) \cdot (-2)$.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

5 Collecting Signatures

Problem Introduction

You are responsible for collecting signatures from all tenants of a certain building. For each tenant, you know a period of time when he or she is at home. You would like to collect all signatures by visiting the building as few times as possible.
The mathematical model for this problem is the following. You are given a set of segments on a line and your goal is to mark as few points on a line as possible so that each segment contains at least one marked point.

Problem Description

Task. Given a set of n segments $\{[a_0, b_0], [a_1, b_1], \ldots, [a_{n-1}, b_{n-1}]\}$ with integer coordinates on a line, find the minimum number m of points such that each segment contains at least one point. That is, find a set of integers X of the minimum size such that for any segment $[a_i, b_i]$ there is a point $x \in X$ such that $a_i \leq x \leq b_i$.

Input Format. The first line of the input contains the number n of segments. Each of the following n lines contains two integers a_i and b_i (separated by a space) defining the coordinates of endpoints of the i-th segment.

Constraints. $1 \leq n \leq 100$; $0 \leq a_i \leq b_i \leq 10^9$ for all $0 \leq i < n$.

Output Format. Output the minimum number m of points on the first line and the integer coordinates of m points (separated by spaces) on the second line. You can output the points in any order. If there are many such sets of points, you can output any set. (It is not difficult to see that there always exist a set of points of the minimum size such that all the coordinates of the points are integers.)

Sample 1.

Input:

```
3
1 3
2 5
3 6
```

Output:

```
1
3
```

In this sample, we have three segments: $[1, 3], [2, 5], [3, 6]$ (of length $2, 3, 3$ respectively). All of them contain the point with coordinate 3: $1 \leq 3 \leq 3$, $2 \leq 3 \leq 5$, $3 \leq 3 \leq 6$.

Sample 2.

Input:

```
4
4 7
1 3
2 5
5 6
```

Output:

```
2
3 6
```

The second and the third segments contain the point with coordinate 3 while the first and the fourth segments contain the point with coordinate 6. All the four segments cannot be covered by a single point, since the segments $[1, 3]$ and $[5, 6]$ are disjoint.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

Solution

A detailed solution (with Python code) for this challenge is covered in the companion MOOCBook. We strongly encourage you to do your best to solve the challenge yourself before looking into the book! There are at least three good reasons for this.

- By solving this challenge, you practice solving algorithmic problems similar to those given at technical interviews.
- The satisfaction and self confidence that you get when passing the grader is priceless =)
- Even if you fail to pass the grader yourself, the time will not be lost as you will better understand the solution from the book and better appreciate the beauty of the underlying ideas.

6 Maximum Number of Prizes

Problem Introduction

You are organizing a funny competition for children. As a prize fund you have n candies. You would like to use these candies for top k places in a competition with a natural restriction that a higher place gets a larger number of candies. To make as many children happy as possible, you are going to find the largest value of k for which it is possible.

Problem Description

Task. The goal of this problem is to represent a given positive integer n as a sum of as many pairwise distinct positive integers as possible. That is, to find the maximum k such that n can be written as $a_1 + a_2 + \cdots + a_k$ where $a_1, \ldots, a_k$ are positive integers and $a_i \neq a_j$ for all $1 \leq i < j \leq k$.

Input Format. The input consists of a single integer n.

Constraints. $1 \leq n \leq 10^9$.

Output Format. In the first line, output the maximum number k such that n can be represented as a sum of k pairwise distinct positive integers. In the second line, output k pairwise distinct positive integers that sum up to n (if there are many such representations, output any of them).

Sample 1.

Input:

```
6
```

Output:

```
3
1 2 3
```

Sample 2.

Input:

```
8
```

Output:

```
3
1 2 5
```

Sample 3.

Input:

```
2
```

Output:

```
1
2
```

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

7 Maximum Salary

Problem Introduction

As the last question of a successful interview, your boss gives you a few pieces of paper with numbers on it and asks you to compose a largest number from these numbers. The resulting number is going to be your salary, so you are very much interested in maximizing this number. How can you do this?

In the lectures, we considered the following algorithm for composing the largest number out of the given *single-digit numbers*.

```
LargestNumber(Digits):
answer ← empty string
while Digits is not empty:
  maxDigit ← −∞
  for digit in Digits:
    if digit ≥ maxDigit:
      maxDigit ← digit
  append maxDigit to answer
  remove maxDigit from Digits
return answer
```

Unfortunately, this algorithm works only in case the input consists of single-digit numbers. For example, for an input consisting of two integers 23 and 3 (23 is not a single-digit number!) it returns 233, while the largest number is in fact 323. In other words, using the largest number from the input as the first number *is not a safe move*.

Your goal in this problem is to tweak the above algorithm so that it works not only for single-digit numbers, but for arbitrary positive integers.

Problem Description

Task. Compose the largest number out of a set of integers.

Input Format. The first line of the input contains an integer n. The second line contains integers $a_1, a_2, \ldots, a_n$.

Constraints. $1 \le n \le 100$; $1 \le a_i \le 10^3$ for all $1 \le i \le n$.

Output Format. Output the largest number that can be composed out of $a_1, a_2, \ldots, a_n$.

Sample 1.

Input:

```
2
21 2
```

Output:

```
221
```

Note that in this case the above algorithm also returns an incorrect answer 212.

Sample 2.

Input:

```
5
9 4 6 1 9
```

Output:

```
99641
```

In this case, the input consists of single-digit numbers only, so the algorithm above computes the right answer.

Sample 3.

Input:

```
3
23 39 92
```

Output:

```
923923
```

As a coincidence, for this input the above algorithm produces the right result, though the input does not have any single-digit numbers.

What To Do

Interestingly, for solving this problem, all you need to do is to replace the check $digit \geq maxDigit$ with a call IsGreaterOrEqual($digit$, $maxDigit$) for an appropriately implemented function IsGreaterOrEqual. For example, IsGreaterOrEqual(2, 21) should return `True`.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.

8 Appendix

8.1 Compiler Flags

C (`gcc 5.2.1`). File extensions: `.c`. Flags:

```
gcc -pipe -O2 -std=c11 <filename> -lm
```

C++ (`g++ 5.2.1`). File extensions: `.cc`, `.cpp`. Flags:

```
g++ -pipe -O2 -std=c++14 <filename> -lm
```

If your `C/C++` compiler does not recognize `-std=c++14` flag, try replacing it with `-std=c++0x` flag or compiling without this flag at all (all starter solutions can be compiled without it). On Linux and MacOS, you most probably have the required compiler. On Windows, you may use your favorite compiler or install, e.g., `cygwin`.

C# (`mono 3.2.8`). File extensions: `.cs`. Flags:

```
mcs
```

Go (`golang 1.12`). File extensions: `.go`. Flags

```
go
```

Haskell (`ghc 7.8.4`). File extensions: `.hs`. Flags:

```
ghc -O2
```

Java (`Open JDK 8`). File extensions: `.java`. Flags:

```
javac -encoding UTF-8
java -Xmx1024m
```

JavaScript (`Node v10.15.3`). File extensions: `.js`. No flags:

```
nodejs
```

Kotlin (`Kotlin 1.2.21`). File extensions: `.kt`. Flags:

```
kotlinc
java -Xmx1024m
```

Python 2 (`CPython 2.7`). File extensions: `.py2` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python2"). No flags:

```
python2
```

Python 3 (`CPython 3.4`). File extensions: `.py3` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python3"). No flags:

```
python3
```

Ruby (`Ruby 2.1.5`). File extensions: `.rb`.

```
ruby
```

Rust (`Rust 1.28.0`). File extensions: `.rs`.

```
rustc
```

Scala (`Scala 2.11.6`). File extensions: `.scala`.

```
scalac
```

8.2 Frequently Asked Questions

Why My Submission Is Not Graded?

You need to create a submission and upload the *source file* (rather than the executable file) of your solution. Make sure that after uploading the file with your solution you press the blue "Submit" button at the bottom. After that, the grading starts, and the submission being graded is enclosed in an orange rectangle. After the testing is finished, the rectangle disappears, and the results of the testing of all problems are shown.

What Are the Possible Grading Outcomes?

There are only two outcomes: "pass" or "no pass." To pass, your program must return a correct answer on all the test cases we prepared for you, and do so under the time and memory constraints specified in the problem statement. If your solution passes, you get the corresponding feedback "Good job!" and get a point for the problem. Your solution fails if it either crashes, returns an incorrect answer, works for too long, or uses too much memory for some test case. The feedback will contain the index of the first test case on which your solution failed and the total number of test cases in the system. The tests for the problem are numbered from 1 to the total number of test cases for the problem, and the program is always tested on all the tests in the order from the first test to the test with the largest number.

Here are the possible outcomes:

- `Good job!` Hurrah! Your solution passed, and you get a point!
- `Wrong answer.` Your solution outputs incorrect answer for some test case. Check that you consider all the cases correctly, avoid integer overflow, output the required white spaces, output the floating point numbers with the required precision, don't output anything in addition to what you are asked to output in the output specification of the problem statement.
- `Time limit exceeded.` Your solution worked longer than the allowed time limit for some test case. Check again the running time of your implementation. Test your program locally on the test of maximum size specified in the problem statement and check how long it works. Check that your program doesn't wait for some input from the user which makes it to wait forever.
- `Memory limit exceeded.` Your solution used more than the allowed memory limit for some test case. Estimate the amount of memory that your program is going to use in the worst case and check that it does not exceed the memory limit. Check that your data structures fit into the memory limit. Check that you don't create large arrays or lists or vectors consisting of empty arrays or empty strings, since those in some cases still eat up memory. Test your program locally on the tests of maximum size specified in the problem statement and look at its memory consumption in the system.
- `Cannot check answer. Perhaps the output format is wrong.` This happens when you output something different than expected. For example, when you are required to output either "Yes" or "No", but instead output 1 or 0. Or your program has empty output. Or your program outputs not only the correct answer, but also some additional information (please follow the exact output format specified in the problem statement). Maybe your program doesn't output anything, because it crashes.
- `Unknown signal 6 (or 7, or 8, or 11, or some other).` This happens when your program crashes. It can be because of a division by zero, accessing memory outside of the array bounds, using uninitialized variables, overly deep recursion that triggers a stack overflow, sorting with a contradictory comparator, removing elements from an empty data structure, trying to allocate too much memory, and many other reasons. Look at your code and think about all those possibilities. Make sure that you use the same compiler and the same compiler flags as we do.
- `Internal error: exception...` Most probably, you submitted a compiled program instead of a source code.
- `Grading failed.` Something wrong happened with the system. Report this through Coursera or edX Help Center.

May I Post My Solution at the Forum?

Please do not post any solutions at the forum or anywhere on the web, even if a solution does not pass the tests (as in this case you are still revealing parts of a correct solution). Our students follow the Honor Code: "I will not make solutions to homework, quizzes, exams, projects, and other assignments available to anyone else (except to the extent an assignment explicitly permits sharing solutions)."

Do I Learn by Trying to Fix My Solution?

My implementation always fails in the grader, though I already tested and stress tested it a lot. Would not it be better if you gave me a solution to this problem or at least the test cases that you use? I will then be able to fix my code and will learn how to avoid making mistakes. Otherwise, I do not feel that I learn anything from solving this problem. I am just stuck.

First of all, learning from your mistakes is one of the best ways to learn.

The process of trying to invent new test cases that might fail your program is difficult but is often enlightening. Thinking about properties of your program makes you understand what happens inside your program and in the general algorithm you're studying much more.

Also, it is important to be able to find a bug in your implementation without knowing a test case and without having a reference solution, just like in real life. Assume that you designed an application and an annoyed user reports that it crashed. Most probably, the user will not tell you the exact sequence of operations that led to a crash. Moreover, there will be no reference application. Hence, it is important to learn how to find a bug in your implementation yourself, without a magic oracle giving you either a test case that your program fails or a reference solution. We encourage you to use programming assignments in this class as a way of practicing this important skill.

If you have already tested your program on all corner cases you can imagine, constructed a set of manual test cases, applied stress testing, etc, but your program still fails, try to ask for help on the forum. We encourage you to do this by first explaining what kind of corner cases you have already considered (it may happen that by writing such a post you will realize that you missed some corner cases!), and only afterwards asking other learners to give you more ideas for tests cases.