Course
CS-E3190
Course materials
Your points

This course has already ended.
The latest instance of the course can be found at: Principles of Algorithmic Techniques: 2023 Autumn

« 4.1 Implementing iterative algorithms Course materials Lecture and Exercise Set 2 - Recursion »
CS-E3190 / Programming Exercise 1 - Stable Matching / 4.2 The stable matching problem

The stable matching problem

Exercise: Stable matching

This exercise asks you to implement an algorithm that computes a stable matching between n applicants and n positions. Each applicant $i = 0, 1, \ldots, n-1$ has preferences for each position $j = 0, 1, \ldots, n-1$, and vice versa. These preferences are given as input in two n^2-element arrays `a_pref` and `b_pref` so that applicant $i = 0, 1, \ldots, n-1$ prefers the n available positions in order

> `a_pref[i*n+0] > a_pref[i*n+1] > ... > a_pref[i*n+n-1]`,

where `a_pref[i*n+0]` is the most preferred position, `a_pref[i*n+1]` is the next most preferred position, and so forth until the least preferred position `a_pref[i*n+n-1]`. Dually, position $j = 0, 1, \ldots, n-1$ prefers the applicants in order

> `b_pref[j*n+0] > b_pref[j*n+1] > ... > b_pref[j*n+n-1]`,

where `b_pref[j*n+0]` is the most preferred applicant, `b_pref[i*n+1]` is the next most preferred applicant, and so forth until the least preferred applicant `b_pref[i*n+n-1]`.

The stable matching is output via the array `s` such that each applicant $i = 0, 1, \ldots, n-1$ is matched to the position `s[i]`. Accordingly, the entries `s[0],s[1],...,s[n-1]` must form a permutation of the integers $0, 1, \ldots, n-1$. Furthermore, for all $i, j = 0, 1, \ldots, n-1$ such that $j \neq s[i]$ it must be that applicant i prefers position $s[i]$ over position j or position j prefers applicant $s^{-1}[j]$ over applicant i; indeed, otherwise the pair (i, j) is unstable. Here we write s^{-1} for the inverse permutation of s.

Your task in this exercise is to complete the subroutine

```
void solver(int n, const int *a_pref, const int *b_pref, int *s)
```

which should compute the array `s` from the given inputs `n`, `a_pref`, and `b_pref`. To locate the subroutine quickly, you can search for "`???`" in the source file. You may assume that $0 \leq n \leq 2048$. The source file contains many subroutines that you may find useful in preparing your solution. For example, the subroutine `perm_inv` computes the inverse of a permutation.

Grading. This exercise awards you up to 10 points in the course grading. The number of points awarded is the maximum points times the number of tests passed over the total number of tests, rounded up. To successfully complete a test, your implementation must use no more than 3 seconds of wall clock time and 100 MiB of memory. Each test will in general require the successful solution of one or more problem instances. In each batch of scaling tests, the first failed test will cause all subsequent tests in the batch to be skipped.

Points **10 / 10** My submissions **3 / 10**
Deadline Monday, 26 September 2022, 20:00
To be submitted alone

⚠ This course has been archived (Friday, 30 December 2022, 16:00).

Stable matching

1. Download the template source file stable-matching.cpp.
2. Study the problem description above.
3. Complete your solution in the template.
4. Test your solution with the unit tests available in the template.
   Recall that the Scratchpad is available for compiling and running your solution.
5. Submit your solution for grading using the dialog below.
   See here for a quick guide to the grader output.

stable-matching.cpp

Choose File No file chosen

Submit

« 4.1 Implementing iterative algorithms Course materials Lecture and Exercise Set 2 - Recursion »