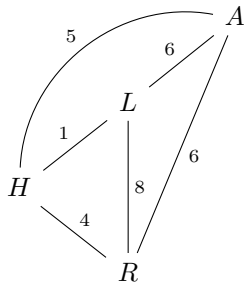
Question 1. *The Itinerant*

Recall that a *circuit* in a graph is a path along edges that begins and ends in the same vertex. A *Hamiltonian circuit* is a circuit that visits every vertex, except for the starting vertex, exactly once. Consider the following weighted undirected graph G



a. Starting in the vertex L, give every possible Hamiltonian circuit of the graph G together with their weight. What is the shortest circuit?

b. Does every undirected graph have a Hamiltonian circuit? If so justify your answer with a general argument, if not provide a counterexample.

c.* Provide an algorithm in pseudocode for generating every Hamiltonian circuits for any completely connected[1] undirected graph.

d. What is the (worst-case) time complexity of your algorithm in **c.**?

Question 2. *Bike rack technician*

Moloch is working as a technician in an OV-Fietsenstalling in a modern train station in the Netherlands. The indoor bike garage contains parallel bike racks and at the end of each bike rack there is a helpful indicator light which lights red or green to indicate whether the rack is full or not. This is done by a set of simple black and white cameras attached to the bike rack. However, one day the indicator light for a couple of bike racks stops working properly; bike racks which are full still display a green indicator which is preventing customers from quickly finding a place for their bike. At night when the garage is closed Moloch is tasked with troubleshooting these faulty racks. Thankfully this can be done remotely from a computer where he can override camera signal to isolate any malfunctioning cameras.

You may assume there is a list-object `bike_rack` with a setter-method for the override signal, and the status of the indicator light is accessed through an boolean value `rack_free`.

`bike_rack[n].set(True)` - Overrides the n:th camera signal to signal occupied,
`bike_rack[n].set(None)` - Removes the override signal,
`rack_free()` - Returns True if the indicator light is green, i.e. rack is not full; and False if the indicator light is red.

a. Assume the bike rack is full of bikes and `rack_free` is True. Design and provide pseudocode for a Brute-force algorithm that detects and outputs which cameras are broken.

b*. One of the bike racks that Moloch is troubleshooting is behaving strangely. Sometimes when only a few bikes are parked the indicator will turn red, only to turn green again after more bikes are placed. Only to switch back to red with a few more bikes are placed or removed. Despite this Moloch is determined to figure out which combinations trigger the signal to show green or red. With the help of a colleague they figure out that the issue is within the logic unit that updates the `rack_free` variable.

For this exercise, assume the bike rack is completely empty, *all cameras are functioning properly*, the issue is in the logic unit which computes the `rack_free`. Design a Brute-force pseudocode algorithm which outputs every combination that result in `rack_free()` to yield True.

c. Before starting these algorithms Moloch wants to be sure he will get to go home at some point. What is the time complexity (big-O) of the algorithms you provided in **a.** and **b.**?

[1] Recall that a graph is completely connected if every two vertices share an edge

Question 3. *Split-Nim*

We consider a new variation on the two player game Nim. Initially, the board consists of a single stack with n chips. There are two players (Sven and Gerard) who alternate turns, and as usual the player to take the last chip wins the game. A player may either (i) split a stack of chips in two **unequal** stacks of chips, or (ii) remove 1 or 2 chips from the same stack. That is, a stack with 2 chips can not be split. For example: Consider the game state whith 2 stacks, one with 4 chips and one with 5 chips. In that case, there are seven possible moves, of which three involve splitting a stack (check this).

a. What are the states and what the actions?

b. Construct the state space for the game with $n = 4$. Assume that Sven starts. Is this game winning for the starting player? Recall that 'winning' means that the there exists a strategy so the player can always win, regardless of the moves of the opponent.

Hint: Do not elaborate on states that you already written out elsewhere, simply write down who wins.

Question 4. *Mountain Goats*

On a very thin mountain path, flanked by a steep canyon and an almost vertical mountainside, two herds of goats meet. One herd of n black goats, and one of n white goats. From our vantage point, the black goats are on the left and the white goats on the right. They stop with exactly one goat-length of empty space in the middle. This situation, for $n = 4$, can be represented as: `zzzz_wwww`.

Any goat that is in front of an empty space can move into that space. There is not enough space to cross sideways. However, the goats can jump over goats of another color if there is an empty space for the goat to land. Goats will never jump over a goat of the same color.

The goats are very stubborn and refuse to turn around or either jump or walk backwards. The black goats only move from left to right and the white goats only move from right to left. The goal is for all goats to move to the other side. For example, a final situation for $n = 5$ could be `wwwww_zzzzz`. As there is only one empty spot, this procedure occurs on a space of $2n + 1$ goat lengths.

NB: Goats of the same color can make several moves after each other, for instance the following sequence is possible:
`zzzz_wwww` $\longrightarrow$ `zzz_zwwww` $\longrightarrow$ `zz_zzwwww` $\longrightarrow$ `z_zzzwwww`

a. What are the states and what are the actions?

b. For $n = 2$ there are only two initial actions possible: `zz_ww` $\longrightarrow$ `zzw_w` or `zz_ww` $\longrightarrow$ `z_zww`
Construct the state-space for $n = 2$. How many steps do you need to get from start to final position.

c. Give an explanation and an estimate of an upper bound for the **number of 'moves'** (walking into the empty space or jump) that have to be performed in order to reach a final position.

d. For every n there are always two solutions: one starting with a black goat and one with a white goat. Consider the case for $n = 3$. Reason how you could reach the final state from the initial state with a black goat.

Hint: There are 15 moves in total; these are the first moves: `zzz_www` $\longrightarrow$ `zz_zwww` (*) $\longrightarrow$ `zzwz_ww` (**) $\longrightarrow$ `zzwzw_w`. From (*) there are two possible moves: walking leads to `z_zzwww`, after which the 3 white goats are all blocked. Therefore, we should start with a jump move. From (**) are two possible walking moves: one leads to `zzw_zww`, and continuing leads to a blocking. Therefore chose the other ...

e*. A possible upper bound for the **number of unique states** could be given by $O\left(\binom{2n}{n} \cdot (2n+1)\right)$.[2] Justify this estimate, can you provide a lower upper bound?

[2]The binomial coefficient $\binom{n}{k}$, pronounced "n choose k", is the number of ways we can choose k elements from a set of n choices (without order), it is a shorthand for the expression $\frac{n!}{(n-k)!k!}$. For example, picking a soccer team of 3 member from a group of 5 can be done in $\binom{5}{3} = \frac{5!}{3!2!} = \frac{5 \cdot 4}{2} = 10$ ways

Question 5. *Rows and Columns* (former exam question)

We consider the following two player game. It is played with $m * n$ checkers pieces, that are positioned in an m times n rectangle (m rows and n columns). There are two players, V (Vertical) and H (Horizontal), that take turns making a move, V **always** makes the first move. The player who removes the last piece wins.

A move for V constitutes removing a column, a move for H constitutes removing a row. Removing a a row or column from the the rectangle generally splits it in two smaller rectangles (unless the row or column was removed from the edge). At the start of the game, there is only one square, but this number grows. A player on turn removes one row (H) or column (V) from exactly one of the rectangles.

Consider the following sequence of moves, for $m = 3$ and $n = 4$ (here, the pieces are represented by `X`, and different rectangles are separated by commas):

```
X X X X    V     X  X X    H     X              V     X           H    X
X X X X   --->   X, X X   --->   X, X X, X X   --->   X, X X, X  --->  X, X
X X X X          X  X X          X                    X                X
(*)                                                                    (**)
```

It's V's turn to move in `(**)`. If she removes the column with 3 pieces, she will lose. Suppose she removes the column with one piece. That leaves a winning position for H (Why?). Therefore, state `(**)` is losing for V.

a. Construct the state-space for the cases $m = 2$ and $n = 4$, given that V starts. Determine whether the game is winning for either V, H or neither. If either player has a winning strategy, provide this strategy.

Hint: Note down for every state whether it is winning for V or H. You do not need to draw states that are winnable in one move, and avoid elaborating on states equivalent to states already constructed – instead simply mention who will win.

b. We consider the special case $m = 1$, consisting of a single row with n pieces. We assume that V starts. Prove that for odd n, player V can always win by describing a winning strategy for V.
Hint: Recall the *Meta-algorithm,* first demonstrate that V will win on the situation of 1 times 3, and later extend this to general odd n.

c. Prove that for $m = 1$ and even values of n, V will always lose by describing a winning strategy for H.