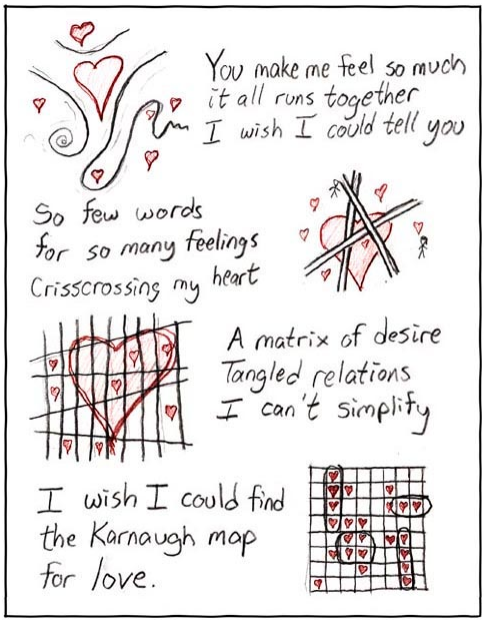
CS51 - Assignment 2
Are you an introvert?

Due: September 11th, at 11:59pm



https://xkcd.com/62/

For this assignment, we will:

- gain practice with constructing the disjunctive and conjunctive normal forms of a Boolean function given its truth table,
- learn about Karnaugh maps, a visual technique that simplifies Boolean expressions, and
- implement a simple Python program that uses Boolean logic to assess whether a user is an introvert or an extrovert.

1 Background

Boolean variables are limited in that they can only support one of two states, but we can think of interesting scenarios where they can be handy. For example, although personality traits are

not strictly binary, we could assume that a person is either an introvert or an extrovert (i.e., not an introvert). To further upset our friends in Psychological Sciences, we will claim that we can determine this based on four simple yes/no questions:

A: Do you prefer spending time alone rather than with a group?

B: Do you find large social gatherings draining?

C: Do you enjoy deep one-on-one conversations over small talk?

D: Do you often reflect before speaking or acting?

Each answer for the questions above corresponds to one of the four *A*–*D* Boolean variables. Answering "yes" means the value of that variable is 1 (TRUE), and "no" means 0 (FALSE).

2 Disjunctive and Conjunctive Normal Forms [4 points]

Based on the assumptions above, we have provided you with a truth table that defines whether a person is considered an introvert ($I = 1$) or not ($I = 0$) based on all possible answers to the four yes/no questions captured in columns *A*–*D*.

A	B	C	D	I
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Your first task is to calculate the minterms and maxterms for the four variables, $A - -D$ and then use them to calculate the disjunctive and conjunctive normal forms of the Boolean function defined by the truth table above. You should include your answer in the corresponding section of the provided `assignment2.tex` file.

3 Karnaugh Maps

Without spoiling the answers, you probably ended up with two rather complicated Boolean expressions. Mathematicians have a number of Boolean algebra rules that allow them to find equivalent Boolean expressions (think of De Morgan's Laws we saw in class), but these can often become equally complicated. Rather than learning all these rules and hoping that by applying them we will end up with a simpler expression, we will learn about **Karnaugh maps** (or **K-maps**).

A K-map is a visual tool used to simplify a Boolean function without needing to apply Boolean algebra rules. Instead, it relies on humans' ability to quickly discern visual patterns. We can use K-maps with Boolean functions of up to $4 - 5$ variables.

A K-map organizes the truth table of a Boolean function into a two-dimensional grid where the cells are ordered in **Gray code**. In Gray code, the ordering of a sequence of binary values is such that two successive values differ in only one bit. For example, if we work with two variables, rather than having the usual sequence 00, 01, 10, and 11, the Gray code sequence is 00, 01, 11, 10. If we work with three variables, the Gray code sequence is 000, 001, 011, 010, 110, 111, 101, 100. Similarly, if we work with four variables, the Gray code sequence is 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

3.1 2-variable K-maps

Let's assume we work with a Boolean function that has two input variables, A and B, and an output variable F captured in the truth table below:

A	B	F
0	0	1
0	1	0
1	0	1
1	1	1

As a first step, we will order the 4 possible input combinations for A and B in Gray code as 00, 01, 11, 10, and will place them in a 2×2 table. Each cell of the completed K-map contains a binary digit representing the output F for that combination of inputs. For our example, that would be:

		B = 0	B = 1
A	0	1	0
	1	1	1

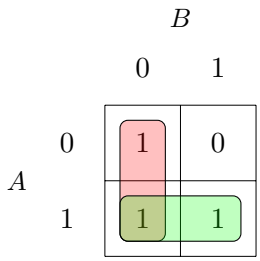
For example, the top right cell corresponds to $A = 0$ and $B = 1$, and the cell entry is equal to 0,

which matches the third row of the truth table. We apply the same logic for every cell.

The next step is creating groups of 1s in the K-map. We will mark different groups with different colors. The grouping should follow the following rules:

- Groups can only contain 1s, not 0s.
- Groups can be squares or rectangles that stretch horizontally or vertically, but *not* diagonally.
- The area of a group has to be a power of 2, i.e. the number of 1s it can include is 1, 2, 4, 8, etc.
- Groups should cover as large an area as possible, while we should also have as few groups as possible.
- Groups can overlap, and a cell with a 1 should be part of at least one group.
- Groups can wrap around. That means we can group a top cell with its matching bottom cell and a left cell with its matching right cell. (*This rule will make more sense when we work with more than 2 variables.*)

Applying these rules, we would get two groups visualized in red and green below:



Once we have the K-map and the groups of 1s, we examine which variables stay the same within each group. Within a group, if a variable remains unchanged and is equal to 1, then it contributes itself as is. If it remains unchanged but is equal to 0, then it contributes its negation/complement. The results of all the groups are connected with a disjunction (logical or).

For example, for the green group, the A variable remains unchanged and equal to 1, so we will include it in the final form as A. B, in contrast, changes from 0 to 1 within the green group, so we will ignore it. Similarly, for the red group, the B variable remains unchanged and equal to 0, so we will include it in the final form as $\neg B$. A, in contrast, changes within the red group, so we will ignore it. The final result will be the disjunction (logical or) of these two expressions, that is: $A \vee \neg B$.

And this is it: $A \vee \neg B$ is the Boolean expression that describes the entire truth table! Just substitute A and B with all possible combinations of 0s and 1s, and you will see that you can recreate the truth table. Pretty concise way of describing the entire truth table, right?

Let's try one more Boolean function with two input variables A and B and an output variable F:

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

We create, as before, the 2×2 table ordered in Gray code.

A \ B	0	1
0	1	0
1	0	1

Take a moment to think about how many groupings of 1s you can create based on the rules we just saw. Did you get the following?

A \ B	0	1
0	1	0
1	0	1

Within the red group, we see that both A and B remain unchanged and equal to 0. When we have two (or more if we work with more than two variables) variables that remain unchanged within a group, then we join them with a conjunction (logical and). That means that the red group will contribute $\neg A \wedge \neg B$.

Similarly, within the green group, we see that both A and B remain unchanged and equal to 1. Following the same process, the green group will contribute $A \wedge B$.

Putting the results of the two groups together, we get the simplified Boolean expression $(\neg A \wedge \neg B) \vee (A \wedge B)$ which describes the truth table we were given.

Make sure you understand how 2-variable K-maps work before you proceed. Try creating a different truth table and test whether you can create its K-map. Talk with one of the course staff if you have any questions or want to validate your understanding.

3.2 3-variable K-maps

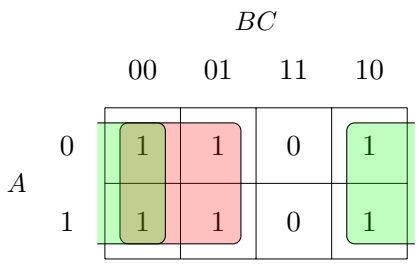
Now that we have established how K-maps work with two variables, let's assume we work with a Boolean function that has three input variables, A, B, and C, and an output variable F captured in the truth table shown below:

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

As a first step, we will order the 8 possible input combinations for A, B, and C in Gray code as 000, 001, 011, 010, 110, 111, 101, 100 in a 2×4 table (we could have gone for a 4×2 table, too). Each cell of the completed K-map contains a binary digit representing the output F for that combination of inputs. For our example, that would be:

		BC			
		00	01	11	10
A	0	1	1	0	1
	1	1	1	0	1

The next step is grouping the 1s, applying the rules we saw above:



Note that there are two groups we can create that have an area that is a power of 2. The red square is easy to detect. The green one is trickier and follows the rule that allows us to wrap a group around the edges.

Now that we have found the groups of 1s, we proceed as before. For example, for the green group,

only the C variable remains unchanged and equal to 0, so we will include it in the final form as $\neg C$. Similarly, for the red group, only the B variable remains unchanged and equal to 0, so we will include it in the final form as $\neg B$.

The final result will be the disjunction (logical or) of these two expressions, that is $F = (\neg B \vee \neg C)$. Compare that with the disjunctive normal form that would result from the above truth table (i.e. $F = (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$) and you can quickly see the value of K-maps.

Just to be sure we are all on the same page, let's work through one more example with 3-variable K-maps. You are given the following table, ordered in Gray code. Go over the grouping rules and take a moment to think about what groups of 1s you can create.

		BC			
		00	01	11	10
A	0	1	1	1	0
	1	0	1	0	1

What groupings of 1s did you create? Based on the rules we discussed, here are the four possible groupings:

		BC			
		00	01	11	10
A	0	1	1	1	0
	1	0	1	0	1

which would result in the Boolean expression $(\neg A \wedge \neg B) \vee (\neg A \wedge C) \vee (\neg B \wedge C) \vee (A \wedge B \wedge \neg C)$. The four operands of the disjunction correspond to the red, green, yellow, and blue groups.

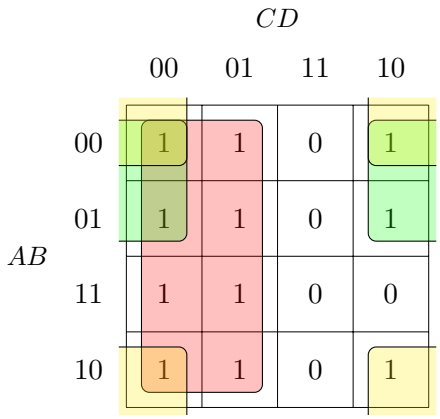
3.3 4-variable K-maps

K-maps can be (relatively) easy to understand up to four variables, which is our setting. Let's see an example of a K-map for four input variables, A, B, C, and D, organized in a 4×4 table.

AB \ CD	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	0
10	1	1	0	1

Let's pause for a moment and think of all the possible groupings of 1s we can create.

This is what you should have ended up with:



The red group will contribute the term $\neg C$. The green group will contribute the term $\neg A \wedge \neg D$. Finally, the (admittedly tricky) yellow group will contribute the term $\neg B \wedge \neg D$. Putting it all together, we get the simplified Boolean expression $(\neg C) \vee (\neg A \wedge \neg D) \vee (\neg B \wedge \neg D)$

3.4 Create a K-map [5 points]

Remember the truth table that captures all possible answers for the four yes/no questions that will determine whether someone is an introvert or not, and the complicated disjunctive and conjunctive normal forms you calculated?

A	B	C	D	I
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

1. **[3 points]** Now is your turn to apply what we just learned about K-maps and simplify the Boolean expression that describes this truth table. You can work on pen and paper and take a picture of the final colored K-map table, which you can include in your LaTeX submission, or you can use the karnaugh-map package directly. For both options, you can find instructions in the comments of the `assignment2.tex` file.

2. **[2 points]** Regardless of which one you choose, make sure you include the final Boolean expression you derived and explain which parts correspond to which colored group of 1s.

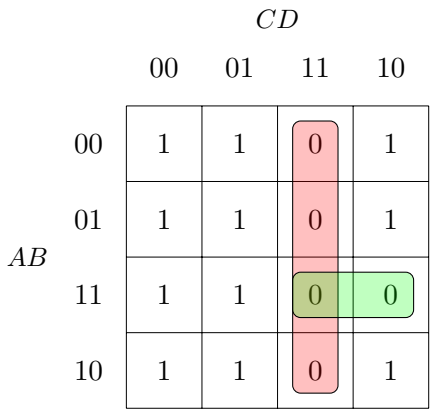
4 From your K-map to Python [1 point]

Now that we have found a concise expression to describe the truth table that corresponds to our personality quiz, we are ready to move on to writing a Python program. You have been given a `assignment2.py` file with a code skeleton that you are asked to fill in. Your task will be to prompt a user to answer the four yes/no questions and provide them with an assessment of whether they are an introvert or not based on the Boolean expression you derived using K-maps. You should test your program for each of the 16 possible inputs and ensure that you receive the correct output.

5 Extra credit: More K-maps [1 point]

When we created the K-map tables, the groups of 1s led to a disjunction of conjunctions, that is, a simplified disjunctive normal form. The mirrored concept of a simplified conjunctive normal form can also be derived using K-maps. Instead of grouping 1s, we can group 0s, following the exact

same rules when creating the groups. For example, the K-map for the 4-variable truth table would lead to the following groupings of 0s:



Within each group, we seek again unchanged variables with a slight twist. If a variable stays unchanged and equal to 0, then it contributes itself. If it's equal to 1, it contributes its negation. If two (or more) variables remain unchanged, we join them with a disjunction (logical or). All the expressions produced by groupings of 0s are joined with a conjunction (logical and). So pretty similar to what we saw before, but we have flipped 0s for 1s and conjunctions for disjunctions, and vice versa.

In our example, the red group would contribute $\neg C \vee \neg D$ because C and D remain unchanged and equal to 1. The green group would contribute $\neg A \vee \neg B \vee \neg C$ because A, B, and C remain unchanged and equal to 1. Putting it all together, we would get $(\neg C \vee \neg D) \wedge (\neg A \vee \neg B \vee \neg C)$. If we were to apply Boolean algebra rules, we would find that the two Boolean expressions we derived are equivalent.

For extra credit, you can use the K-map table you already have, group the 0s, and derive a simplified disjunctive normal form that describes the personality questionnaire truth table.

6 Style

Before you start coding, make sure you have reviewed the guide to Python syntax `https://cs.pomona.edu/classes/cs51/assignments/python_syntax_guide.pdf`. In particular, make sure you keep the following in mind as you're writing your program:

- All functions should have appropriate docstrings.
- Comment your code appropriately. Comment complicated parts of the code and include your name, date, etc. at the top of the file.
- Follow the variable naming conventions discussed in class and use good variable names.
- Use whitespace appropriately to make your program easier to read.

7 Feedback

Create a file named `feedback2.txt` that answers the questions:

- How long did you spend on this assignment?
- Any comments or feedback? What things did you find interesting, challenging, or boring?

8 When you're done

Submit your `assignment2.py`, `assignment2.tex`, and resulting `assignment2.pdf`, and `feedback2.txt` files online using Gradescope; be sure you're uploading them to the right assignment. Note that you can resubmit the same assignment as many times as you would like up until the deadline.

9 Grading

	points
Part 1: minterms	1
Part 1: disjunctive normal function	1
Part 1: maxterms	1
Part 1: conjunctive normal function	1
Part 2: K-map grouping of 1s	3
Part 2: K-map derived expression	2
Part 3: Prompting the user	0.5
Part 3: Correct translation of Boolean expression to Python code	0.5
Extra credit	1
Total	10 (+1)