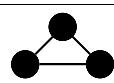
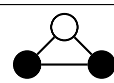
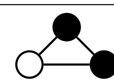
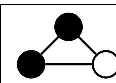
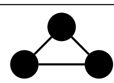
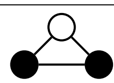
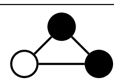
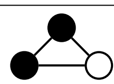
Homework 7

Obliviate

May 2021

1 Solution:

By **Exercise 2**, Alice always chooses a vertex cover. Then we can write the payoff matrix of this game as follows.

Bob / Alice				
	1	1/2	1/2	1/2
	1/2	1	1/4	1/4
	1/2	1/4	1	1/4
	1/2	1/4	1/4	1

We use a mix strategy to obtain the optimal solution. By symmetry, the probability, denoted by p, that each vertex cover of size 2 is chose by Alice is equal. The same thing, denoted by q, holds for Bob.

Then we can compute the value of this game:

$$\begin{pmatrix} 1-3p & p & p & p \end{pmatrix} \begin{pmatrix} 1 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1 & 1/4 & 1/4 \\ 1/2 & 1/4 & 1 & 1/4 \\ 1/2 & 1/4 & 1/4 & 1 \end{pmatrix} \begin{pmatrix} 1-3q \\ q \\ q \\ q \end{pmatrix} = \frac{1}{2} + \left(\frac{1}{\sqrt{2}} - \frac{3}{\sqrt{2}}p\right)\left(\frac{1}{\sqrt{2}} - \frac{3}{\sqrt{2}}q\right)$$

If p is greater or less than $\frac{1}{3}$, q can choose to be less or greater than $\frac{1}{3}$ and the total value then becomes less than $\frac{1}{2}$.

If q is greater or less than $\frac{1}{3}$, q can choose to be greater or less than $\frac{1}{3}$ and the total value then becomes greater than $\frac{1}{2}$.

Hence both Alice and Bob will choose $\frac{1}{3}$.

Set $p = q = \frac{1}{3}$ and then we can get the value $\frac{1}{2}$.

2 Solution:

For any set A that isn't a vertex cover, we can always find a better response for Alice. Since A isn't a vertex cover, we can always find an edge that points at both ends of it are not covered, and call the two points u and v. Then consider another strategy A': A' covers all the points in A and u, v.

Bob always choose a vertex cover, so any strategy for Bob must cover at least one of points u and v. For strategies that only cover u or v, the payoff of A' is the same as A. For strategies that cover both u and v, the payoff of A' is larger than A. Therefore, A is always not the best response. Let x be an optimal mixed strategy for Alice, then if A isn't a vertex cover, $x_A = 0$. That is to say, $x_A > 0$ then A is a vertex cover.

3 Solution:

α is the solution of $f(\alpha) = \alpha^k - (\alpha^{k-1} + \cdots + \alpha + 1) = 0$. When $k = 1$, $f(\alpha) = \alpha - 1$ has only one positive solution $\alpha = 1$. Then we only consider the case when $k > 1$.

Because $f(1) = 1 - k < 0$ and $f(0) = -1 < 0$, 1 and 0 aren't solutions. So we have $f(\alpha) = \alpha^k - \frac{1-\alpha^k}{1-\alpha} = 0, \alpha \neq 1$. Let $g(\alpha) = (\alpha - 1)f(\alpha) = \alpha^{k+1} - 2 \times \alpha^k + 1 = 0$. And $g'(\alpha) = (k+1)\alpha^k - 2k\alpha^{k-1}$.

$$g'(\alpha) \begin{cases} < 0, & 0 < \alpha < \frac{2k}{k+1} \\ = 0, & \alpha = \frac{2k}{k+1} \\ > 0, & \alpha > \frac{2k}{k+1} \end{cases}$$

Obviously, $1 < \frac{2k}{k+1} < 2$ and $g(1) = 0, g(2) = 1 > 0$. So there is at least one positive solution $x \in (1, 2)$ of $g(\alpha) = 0$. And this is also the only solution of $f(\alpha) = 0$.

4 Solution:

If $k = 1$, $\forall n \geq 1, L_1(n) = \alpha_k^n = 1$. Next, we only consider the case when $k > 1$.

When $2 \leq n \leq k$, we have $L_k(n) = 2^{n-2}$. And $\frac{1}{4} = \frac{2^{n-2}}{2^n} < \frac{L_k(n)}{\alpha_k^n} < \frac{L_k(k)}{1} = 2^{k-2}$. For $m > k$, if we have $\forall n < m$, $\frac{L_k(n)}{\alpha_k^n} \in (\frac{1}{4}, 2^{k-2})$, then:

$$\frac{L_k(m)}{\alpha_k^m} = \frac{\sum_{i=1}^{k} L_k(m-i)}{\sum_{i=1}^{k} \alpha_k^{m-i}} \in \left(\frac{1}{4}, 2^{k-2}\right)$$

That is to say, we can prove $\forall n \geq 2$, $\frac{L_k(n)}{\alpha_k^n} \in (\frac{1}{4}, 2^{k-2})$ by induction. So $L_k(n) \in \Theta(\alpha_k^n)$.

5 Solution:

F' is a formula whose shortest clause has size k, and this means all the clauses in F' have size k. So any variable that is assigned during F to F' doesn't occur in the clauses of F'. That is to say, the assignment of those variables is feasible and doesn't make a difference to the recursive call lazyBB(F'). This is to say, $F' \subseteq F$. So if F' can't be satisfiable, F can't be satisfiable either.

if F' is satisfiable, this also means there's no contradiction for the assignment of the variables from F to F', so F is satisfiable, too.

In conclusion, F is satisfiable if and only if F' is satisfiable.

6 Solution:

Algorithm 1 Determine whether a given k-CNF formula is satisfiable

```
procedure MODIFIED-LAZYBB(F)
    if F is empty, i.e., already satisfied then
        return F
    else if F contains □ then
        return □
    else
        choose a shortest clause C = (u_1 ∨ ··· ∨ u_ℓ)
        if ℓ = k then
            return F
        else
            for i = 1, ..., ℓ do
                F_i := F|_{u_1 = ··· = u_{i-1} = 0, u_i = 1}
                // set u_i to true and all previous literals to false
                if MODIFIED-LAZYBB(F_i) ≠ □ then
                    return F
                end if
            end for
            return □
        end if
    end if
end procedure
procedure k-SAT(F)
    if F contains □ then
        return unsatisfiable
    else
        choose an arbitrary variable x
        let F_1 = MODIFIED-LAZYBB(F|_{x=1})
        if F_1 is empty, i.e., already satisfied then
            return satisfiable
        else if F_1 contains □ then
            return k-SAT(F|_{x=0})
        else
            // F_1 contains only k-clauses
            return k-SAT(F_1)
        end if
    end if
end procedure
```

MODIFIED-LAZYBB tries to reduce F to an exactly-k-CNF F' by assigning values to variables.

k-SAT choose an arbitrary variable and set it true to obtain F'. It then applies MODIFIED-LAZYBB(F'). If succeeded, we will obtain F'' and by exercise 5, F' is satisfiable if and only if F'' is satisfiable. Otherwise the variable we choose should be false.

After each recursion of k-SAT, the number of variables in F will decreased by at least 1. And the complexity of MODIFIED-LAZYBB is $\Theta\left(\alpha_{k-1}^n \cdot \text{poly}(|F|)\right)$ since during the procedure, the length of the shortest clause is less than k.

Hence the complexity of this algorithm is $\Theta\left(\alpha_{k-1}^n \cdot \text{poly}(|F|)\right)$.

7 Solution:

For each clause C, there are exactly 2^{n-k} configurations of variables that dissatisfy C.

So there are at most $m2^{n-k}$ configurations that dissatisfy F, while the number of total configurations is 2^n.

If $m < 2^k$, then there must exist a configuration that satisfies F.

8 Solution:

According to Hall's Theorem, if we can prove that $\forall S \subseteq \{C_1, C_2, ..., C_m\}, |S| \leq |N(S)|$, then the incidence graph has a matching of size m. ($N(S)$ means the set of neighbours of S in $\{x_1, x_2, ..., x_n\}$)

We have $\Delta(F) \leq k$, and this means for any $x \in \{x_1, x_2, ..., x_n\}$, $deg(x) \leq k$. We also have F is an exactly-k-CNF formula, and this means for any $C \in \{C_1, C_2, ..., C_m\}$, $deg(C) = k$. Since $\sum_{C\in S} deg(C) = \sum_{x\in N(S)} deg(x)$, $|S| \leq |N(S)|$. Using Hall's Theorem, we can prove that the incidence graph has a matching of size m.

9 Solution:

The incidence graph is a binary graph. And it has a matching of size m. So for every clause $C \in F$, it is matched to a variable x_C. We define x_C is true if C contains x_C, and x_C is false otherwise. For the variables which aren't matched, we can choose their value freely. Every clause has at least one literals making the clause true. So F is satisfiable.