Matching mechanisms for kidney transplantations

Tristan François

January 13, 2021

1 Two Simple Approaches

Question 1

Algorithm 1: Direct Donation

Input: A number n of patient and, for all patients $i \in \mathbb{N}$, their set of compatible donors $K_i \subseteq [\![1, n]\!]$.

Output: A set of pairs donor-patient pairs c and a waiting list w.

```
begin
    c = ∅;
    w = [];
    for i = 1 to n do
        if i ∈ K_i then
            c = c ∪ {(i, i)};
        else
            w.add(i)
    return c, w
```

The implementation in python is pretty straightforward. The python function takes as parameter an integer n and an integer set array of size n. The use of python set allows to test the if condition in constant times. The only difference with pseudo-code is the use of a list rather than a set for the variable c. This choice was made to avoid the unnecessary heaviness of the set data structure, but does not change the complexity.

Question 2

Algorithm 2: Greedy Matching

Input: A number n of patient, a strict priority list of the patients U and, for all patients $i \in \mathbb{N}$, their set of compatible donors $K_i \subseteq [\![1, n]\!]$ and their strict preference relation P_i over $K_i \cup \{k_i, w\}$. The priority list U starts with the patient with the highest priority and goes decreasingly to the lowest priority.

Output: A set of pairs donor-patient pairs c and a waiting list w.

begin
  $c = \emptyset$;
  $w = []$;
  $matched = []$;
  for $i = 1$ **to** n **do**
    matched.add(False);
  return c, w

2 Efficient stategy-proof exchange mechanism

Question 3

Let (k_1, t_1) be a pair of donor-patient and assume there is no cycle. Consider the chain starting from (k_1, t_1). Since there is a finite number n of donor-patient pairs and there is no cycle, the chain cannot be infinite. Finally, since the only way for a chain to stop is that w belong to the chain, (k_1, t_1) is indeed a tail of a w-chain.

Thus either there exists a cycle, or each pair is the tail of some w-chain.