This assignment is designed to be an introduction to the concepts used in our study of NP-completeness later in the semester. Some of the questions are trivial, some are subtle, and some are difficult. You should have enough background to do at least the first two parts.

The scenario: You are working as a computer programmer for the Alpha Beta Gamma Software company. It is your dream job.

1 Factoring

You have a meeting with your manager.

manager: See this number?
14474011154664524427946373126085988481573677491474835889066354349131199152128
Pretty large. Huh?

you: Wow. That *is* a large number.

manager: It is the twelfth perfect number.

you: What's a perfect number?

manager: Look it up.

you: If that's the twelfth perfect number, what are the first few perfect numbers?

manager: Look them up.

you: Why are we talking about about perfect numbers?

manager: Some questions you just have to answer for yourself.

you: It feels like I have to answer all of my questions myself.

manager: To answer one's own question is to be enlightened.

you: Is that a quote from Lao Tzu?

manager: I doubt it. Who's Lao Tzu?

you: Look it up.

manager: Touché. I have a feeling that you will soon be enlightened.

In any case, Alpha Beta Gamma is going into the burgeoning field of computational number theory. Starting next month, we will be receiving huge numbers that we would like to factor into primes. The twelfth perfect number has only 77 digits. Some of the numbers that we receive will have five hundred digits.

We hired you because you have a Computer Science degree from one of those fancy colleges. Nobody else here at Alpha Beta Gamma Software has your education or ability. You have two weeks to write an efficient program to factor numbers into primes. Do you think you can handle it?

you: Sure. I took a course in Discrete Math as part of my Computer Science major.

manager: Good. I knew I could count on you.

As you leave you are thinking that you would like a new manager, and wondering if you really can write this program.

1. *You may not discuss this question, directly or indirectly, with anyone involved with this homework assignment. You may use other resources.*
   (a) What is a perfect number?
   (b) What are the first three perfect numbers?
   (c) Parts (a) and (b) of this problem seem more like Math questions than Computer Science questions. Give three plausible reasons that Problem 1 is on the homework. Label your reasons (i), (ii), and (iii).

For the following problems, your factoring algorithms should input a number and print its prime factors. For example, if the input is 2020, then the output should be its prime factors in any order, such as 2, 2, 5, 101 or 5, 2, 101, 2.

You have routines available to add, subtract, multiply, and divide large integers. Dividing two integers will return the quotient and remainder. Note that the remainder is the mod. Here is the syntax for the integer operations:

```
sum ← add(a,b)
difference ← subtract(a,b)
product ← multiply(a,b)
(quotient,remainder) ← divide(a,b)
```

2. You recall an algorithm from elementary school for factoring a number N: Divide out all factors of 2, then of 3, then of 4, then of 5, then of 6, then of 7, etc. Finally, divide out all factors of N (of which there can be at most one).
   (a) Write pseudo-code for this algorithm (and print the prime factors).
   (b) Assume that it takes unit time to add, subtract, multiply, and divide integers. How fast is your algorithm, in order notation, as a function of N?
   (c) Normally we measure the time for such an algorithm as a function of the size of the input. In this case it is the number of digits in N, rather than the size of N. Assume that N is input as an n-digit binary number. How fast is your algorithm, in order notation, as a function of n?
   (d) If N is very large, you can no longer assume that arithmetic operations take constant time. Assume that it takes linear time, i.e. $\Theta(n)$ time, to add, subtract, multiply, and divide two n-digit binary numbers. How fast is your algorithm, in order notation, as a function of n (in the worst case)?

3. You are satisfied with your computer program, but your manager complains that it is much too slow. You recall that in high school you realized that you only have to go up to about $\sqrt{N}$ rather than all of the way up to N.
   (a) Assume that it takes unit time to add, subtract, multiply, and divide integers. How fast is this new version of your algorithm, in order notation, as a function of N?
   (b) Assume that it takes linear time, i.e. $\Theta(n)$ time, to add, subtract, multiply, and divide two n-digit binary numbers. How fast is this new version your algorithm, in order notation, as a function of n (in the worst case)?
   (c) You are surprised when your manager complains that it is still much too slow. Is your manager right? Justify.

4. You have an idea for a faster program. You scour the Internet and, as luck would have it, you find an algorithm

   `one_prime_factor(N)`

   that, given an n-digit binary number, N, will find one prime factor of the number in $\Theta(n^2)$ time.

   (a) Using this algorithm (as a black box), give an efficient algorithm to factor an n-digit binary number into primes (and print the prime factors).

   (b) Approximately how many times does your factoring algorithm use the `one_prime_factor(N)` routine in the worst case, as a function of n?

   (c) How fast is your algorithm (in the worst case), in order notation, as a function of n? As above, assume that it takes linear time, i.e. $\Theta(n)$ time, to add, subtract, multiply, and divide two n-digit binary numbers.

   (d) Is your new program fast enough? Justify.

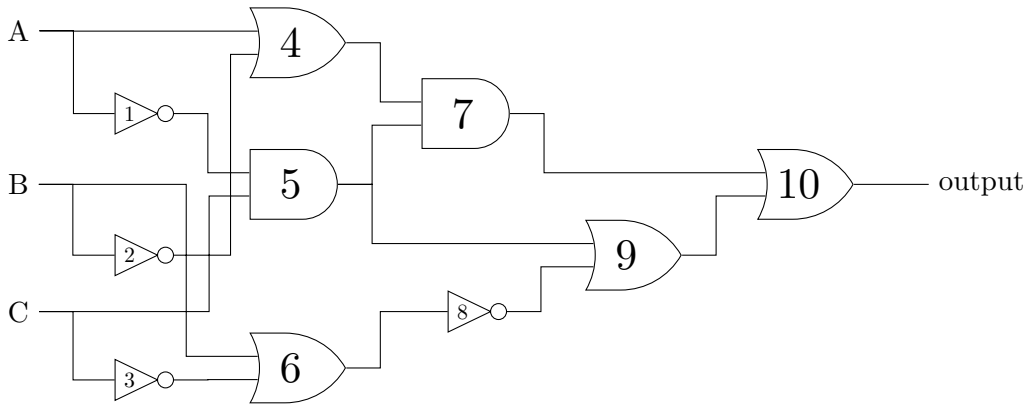
2 Boolean Formula Evaluation

5. (a) Consider the Boolean formula

$$((A \wedge \overline{B}) \vee (\overline{A} \wedge C)) \wedge (B \wedge \overline{C})$$

with assignment to the variables $A, B \equiv$ TRUE, $C \equiv$ FALSE. Evaluate the formula. You do not need to show your work.

(b) Consider the Boolean circuit, with assignment to the inputs $A, B \equiv$ TRUE, $C \equiv$ FALSE. Evaluate the Boolean circuit. Show your work by indicating the truth value produced by



each gate. Use the table in the back of this assignment.

The senario: You are working as a computer programmer for the Alpha Beta Gamma Software company. It is your dream job.

Your manager calls you into the office with the following comment:

> We hired you because you have a Computer Science degree from one of those fancy colleges. Nobody else here at Alpha Beta Gamma Software has your education or ability. We are moving into the business of Boolean formula evaluation. Starting next month, every morning we will be receiving a large number of large Boolean formulas. For each formula, the assignment of TRUE's and FALSE's to the variables will be given. We will evaluate each formula. We at Alpha Beta Gamma believe that we can rely on you to write a lightning fast program to evaluate these Boolean formulas.

You have no idea how to write such a program. You scour the internet but cannot find a satisfactory program to evaluate Boolean formulas. However, you do find a great program to evaluate Boolean circuits.

6. It turns out that it is easy to evaluate Boolean formulas directly in linear time. The spirit of this problem is to avoid doing so!

(a) Assume that you are allowed to use the program that evaluates Boolean circuits *only once*. Briefly explain how you would use the Boolean circuit evaluation program to evaluate a Boolean formula.

(b) Show what you would do on the following formula (from Part 4a):

$$((A \wedge \overline{B}) \vee (\overline{A} \wedge C)) \wedge (B \wedge \overline{C})$$

Make sure to draw the circuit that is input to the program that evaluates Boolean circuits.

7. Assume the scenario is reversed: You need to evaluate Boolean circuits, but you have a program that evaluates Boolean formulas. Analogously to the previous problem, it is easy to evaluate Boolean circuits directly in linear time. The spirit of this problem is to avoid doing so!

   (a) Assume that you are allowed to use the program that evaluates Boolean formulas *only once*. Briefly explain how you would use the Boolean formula evaluation program to evaluate a Boolean circuit.

   (b) Show what you would do on the circuit in Part 4b. Make sure to state the formula that is input to the program that evaluates Boolean formulas.

3 Boolean Formula Satisfiability

A Boolean formula is *satisfiable* if there is an assignment of TRUE's and FALSE's to the variables that makes the formula TRUE. A Boolean circuit with one output wire is *satisfiable* if there is an assignment of TRUE's and FALSE's (or 1's and 0's) to the inputs that makes the output wire TRUE (or 1).

8. This question works with Boolean formulas in Conjuntive Normal Form (CNF). If you are not sure what that is, look it up. A CNF formula is in *k-CNF* if every clause has exactly k literals (for some natural number k).
   (a) Give a 2-CNF formula that is satisfiable, where no variable occurs twice in the same clause. No justification needed.
   (b) Give a 2-CNF formula that is not satisfiable, where no variable occurs twice in the same clause. No justification needed.
   (c) Consider the following 3-CNF formula (with four variables and sixteen clauses):

$$(A \vee B \vee C)(A \vee B \vee \overline{C})(A \vee \overline{B} \vee C)(\overline{A} \vee B \vee \overline{C})(A \vee B \vee \overline{D})(\overline{A} \vee B \vee \overline{D})(A \vee \overline{B} \vee \overline{D})(\overline{A} \vee B \vee D)$$
$$(A \vee C \vee D)(A \vee C \vee \overline{D})(\overline{A} \vee C \vee D)(\overline{A} \vee \overline{C} \vee D)(B \vee C \vee D)(B \vee \overline{C} \vee D)(\overline{B} \vee C \vee D)(\overline{B} \vee \overline{C} \vee \overline{D})$$

   After much struggling, you discover that the formula is satisfiable with the assignment $A, D \equiv$ FALSE and $B, C \equiv$ TRUE. Confirm this by circling exactly one literal in each clause such that this assignment makes the clause TRUE. Use the table in the back of this assignment.

9. Imagine that there is a large Boolean formula (not necessarily in CNF) written on the whiteboard, where n is the number of connectives (AND's, OR's, and NOT's). Perhaps $n = 200$. You claim that the formula is satisfiable; I claim that it is not.
   (a) What do you have to do to convince me that you are right? How much time (in order notation) do you need (as a function of n)? Note: Only count the time needed to actually show me that you are right, not any time needed to figure out how to do so.
   (b) What do I have to do to convince you that I am right? How much time (in order notation) do I need (as a function of n)? Note: Only count the time needed to actually show you that I am right, not any time needed to figure out how to do so.

   Your manager calls you into the office with the following comment:

   > We are now moving into the business of Boolean formula satisfiability. Starting next month, every morning we will be receiving a large number of large Boolean formulas. For each formula, we will need to determine if it is satisfiable. Note that we do not have to actually find the satisfying assignment; we just need a YES/NO answer for each formula.
   >
   > Once again we need your unique skills. You have two weeks to write a lightning fast program to solve satisfiability for these formulas.

10. Of course, you have no idea how to write such a program. You scour the internet but cannot find a satisfactory program to solve satisfiability for Boolean formulas. However, you do find a great program to solve satisfiability for Boolean circuits.

(a) Explain very briefly in English how you would use this Boolean circuit program to solve satisfiability for Boolean formulas.

(b) Assume that the Boolean circuit satisfiability program works in linear time $\Theta(m)$, where m is the number of gates and/or wires in the Boolean circuit. How fast can you determine if a formula with n connectives is satisfiable? Justify.

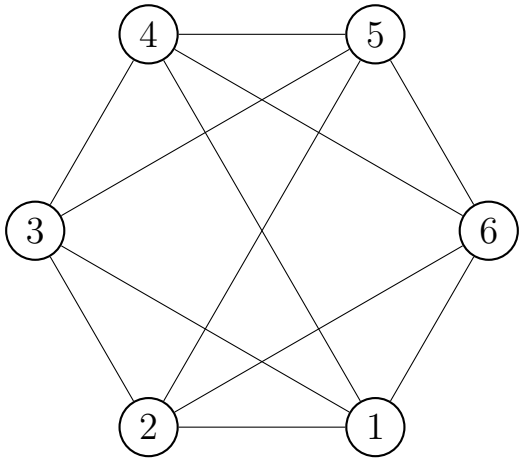
(c) Assume that the Boolean circuit satisfiability program works in time $\Theta(m^r)$, where m is the number of gates and/or wires in the Boolean circuit. How fast can you determine if a formula with n connectives is satisfiable? Justify.

4 Graph Coloring

An undirected graph $G = (V, E)$ is *c-colorable* if each vertex can be assigned a color such that at most c colors are used and no two vertices that share an edge have the same color.

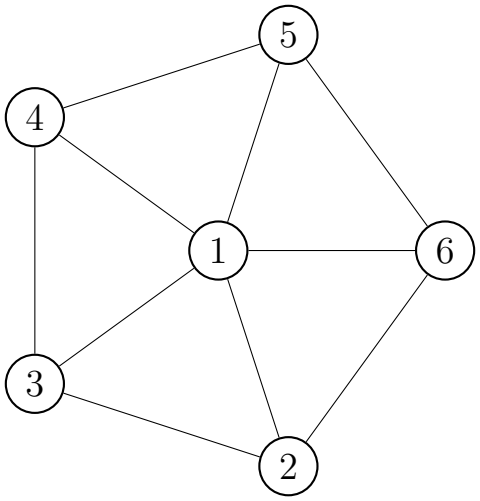
11. You must solve these problems as described. There may be much more clever methods, but we do not care.

    (a) Show that the following graph is 3-colorable, by assigning a color to each vertex and showing that no edge has the same color on its two endpoints.



    The three vertices in triangle (1,2,3) must be different colors. By symmetry, we can set those three colors to Red (R), Blue (B), and Green (G), respectively. This way we can guarantee that there is a unique solution to this problem. Use the two tables provided in the back of this assignment.

    (b) Show that the following graph is not 3-colorable, by showing that each possible assignment of colors to the vertices is not a 3-coloring. This is done by finding an edge in which both



    endpoints have the same color.

There are too many possible colorings to make this practical to do by hand. Once again, the three vertices in the triangle (1,2,3) must be different colors, so by symmetry, we can set those three colors once and for all to, say, Red, Blue, and and Green, respectively. Then we only need to color the remaining three vertices. Use the table in the back of this assignment.

12. Imagine that there is a large undirected graph with n vertices drawn on the whiteboard. Perhaps $n = 200$. You claim that the graph is c-colorable; I claim that it is not.

    (a) What do you have to do to convince me that you are right? How much time (in order notation) do you need (as a function of n and c)? Note: Only count the time needed to actually show me that you are right, not any time needed to figure out how to do so.

    (b) What do I have to do to convince you that I am right? How much time (in order notation) do I need (as a function of n and c)? Note: Only count the time needed to actually show you that I am right, not any time needed to figure out how to do so.

5 Finding Satisfying Assignments

Your manager calls you into the office with the following comment:

> It turns out that not only do we need to determine whether a Boolean formula is satisfiable, but, if so, we need to find a satisfying assignment. Furthermore, the inputs are more complicated than we expected: Boolean formulas can not only have variables, but they can also have TRUE's and FALSE's. So an input might, for example, be
>
> $$((A \wedge (\overline{B} \wedge \text{TRUE})) \vee (\overline{A} \wedge C)) \wedge ((B \wedge \overline{C}) \vee \text{FALSE})$$
>
> Once again we need your unique skills. You have two weeks to write a fast program to determine if a formula is satisfiable, and, if so, to find the satisfying assignment.

You find a program

`satisfiable(`H`)`

that allows TRUE's and FALSE's in the input, H, and returns YES or NO depending on whether the Boolean formula is satisfiable. It runs in time $\Theta(n^r)$, for some constant $r \geq 1$, where n is the number of variables.

You also find a program

`substitute(`H, X, V`)`

that substitutes the value V for variable X in formula H, where V is either TRUE or FALSE, and returns the new formula. It runs in linear time.

13. (a) Show how to use (the Boolean formula satisfiability program) `satisfiable` (and `substitute`) to efficiently find a satisfying assignment. Write the pseudo-code.
    (b) How fast is your algorithm? Justify.

Your manager calls you into the office, compliments you on a great job, and continues:

> It turns out that not only do we need to find a satisfying assignment, but the assignment must minimize the number of variables set to TRUE.

You find a progam

`satisfiable_num(`H`,`k`)`

that allows TRUE's and FALSE's in the input, H, and returns YES or NO depending on whether the Boolean formula is satisfiable with at most k variables set to TRUE. It runs in time $\Theta(n^r)$, for some constant $r \geq 1$, where n is the number of variables.

14. (a) Show how to use (the Boolean formula satisfiability program) `satisfiable_num` (and `substitute`) to find a satisfying assignment that minimizes the number of variables set to TRUE. Write the pseudo-code.

        HINT: eurtottesebtsumselbairavynamwohenimretedtsrif.

        HINT for hint: sdrawkcabtidaer.

    (b) How fast is your algorithm? Justify.

6 Sending Secret Messages

The CEO of Alpha Beta Gamma wants to send out secret messages to all of the employees using their newly created BOOM system. Each employee will receive the message from some other employee and forward it to at most two other employees. (The CEO will send the original secret message to one or two other employees.)

Alpha Beta Gamma will set up each employee with special hardware so that it costs $C[i, j]$ for employee i to send a message to employee j. It has some flexibility on how to set up the hardware. Alpha Beta Gamma would like to minimize the total cost of broadcasting a message.

(1) Given n employees (numbered $1, \ldots, n$), costs $C[i, j]$ for employee i to send a message to employee j, where $C[i, j]$ is an integer between 1 and 10, and a maximum cost M, determine if there is a way to broadcast a secret message for total cost at most M.

(2) Given n employees (numbered $1, \ldots, n$), costs $C[i, j]$ for employee i to send a message to employee j, where $C[i, j]$ is an integer between 1 and 10, find a way to broadcast a secret message for cheapest total cost.

These two problems seem to be hard to solve efficiently. Not surprisingly, your manager asks you to write programs to solve the two problems. As usual, you have no idea how to write such programs. You find efficient programs on the Internet that solve both problems. Unfortunately your budget will only allow you to buy one such program.

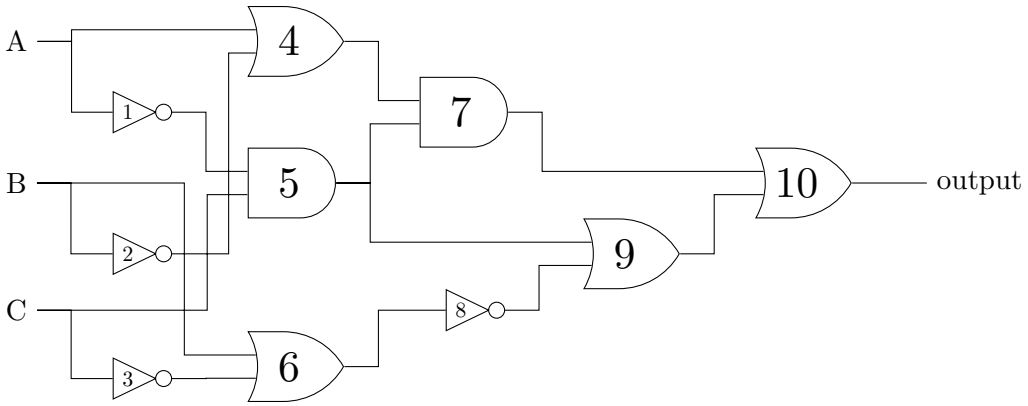
15. Assume that you have a program that solves the *second* problem in time $\Theta(n^7)$. Can you use it to solve the *first* problem in polynomial time? If so, how, and how fast is your algorithm?

16. Assume that you have a program that solves the *first* problem in time $\Theta(n^5)$. Can you use it to solve the *second* problem in polynomial time? If so, how, and how fast is your algorithm?

    NOTE (and HINT): You can assume that the program can handle arbitrary integer costs for $C[i, j]$, not just integers between 1 and 10.

17. **Challenge problem, will not be graded.** Solve the previous problem assuming that the program can only handle integer costs for $C[i, j]$ between 1 and 10.

Assignment to the inputs $A, B \equiv$ TRUE, $C \equiv$ FALSE.



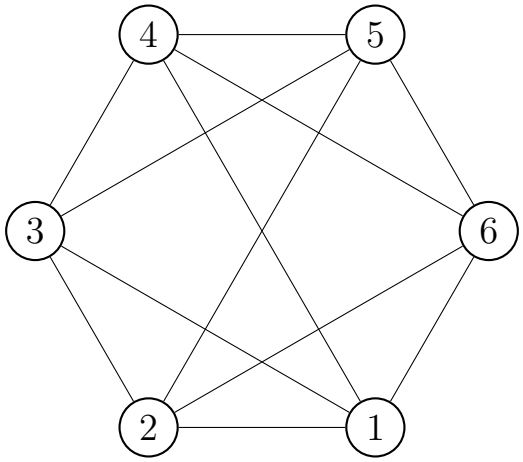
Gate	Output value
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Satisfying assignment: $A, D \equiv$ FALSE and $B, C \equiv$ TRUE.

Clause
$(A \vee B \vee C)$
$(A \vee B \vee \overline{C})$
$(A \vee \overline{B} \vee C)$
$(\overline{A} \vee B \vee \overline{C})$
$(A \vee B \vee \overline{D})$
$(\overline{A} \vee B \vee \overline{D})$
$(A \vee \overline{B} \vee \overline{D})$
$(\overline{A} \vee B \vee D)$
$(A \vee C \vee D)$
$(A \vee C \vee \overline{D})$
$(\overline{A} \vee C \vee D)$
$(\overline{A} \vee \overline{C} \vee D)$
$(B \vee C \vee D)$
$(B \vee \overline{C} \vee D)$
$(\overline{B} \vee C \vee D)$
$(\overline{B} \vee \overline{C} \vee \overline{D})$

Vertex	1	2	3	4	5	6
Color (R,B,G)	R	B	G			

Edge	Color left endpoint	Color right endpoint
(1,2)		
(1,3)		
(1,4)		
(1,6)		
(2,3)		
(2,5)		
(2,6)		
(3,4)		
(3,5)		
(4,5)		
(4,6)		
(5,6)		



123456	Bad edge	Color on endpoints
RBGRRR		
RBGRRB		
RBGRRG		
RBGRBR		
RBGRBB		
RBGRBG		
RBGRGR		
RBGRGB		
RBGRGG		
RBGBRR		
RBGBRB		
RBGBRG		
RBGBBR		
RBGBBB		
RBGBBG		
RBGBGR		
RBGBGB		
RBGBGG		
RBGGRR		
RBGGRB		
RBGGRG		
RBGGBR		
RBGGBB		
RBGGBG		
RBGGGR		
RBGGGB		
RBGGGG		