Programming Assignment 1

Assigned: Sept. 8
Due: Sept. 29

Part A: Systematic search

Write a program that solves the scheduling problem described in Problem set 2 using iterative deepening.

Input

The input to your code should be a plain text file of the following format:

> **First Line:** The lengths of the tasks as floating points separated by white space.
> **Second Line:** The speeds of the processors as floating points separated by white space.
> **Third Line:** The deadline and the target values, as floating point.

For example, the example in Problem set 1 problem 2 would be

```
12.0 42.0 48.0 54.0
2.0 3.0
25.0 110.0
```

You may assume that all numbers are positive. You may assume that the input is correctly formatted. You do not have to do error handling for incorrect inputs.

Output

The output should be the sequence of processor assignments in order of tasks, with 0 if task is not executed. So the output for the above problem is

```
2 0 1 2
```

If there is no solution then the program should output `No solution.`

Iterative deepening

There is no point in looking for solutions that involve so few tasks that they can't possibly hit the target value.

Specifically, if you sort the tasks by length from highest to lowest, then the minimum number of tasks in the solution, Q, must satisfy $\sum_{i=1}^{Q} T_i.length \geq Target$. So compute that value of Q, and start the iterative deepening at that depth.

Part B: Hill climbing

Solve the same problem using hill climbing with random restart. Here:

- A *state* is a complete assignment to each task of either a processor number or 0, meaning not on the schedule. For example ⟨ `T1` → `P2`, `T2` → 0, `T3` →`P2`, `T4` → 0 ⟩.
- The operators on a state are either to change the assignment of one tasks, or to swap the assignment of two tasks.
  For instance, one neighbor of the above state would be
  ⟨ `T1` → `P2`, `T2` → `P3`, `T3` →`P2`, `T4` → 0 ⟩, changing the assignment of `T2`.
  Another would be ⟨ `T1` → `0`, `T2` → 0, `T3` →`P2`, `T4` → `P2` ⟩, swapping `T1` and `T4`.

  Thus, for any state there are fewer than $N(K-1) + N(N-1)/2$ operators. (Fewer, because swapping two tasks with the same assignment doesn't accomplish anything; and the number of pairs of tasks with the same assignment depends on the state.)
- The cost function is (the shortfall on the value) + (the overflow on the time).

Use a simple hill-climbing strategy with 10 random restarts. (You do not have to worry about sideways motion, because ties will be very rare.)

Coding

Programs will be accepted in C, C++, Java, Python, or Matlab. If you wish to write programs in other languages, you should discuss that with the grader; I leave that decision entirely up to his/her discretion. Programs must run on the department Linux machines; you should check this before you submit your code.

Submit by uploading to the NYU Classes site, the source code for the two parts of the assignment a README file explaining how to compile and run your code.

Grading

8 points for correctly running code. 2 points for well-written code. A programming assignment count equally with a problem set in computing the overall grade.

A program that hard codes any of the values that are supposed to be supplied from input will get 0 points.

Late policy

Programming assignments are due at the beginning of class on the due date. I will accept assignments late with a penalty of 1 point out of 10 for each week late (fractions of a week are rounded up.)