Homework #4
Greedy Algorithms, Dynamic Programming and NP-Completeness
Presentation: 3-4 Jan, 2019

- This is an oral presentation assignment. You will present and discuss your answers with the professor using a white board (you may bring auxiliary written material to aid you).
- You should work in **groups of three**. At some point before the 3nd of January (thursday) at 11:59am, your group should sign up for a 30-minutes time slot on the signup sheet that you will receive by email.
- Each person in the group must be able to present every problem. The professor will select who presents which problem. The other group members may assist the presenter in case of need.
- You are not required to hand anything in at your presentation, but you may if you choose.

---

1. Greedy cakes

You have available a set of n cakes, each with a certain satisfaction value s_i and an expiration date e_i (meaning it must be consumed at most in e_i days starting from today). You can only eat one cake per day (or will you get sick) and you must decide what is the best strategy to eat in order to **maximize the total satisfaction** (sum of the values of the cakes you end up eating). This means you must decide which cake you should eat in each day, starting in day 1 (remember, you can only eat one cake per day, at most).

To give an example, imagine you have the following set of $n = 9$ cakes:

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
s_i	10	20	10	15	14	40	18	5	20
e_i	5	3	3	3	5	4	5	7	5

Because the maximum expiration date is day 7, we need to find which cake to eat in days $1 \ldots 7$. Cake i can only be eaten in day j if $j \leq e_i$, because of the expiration date. Suppose we indicate a possible allocation of cakes to days with $\{c_1, \ldots, c_n\}$, where c_j is the index of the cake we eat in day j (with " $-$ " indicating that no cake is eaten on that day). Here are some possible valid allocations of cakes to days:

- $\{1, 2, 3, 5, 7, 8, -\}$ which would have total satisfaction $77 = 10 + 20 + 10 + 14 + 18 + 5$
- $\{1, 2, 3, 6, 7, -, 8\}$ which would have total satisfaction $103 = 10 + 20 + 10 + 40 + 18 + 5$
- $\{2, 4, 6, 7, 9, -, 8\}$ which would have total satisfaction $118 = 20 + 15 + 40 + 18 + 20 + 5$

For this particular instance, 118 is indeed the optimal total satisfaction, and no other allocation would give a better value. And for a general case, can you devise an optimal strategy to follow?

1. Explain a **greedy strategy** that will give origin to an optimal solution, that is, one that maximizes the total satisfaction.

   Hint: you want to maximize satisfaction, so your strategy should take that into account.

2. **Prove** that the strategy you gave is optimal.

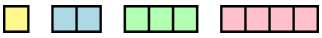
   Hint: you must show that no other solution would be better. You can, for instance, use the "exchange "argument, or the "stay-ahead" strategy.

3. Explain an **implementation** of your greedy strategy that runs in $\mathcal{O}(n \log n)$ time.

   Hint: what operations do you need? what algorithms and/or data structures would give you the needed execution time guarantees?

2. Counting ways...

You have available a set of pieces of sizes $1\times1, 1\times2, 1\times3, \ldots, 1\times s$ (with $s \geq 2$). For instance, if $s = 4$, you would have available the following 4 different pieces:



You should explain a **dynamic programming (DP) solution** for each of the following two problems. You should characterize your DP state, and indicate a recurrence that uses solutions of subproblems. You should also indicate a way to compute your DP table (referring the corresponding complexity) and be prepared to show the table values for a given instance.

So that you can verify your answers, it is advisable you implement your DP solutions on any programming language. You can check their validity by seeing if your are giving the correct answer for the example instances.

1. **$1 \times n$ board:** imagine you have available a board of size $1 \times n$ (with $n \geq 2$). You want to know on how many different ways you could fill up the entire board using the pieces. For instance, if $s = 4$ as before, and $n = 4$, you would have 8 different ways of filling the board:

   Can you give a **DP solution** for a general case given s and n? Be prepared to exemplify your DP table for the case where $s = 3$ and $n = 8$.

   Hint: After putting one piece, what part of the board is left to compute? It is a smaller instance of the original problem?

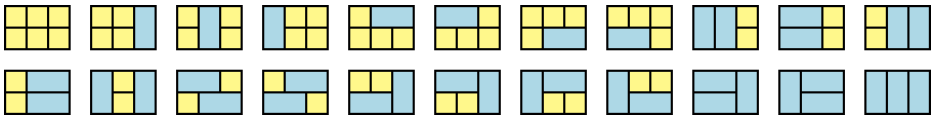
   More example instances:

   - For the case $s = 2$ and $n = 4$, there are 5 different ways to fill up the board.
   - For the case $s = 3$ and $n = 6$, there are 24 different ways to fill up the board.
   - For the case $s = 4$ and $n = 10$, there are 401 different ways to fill up the board.
   - For the case $s = 5$ and $n = 15$, there are 13624 different ways to fill up the board.

2. **$2 \times n$ board:** imagine you now have a board of size $2 \times n$ (with $n \geq 2$). You want to know on how many different ways you could fill up the entire board using the pieces, knowing that pieces can be rotated 90 degrees. For instance, if $s = 2$ and $n = 2$, you would have 7 different ways of filling the board:

   In the case where $s = 2$ and $n = 3$, you would have 22 different ways of filling up the board:

   Hint: What is different here from the $1 \times n$ case? Can you reuse the solutions from that smaller case? How can you divide a configuration of a board into smaller instances of the same problem? Be careful not to count more than once the same configuration...

   More example instances:

   - For the case $s = 2$ and $n = 4$, there are 71 different ways to fill up the board.
   - For the case $s = 3$ and $n = 6$, there are 1596 different ways to fill up the board.
   - For the case $s = 4$ and $n = 10$, there are 574903 different ways to fill up the board.
   - For the case $s = 5$ and $n = 15$, there are 920649043 different ways to fill up the board.

3. When is your exam?

Every year, universities around the world face the same problem: given a list of n courses, m students, the list of courses each student is taking and an integer k representing the duration (in days) of the exam phase, is there an exam schedule consisting of k dates so that there are no conflicts? (i.e. is it possible to find an assignment of courses to days so that no student has more than one exam on the same day?)

This apparently simple problem is more difficult than it appears; every known solution requires an exponential execution time. Can you **show that it is NP-Complete**? More specifically, can you show that this problem is as difficult as the Clique-Cover problem?

Hint: In order to prove that a problem is NP-Complete, you must prove:

1. It is inside the NP class.
2. There exists a mapping function from another problem known to be NP-hard to your problem, so that a polynomial time solution to your problem would imply a polynomial time solution to the original problem.
3. This mapping can be done in polynomial time.