Young and Successful

Dr. Maria Keet
University of Cape Town, South Africa

ICPC Training Session, Aug 16, 2014

The Institute for Compassionate Programming Celebrities is always on the lookout for new members and is trying to come up with schemes to identify candidates, especially from underrepresented groups. August being Women's month, they want to extract from the pool of candidates the successful young women that have a bright future ahead. Calculating what constitutes 'success' is no trivial matter, however. After much deliberation, they have come up with the following celebrity index (C_i) formula for being young and successful (Y_s), which takes into account degree obtained (D, coded as a number), weekly time (T) spent on selfless contribution to society (like teaching maths to kids), age (A), number of leadership activities (L, like coaching the SA delegation to the IOI), and yearly salary (S), with γ an additional goodness factor and ϕ the percentage of women on the candidate's team at work or in the student cohort for her year of starting her study:

$$C_i = \sqrt{\frac{(\frac{T}{10} + \frac{L}{2})\gamma + D^2}{\frac{S}{100000} + \frac{10}{A}}} + \frac{50}{\phi} \tag{1}$$

A candidate is successful if $C_i > 1.5$ whilst $A \leq 30$, resulting in a set of candidates Y_s of the overall entries M in their database (so, $Y_s \subseteq M$).

The Institute for Compassionate Programming Celebrities has tasked a grumpy grad student to implement it and run it over the entries in their database. The grumpy student, however, is convinced that just mapping age against salary will give the same set Y_s of candidates, using $20 \leq A < 30$ and $400000 < S \leq 800000$. The grumpy grad now has delegated this shortcut to you to implement it.

You have extracted from the database the age and salary of the entries, constituting the input file. You see that A ranges from 12 to 65 and S from 50000 to 1000000. Each potential candidate is represented on one line, with a 5-digit identifier, then age, then salary, separated by a space. The input ends with a `-1`. The output lists the identifiers of the members of Y_s, one line for each candidate.

Sample input

```
10101 15 50000
10102 25 350000
21101 24 410000
24500 28 690000
24345 31 555000
-1
```

Sample output

```
21101
24500
```

WARNING: the outline of the solution is on the next page, so do NOT scroll down further if you're still working on the solution!

Solution

I made up the formula (and didn't even bother to check what values would come out of it), which was in the problem description only to test whether you have read it and saw that it has an algorithmic solution rather than something maths-y, which it might have suggested to be at first sight.

Recognise that we have an XY scatter plot, with age A on the X-axis and salary S on the Y-axis, and the individual 'candidates' are points within the area demarcated by a lowest value on the X-axis 12 and highest 65, and for the Y-axis, the values range between 50000 and 1000000. To find Y_s, we need to slice the area so that only those points within the rectangle of $20 \leq A < 30$ and $400000 < S \leq 800000$ will be in the answer (output).

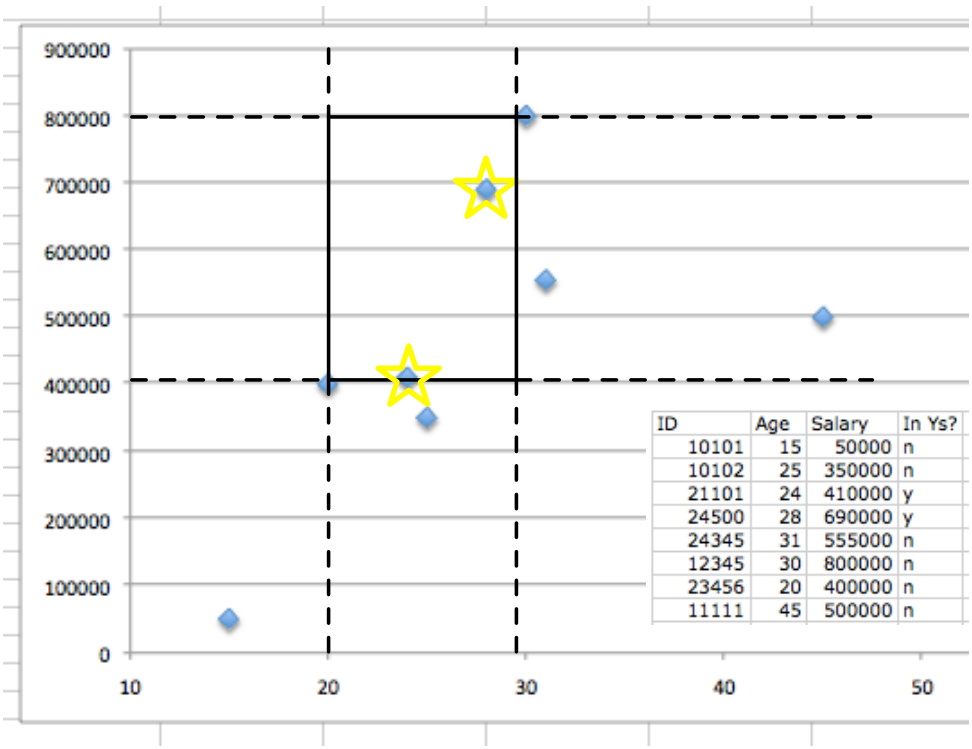
What is algorithmic about it? The way how we will find those points. First, what we need to implement is a *range search*, which is specified as follows (taken from Skiena's manual):

> **Input description:** A set S of n points in E^d and a query region Q.
> **Problem description:** What points in S lie in Q?

In other scenarios, there can be many dimensions, here we have just two: age and salary, and our 'slice' is a simple rectangle, making the search an orthogonal range query.

The laborious way to code it, is to ask for each input value pair whether it is within the specified range for A, and if so, then whether it is within the specified range for S. For small numbers and few dimensions, this is fine, but with larger numbers, you'll run out of the allotted time to compute the answer. There is a faster way of designing and implementing this: use a k-d tree (a space-partitioning data structure that organizes the points in a k-dimensional space), and then do the range search over that. The reason why this is faster than looping through all points, is thanks to the difference in data structure—list v. k-d tree—as the latter divides the range of a domain in half at each level of the tree, hence chopping the search space. In the implementation, first create the data structure, then specify the query with the ranges.

A slightly extended set of data points is depicted in Fig. 1, which also includes a few borderline cases you should check against ($(20, 400000) \notin Y_s$ and $(30, 800000) \notin Y_s$ because of the "<").



ID	Age	Salary	In Ys?
10101	15	50000	n
10102	25	350000	n
21101	24	410000	y
24500	28	690000	y
24345	31	555000	n
12345	30	800000	n
23456	20	400000	n
11111	45	500000	n

Figure 1: Visualisation of the relevant section of the 2-d space, the data set (insert, right), the specified range (solid rectangle). The two points that are in the specified range are highlighted with a star.