Advanced information, computation, communication I
EPFL - Fall semester 2021-2022

Week 6

October 29, 2021

1 Open Questions

Exercise 1. (**)

```
function f1() {
  x=0
  i=1
  while (i ≤ n) {
    x=x+1
    i=x+x
  }
  a=x
}
```

```
function f2() {
  y=0
  j=1
  while (j ≤ n) {
    y=y+1
    j=y*y
  }
  b=y
}
```

After execution of the two program fragments `f1` and `f2`, it is the case that

✓ $a \approx \frac{n}{2}$, $b \approx \sqrt{n}$.

◯ $a \approx n$, $b \approx \log_2(n)$.

◯ $a \approx \frac{n}{2}$, $b \approx \log_2(n)$.

◯ $a \approx n$, $b \approx \sqrt{n}$.

Solution. For the first program fragment, x acts as a variable that counts the number of times the "while" is executed, where the while terminates as soon as twice the counter is $> n$, i.e., as soon as $2x > n$. The final x is therefore about $\frac{n}{2}$.

For the second program fragment the argument is identical: y acts as a variable that counts the number of times the "while" is executed, where the while terminates as soon as the square of the counter y is $> n$, i.e., as soon as $y^2 > n$. The final y is therefore about $\sqrt{n}$.

Exercise 2. (**) The three algorithms below sort the input sequence $a_1, \ldots, a_n$ in ascending order.

Algorithm 1 Bubble Sort

```
for i ← 1 to n − 1 do
  for j ← 1 to n − i do
    if a_j > a_{j+1} then
      swap a_j and a_{j+1}
    end if
  end for
end for
```

Algorithm 2 Selection Sort

```
for i ← 1 to n − 1 do
  min ← i + 1
  for j ← i + 1 to n do
    if a_min > a_j then
      min ← j
    end if
  end for
  if a_i > a_min then
    swap a_i and a_min
  end if
end for
```

Algorithm 3 Insertion Sort

```
for j ← 2 to n do
  i ← 1
  while a_j > a_i and i < j do
    i ← i + 1
  end while
  m ← a_j
  for k ← 0 to j − i − 1 do
    a_{j−k} ← a_{j−k−1}
  end for
  a_i ← m
end for
```

Use Bubble Sort, Selection Sort and Insertion Sort to sort the following sequence:

`9, 12, -43, 20, -2, 3, 7, 28, 19`

Solution.

1. **Bubble Sort**:

   We have a sequence of length 9. The procedure passes through the entire sequence, compares each pair of consecutive numbers (x_i, x_{i+1}) and swaps them if $x_i > x_{i+1}$. Hence, the resulting sequences after each pass(8 in total) are the following:

   ```
    9, -43, 12, -2, 3, 7, 20, 19, 28.
   -43, 9, -2, 3, 7, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   -43, -2, 3, 7, 9, 12, 19, 20, 28.
   ```

   Apparently, the last few passes are not actually doing anything since the sequence has already been sorted. We'll see a way of optimizing this away in one of the exercises below.

2. **Selection Sort**:

   (a) $i = 1$

   $$\min \leftarrow \text{argmin}\{a_2, ..., a_9\} = 3, \quad a_{\min} = a_3 = -43$$

   $$a_1 = 9 > a_{min} = -43 \implies \text{swap } a_1, a_3$$

   resulting in `-43, 12, 9, 20,-2, 3, 7, 28, 19`

   (b) $i = 2$

   $$\min \leftarrow \text{argmin}\{a_3, ..., a_9\} = 5, \quad a_{\min} = a_5 = -2$$

   $$a_2 = 12 > a_{min} = -2 \implies \text{swap } a_2, a_5$$

   resulting in `-43, -2, 9, 20, 12, 3, 7, 28, 19`

   (c) $i = 3$

   $$\min \leftarrow \text{argmin}\{a_4, ..., a_9\} = 6, \quad a_{\min} = a_6 = 3$$

   $$a_3 = 9 > a_{min} = 3 \implies \text{swap } a_3, a_6$$

   resulting in `-43, -2, 3, 20, 12, 9, 7, 28, 19`

(d) $i = 4$

$$\min \leftarrow \text{argmin}\{a_5, ..., a_9\} = 7, \quad a_{\min} = a_7 = 7$$

$$a_4 = 20 > a_{min} = 7 \implies \text{swap } a_4, a_7$$

resulting in `-43, -2, 3, 7, 12, 9, 20, 28, 19`

(e) $i = 5$

$$\min \leftarrow \text{argmin}\{a_6, ..., a_9\} = 6, \quad a_{\min} = a_6 = 9$$

$$a_5 = 12 > a_{min} = 9 \implies \text{swap } a_5, a_6$$

resulting in `-43, -2, 3, 7, 9, 12, 20, 28, 19`

(f) $i = 6$

$$\min \leftarrow \text{argmin}\{a_7, ..., a_9\} = 9, \quad a_{\min} = a_9 = 19$$

$$a_6 = 12 < a_{min} = 9 \implies \text{do nothing}$$

resulting in `-43, -2, 3, 7, 9, 12, 20, 28, 19`

(g) $i = 7$

$$\min \leftarrow \text{argmin}\{a_8, a_9\} = 9, \quad a_{\min} = a_9 = 19$$

$$a_7 = 20 > a_{min} = 19 \implies \text{swap } a_7, a_9$$

resulting in `-43, -2, 3, 7, 9, 12, 19, 28, 20`

(h) $i = 8$

$$\min \leftarrow \text{argmin}\{a_9\} = 9, \quad a_{\min} = a_9 = 20$$

$$a_8 = 28 > a_{min} = 20 \implies \text{swap } a_8, a_9$$

resulting in `-43, -2, 3, 7, 9, 12, 19, 20, 28`

3. **Insertion Sort**:
(NOTE: the maximum of a set $\max\{a_1, ..., a_n\}$ is defined to be zero if the set is empty, i.e. $\max \emptyset = 0$)

(a) $j = 2$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 2$$

Nothing to insert. The sequence remains

`9, 12, -43, 20, -2, 3, 7, 28, 19.`

(b) $j = 3$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = \max \emptyset + 1 = 1$$

Shift a_1, a_2 forward. $a_1 \leftarrow a_3$. The sequence becomes

`-43, 9, 12, 20, -2, 3, 7, 28, 19.`

(c) $j = 4$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 4$$

Nothing to insert. The sequence remains

`-43, 9, 12, 20, -2, 3, 7, 28, 19.`

(d) $j = 5$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 2$$

Shift a_2, a_3, a_4 forward. The sequence becomes

`-43, -2, 9, 12, 20, 3, 7, 28, 19.`

(e) $j = 6$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 3$$

Shift a_3, a_4, a_5 forward. The sequence becomes

`-43, -2, 3, 9, 12, 20, 7, 28, 19.`

(f) $j = 7$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 4$$

Shift a_4, a_5, a_6 forward. The sequence becomes

`-43, -2, 3, 7, 9, 12, 20, 28, 19.`

(g) $j = 8$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 8$$

Nothing to insert. The sequence remains

`-43, -2, 3, 7, 9, 12, 20, 28, 19.`

(h) $j = 9$

$$i \leftarrow \max\{i \mid a_j > a_i \text{ and } i < j\} + 1 = 7$$

Shift a_7, a_8 forward. The sequence becomes

`-43, -2, 3, 7, 9, 12, 19, 20, 28.`

Exercise 3. (*) Recall the stable maximum matching problem/algorithm introduced in class (Session 34). Is a stable maximum matching unique? Either prove that every stable maximum matching is unique, or disprove it with a counterexample.

Solution. In general, stable matchings are not unique. We show this by giving a counterexample. Consider the simplest case, with two students and two universities and the following preferences:

Student	Most preferable	least preferable
Giulia	ETHZ	EPFL
Charlotte	EPFL	ETHZ

University	Most preferable	least preferable
EPFL	Giulia	Charlotte
ETHZ	Charlotte	Giulia

It is not hard to check that both of the following are stable matchings:

Student	**University**
Giulia	EPFL
Charlotte	ETHZ

Student	**University**
Giulia	ETHZ
Charlotte	EPFL

In fact, one is Student-optimal and the other is University-optimal, so they have to be stable.

Exercise 4. (**) Let $\{A, B, C, D\}$ be a set of men, and $\{\alpha, \beta, \gamma, \delta\}$ a set of women. We want to match up men and women using the Gale-Shapley algorithm in two different ways. The preferences of men and women are given in the following lists, going from most preferable on the left to least preferable on the right.

Men	1st	2nd	3rd	4th
A	γ	δ	β	α
B	δ	γ	α	β
C	α	γ	β	δ
D	β	δ	α	γ

Women	1st	2nd	3rd	4th
α	D	A	B	C
β	C	B	A	D
γ	C	B	A	D
δ	D	A	B	C

1. If the men propose, and women accept/reject, what is the matching after the algorithm terminates?
   Solution. After all men propose to their 1st prefered woman, since they are all different, they accept and the algorithm terminates with the following matching:

   $$A - \gamma, B - \delta, C - \alpha, D - \beta$$

2. If the women propose, and men accept/reject, what is the matching after the algorithm terminates?
   Solution.
   Firstly α proposes to D, and β proposes to C, and they accept because it is the first proposal.

   $$\alpha - D, \beta - C; \quad \gamma, \delta, A, B \text{ unmatched}$$

   Then γ proposes to C. Since C prefers γ more than β, β gets rejected, and C accepts γ instead.

   $$\alpha - D, \gamma - C; \quad \beta, \delta, A, B \text{ unmatched}$$

   Now δ proposes to D, and again because of preferences of D, α gets rejected and D accepts δ instead.

   $$\gamma - C, \delta - D; \quad \alpha, \beta, A, B \text{ unmatched}$$

   α and β are now unmatched, so they proposed to their second preference, A and B respectively. Since they are free, A and B accept and the algorithm terminates with:

   $$\alpha - A, \beta - B, \gamma - C, \delta - D$$

3. Who is the best possible (stable) valid partner for "α"?

   Solution. First we will show that a matching where α is paired with D is unstable. This is because D prefers δ to α, and δ will prefer D whoever she is matched with (D is her 1st preference). So α cannot be matched with D. On the other hand, α can be matched with A, as we have seen from 2. So the best matching for α is A.

   One can also simply argue that the solution found in 2 is Women-optimal, so no women can do better in any stable matching than the one found in 2, so A is the best partner than α can find.

Exercise 5. (*) (Hint. The algorithmic steps are fairly similar for all four. Only showing steps for the first will suffice.) Use the cashier's algorithm to make change using quarters, dimes, nickels, and pennies for:

1. 87 cents.
   Solution.

   (a) First pass (quarters) $c_1 = 25$
   We note that $n \geq 25$, thus we add 1 quarter and decrease the total amount by 25.

   $$d_1 = 1, \quad n = n_{\text{old}} - 25 = 87 - 25 = 62$$

   We note that $n \geq 25$, thus we add 1 quarter and decrease the total amount by 25.

   $$d_1 = 1 + 1 = 2, \quad n = n_{\text{old}} - 25 = 62 - 25 = 37$$

   We note that $n \geq 25$, thus we add 1 quarter and decrease the total amount by 25.

   $$d_1 = 2 + 1 = 3, \quad n = n_{\text{old}} - 25 = 37 - 25 = 12$$

   We note n $<$ 25 and thus the first pass ends.

   (b) Second pass (dimes) $c_2 = 10$
   We note that $n \geq 10$, thus we add 1 dime and decrease the total amount by 10.

   $$d_1 = 3, \quad d_2 = 1, \quad n = n_{\text{old}} - 10 = 12 - 10 = 2$$

   We note n $<$ 10 and thus the second pass ends.

(c) Third pass (dimes) $c_3 = 5$
We note n < 5 and thus the third pass ends.

(d) Fourth pass (pennies) $c_4 = 1$
We note that $n \geq 1$, and thus we add 1 penny and decrease the total amount by 1.

$$d_1 = 3, \quad d_2 = 1, \quad d_4 = 1, \quad n = n_{\text{old}} - 1 = 2 - 1 = 1$$

We note that $n \geq 1$, and thus we add 1 penny and decrease the total amount by 1.

$$d_1 = 3, \quad d_2 = 1, \quad d_4 = 2, \quad n = n_{\text{old}} - 1 = 1 - 1 = 0$$

We note n < 1 and thus the fourth pass ends.
Conclusion: 87 cents is 3 quarters, 1 dime, 0 nickels and 2 pennies.

2. 49 cents.
Solution. 49 cents is 1 quarter, 2 dimes, 0 nickels and 4 pennies.

3. 99 cents.
Solution. 99 cents is 3 quarters, 2 dimes, 0 nickels and 4 pennies.

4. 33 cents.
Solution. 33 cents is 1 quarter, 0 dimes, 1 nickel and 3 pennies.

Exercise 6. (*) Describe an algorithm that determines whether a function f from a finite set $\{a_1, a_2, ..., a_n\}$ to its image $\{f(a_1), f(a_2), ..., f(a_n)\}$ is one-to-one.
Solution.

```
for i = 1 to n
    for j = i+1 to n
        if (a_i ≠ a_j and f(a_i) = f(a_j))
            return False
return True
```

Exercise 7. (***) Adapt the bubble sort algorithm so that it stops when no more swaps is required. Express this more efficient version of the algorithm in pseudocode.
Solution.

Algorithm 4 Better Bubble Sort

```
for i ← 1 to n − 1 do
  sorted ← TRUE
  for j ← 1 to n − i do
    if a_j > a_{j+1} then
      swap a_j and a_{j+1}
      sorted ← FALSE
    end if
  end for
  if sorted then
    break      // no further swap needed; already sorted
  end if
end for
```

Exercise 8. (*) Two strings are anagrams if each can be formed from the other by rearranging its characters. Devise an algorithm to determine whether two strings are anagrams.
Solution. Sort both strings using any sorting algorithm (the ”value” of each character can be defined

to be its position in the alphabet). The two original strings are anagrams if the two sorted strings are exactly the same.

Exercise 9. (*) (Book Chapter 3.1 Exercise 5) Describe an algorithm that takes as input a list of n integers in non-decreasing order, $a_1 \leq a_2 \leq ... \leq a_n$, and produces the list of all values that occur more than once.
Solution.

Algorithm 5 Find Duplicates

```
curr ← a_1
count ← 1
for i ← 2 to n do
  if a_i == curr then
    count ← count + 1
  else
    if count > 1 then
      OUTPUT curr
    end if
    curr ← a_i
    count ← 1
  end if
end for
if count > 1 then
  OUTPUT curr        // remember to flush out the last value, if needed
end if
```

2 Exam Questions

Exercise 10. (**)

$$\begin{aligned} L_{x_1} &= (y_3, y_1, y_2) & L_{y_1} &= (x_2, x_1, x_3) \\ L_{x_2} &= (y_2, y_3, y_1) & L_{y_2} &= (x_1, x_3, x_2) \\ L_{x_3} &= (y_1, y_2, y_3) & L_{y_3} &= (x_3, x_2, x_1) \end{aligned}$$

(*français*) Soit L_x pour $x \in X = \{x_1, x_2, x_3\}$ la liste de préférence de x donnée ci-dessus et soit L_y pour $y \in Y = \{y_1, y_2, y_3\}$ la liste de préférence de y donnée ci-dessus. Le couplage $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$ est

(*English*) Let L_x for $x \in X = \{x_1, x_2, x_3\}$ be the preference list of x as given above and let L_y for $y \in Y = \{y_1, y_2, y_3\}$ be the preference list of y as given above. The matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$ is

○ instable. / unstable.

○ stable et optimal pour Y. / stable and Y-optimal.

○ stable et optimal pour X. / stable and X-optimal.

✓ stable, mais n'est pas un couplage stable optimal pour X ou pour Y. / stable but not a stable matching that is X- or Y-optimal.

Solution.
In the following matching: $\{(x_1, y_3), (x_2, y_2), (x_3, y_1)\}$, all $x \in X$ are matched with their optimal choices and therefore the matching $\{(x_1, y_3), (x_2, y_2), (x_3, y_1)\}$ is stable and X-optimal. Because this matching $\{(x_1, y_3), (x_2, y_2), (x_3, y_1)\}$ is different from the given matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$, the matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$ is not X-optimal, so that the third answer is not correct. (Note that in the matching $\{(x_1, y_3), (x_2, y_2), (x_3, y_1)\}$ all $y \in Y$ are matched to their least favorite choices.)

In the following matching: $\{(x_1, y_2), (x_2, y_1), (x_3, y_3)\}$, all $y \in Y$ are matched with their optimal choices and therefore the matching $\{(x_1, y_2), (x_2, y_1), (x_3, y_3)\}$ is stable and Y-optimal. Again, because this matching $\{(x_1, y_2), (x_2, y_1), (x_3, y_3)\}$ is different from the given matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$, the matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$ is not Y-optimal, so that the second answer is not correct. (Note that in the matching $\{(x_1, y_2), (x_2, y_1), (x_3, y_3)\}$ all $x \in X$ are matched to their least favorite choices.)

Actually, in the matching $\{(x_1, y_1), (x_2, y_3), (x_3, y_2)\}$, no $x \in X$ or $y \in Y$ is matched with its optimal or with its worst choice. So, in principle, there could be a quite a lot of room to move things around:

in (x_1, y_1)**:**

x_1 would prefer to be matched with y_3, but y_3 prefers its current x_2 to x_1. Because there is no other $y \in Y$ that x_1 would prefer to y_1, the pair (x_1, y_1) is stable from x_1's point of view.

y_1 would prefer to be matched with x_2, but x_2 prefers its current y_3 to y_1. Because there is no other $x \in X$ that y_1 would prefer to x_1, the pair (x_1, y_1) is stable from y_1's point of view as well.

It follows that the pair (x_1, y_1) is stable.

in (x_2, y_3)**:**

x_2 would prefer to be matched with y_2, but y_2 prefers its current x_3 to x_2. Because there is no other $y \in Y$ that x_2 would prefer to y_3, the pair (x_2, y_3) is stable from x_2's point of view.

y_3 would prefer to be matched with x_3, but x_3 prefers its current y_2 to y_3. Because there is no other $x \in X$ that y_3 would prefer to x_2, the pair (x_2, y_3) is stable from y_3's point of view as well.

It follows that the pair (x_2, y_3) is stable.

in (x_3, y_2)**:** a similar argumentation can be used again, or we can simply argue that the pair (x_3, y_2) must be stable because its instability would imply instability of another pair which is impossible because all other pairs are stable.

It follows that only the fourth answer is correct.

Exercise 11. (**)

Charlotte, Giulia, Kevin and Patrick are starting university next year. They have applied to EPFL, ETHZ, USI and HSG, and their preferences are listed as follows:

Student	most preferred	$\longrightarrow$	$\longrightarrow$	least preferred
Patrick	ETHZ	EPFL	USI	HSG
Giulia	EPFL	USI	ETHZ	HSG
Charlotte	USI	ETHZ	EPFL	HSG
Kevin	HSG	ETHZ	EPFL	USI

The universities, on the other hand, have their own lists of preferred students

University	most preferred	$\longrightarrow$	$\longrightarrow$	least preferred
EPFL	Giulia	Charlotte	Patrick	Kevin
ETHZ	Giulia	Patrick	Charlotte	Kevin
USI	Patrick	Charlotte	Giulia	Kevin
HSG	Patrick	Giulia	Charlotte	Kevin

Which of the matchings below is not stable?

- ◯ (Kevin, EPFL) (Charlotte, USI)
- ✓ (Kevin, ETHZ) (Patrick, HSG)
- ◯ (Kevin, HSG) (Giulia, EPFL)
- ◯ (Kevin, USI) (Patrick, ETHZ)

Solution. Notice in the second answer: Patrick prefers ETHZ over HSG and ETHZ also prefers Patrick over Kevin. This gives them incentive to form the new pair (Patrick, ETHZ), and leave the other two entities with (Kevin, HSG).