SOLUTION TO EXERCISES OF DISCRETE MATHEMATICS

CAI SIXIANG

1. Set relation and functions

Problem 1. Prove$|A \cup B| = |A| + |B| - |A \cap B|$.

Proof: To see this, we can split $A \cup B$ into $A \cap B^c$,$A \cap B$ and $B \cap A^c$. So we have $|A \cup B| = |A \cap B^c| + |A \cap B| + |B \cap A^c|(1)$. We also note that we can split A into $A \cap B$ and $A \cap B^c$. So $|A| = |A \cap B| + |A \cap B^c|(2)$. Likewise we have $|B| = |A \cap B| + |B \cap A^c|(3)$. Combine (1),(2),(3), we have $|A \cup B| = |A| + |B| - |A \cap B|$.

Problem 2. Proof:

$|A \cup B \cup C| = |A \cup B| + |C| - |(A \cup B) \cap C| = |A| + |B| - |A \cap B| + |C| - |(A \cap C) \cup (B \cap C)| = |A| + |B| - |A \cap B| + |C| - (|A \cap C| + |B \cap C| - |A \cap B \cap C|) = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$.

Problem 3. Prove $|A_1 \cup \cdots \cup A_n| = \sum_{\emptyset \neq I \subseteq [n]} (-1)^{|I|+1} |\cap_{i \in I} A_i|$.

Proof: We prove by induction. Case for n=1 is obvious.
Suppose $|A_1 \cup \cdots \cup A_n| = \sum_{\emptyset \neq I \subseteq [n]} (-1)^{|I|+1} |\cap_{i \in I} A_i|$. Then

$$
\begin{aligned}
&|A_1 \cup \cdots \cup A_n \cup A_{n+1}| \\
=&|A_1 \cup \cdots \cup A_n| + |A_{n+1}| - |(A_1 \cup \cdots \cup A_n) \cap A_{n+1}| \\
=& \sum_{\emptyset \neq I \subseteq [n]} (-1)^{|I|+1} |\cap_{i \in I} A_i| + |A_{n+1}| - |\bigcup_{k=1}^{n} (A_k \cap A_{n+1})| \\
=& \sum_{\emptyset \neq I \subseteq [n]} (-1)^{|I|+1} |\cap_{i \in I} A_i| + |A_{n+1}| - \sum_{\emptyset \neq I \subseteq [n]} (-1)^{|I|+1} |\cap_{i \in I} A_i \cap A_{n+1}| \\
=& \sum_{\emptyset \neq I \subseteq [n+1]} (-1)^{|I|+1} |\cap_{i \in I} A_i|
\end{aligned}
$$

Then by induction we have proved the formula.

2. Partial order,chians,antichains

Problem 4. For $n \geq 1$ we consider the partial order $(\mathbb{N}_0^n, \leq)$. To be more specific,$\mathbb{N}_n^0$ is the set of all n-tuples of natural numbers, for example $(3, 0, 4) \in \mathbb{N}_0^3$. For $x, y \in \mathbb{N}_0^n$

Date: 2018-05-06.

This is the solution to exercises in Discret Mathematics given by Prof. Dominik Scheder on Coursera. Thanks Prof Scheder for the wonderful experience.

we write $x \leq y$ if $x_i \leq y_i$ for all $1 \leq i \leq n$. For example $(3,0,4) \leq (4,1,4)$ but $(3,0,4) \not\leq (4,0,3)$.

Show that the partial order $(\mathbb{N}_0^n, \leq)$ has arbitrarily large antichains. That is, for every $k \in \mathbb{N}_0$, there is an antichain of size k.

Proof. Observe that if $x \preceq y$, then $\sum_{i=1}^n x_i \leq \sum_{i=1}^n y_i$. If $\sum_{i=1}^n x_i = \sum_{i=1}^n y_i$, then either $x = y$ or x and y are not comparable. Now let $AC_k^n = \{x | \sum_{i=1}^n x_i = k, x_i \geq 0\} \subset \mathbb{N}_0^n$. We claim that AC_k^n is an antichain with a size at least k. It suffice to count the distinct elements in AC_k. This is a combinatorial excersie. We can easily see that $|AC_k^n| \geq |AC_k^2| = \#\{(0,k),(1,k-1),\ldots,(k,0)\} = k+1$. So any subset of size k of AC_k^n is an antichian of size k. ■

Problem 5. Show that the partial order $(\mathbb{N}_0^n, \leq)$ has no infinite antichain.

Proof. We will prove by contradiciton.

First we consider the case $n = 2$. Suppose $(\mathbb{N}_0^2, \leq)$ has infinite antichain AC. We could always find an index $j \in \{1,2\}$ and an infinite sequence of elements of AC, $\{x^{(1)}, x^{(2)}, \cdots\}$ such that $\{x_j^{(k)}\}$ forms a descending sequence. We describe the algorithm.

Step 1: pick a $x^{(1)} \in AC$. Without lose of generality, we suppose that $AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(1)}\}$ is an infinite set.

Step 2: pick a $x^{(2)} \in AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(1)}\}$. We have $x_1^{(1)} > x_1^{(2)}$. Then $AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(2)}\}$ is still an infinite set. To see why this is true, if we suppose that$AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(2)}\}$ is finite, as $AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(1)}\}$ is infinite, then $AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | x_1^{(2)} < u_1 < x_1^{(1)}\}$ must be inifite. But since all elements of AC are not $\geq x^2$

$$AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | x_1^{(2)} < u_1 < x_1^{(1)}\} \subseteq U = \{u = (u_1, u_2) \in \mathbb{N}_0^2 | x_2^{(1)} < u_1 < x_1^{(1)}, x_2^{(1)} \leq u_2 \leq x_2^{(2)}\}.$$

But the set U is finite. Thus a contradiction.

Step 3: We could then procede the same argument in Step 3 to find a $x^{(3)} \in AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(2)}\}$. And the set $AC \cap \{u = (u_1, u_2) \in \mathbb{N}_0^2 | u_1 < x_1^{(3)}\}$ is infinite.

By induction, we have found our sequence. But $\{x_j^{(k)}\}$ being an infinite descending sequence contradicts the fact that 0 is a minimal elemnt of $\mathbb{N}$.

Thus $(\mathbb{N}_0^2, \leq)$ has no infinite antichain.

Now suppose the $(\mathbb{N}_0^k, \leq)$ has no infinite antichain. We claim $(\mathbb{N}_0^{k+1}, \leq)$ has no infinite antichain.

Suppose otherwise $(\mathbb{N}_0^{k+1}, \leq)$ has an infinite antichain AC. But the restriction of AC to $(\mathbb{N}_0^k, \leq)$ is not an antichain and since the restriction of AC to $(\mathbb{N}_0^k, \leq)$ is also infinite it actually contains an infinite chain in $(\mathbb{N}_0^k, \leq)$. Pick an infinite sequence$\{x^{(1)}, x^{(2)}, \cdots\} \subseteq AC$ which its restriction to $(\mathbb{N}_0^k, \leq)$ forms an infinite ascending chain , then one of two case must be true: 1) $\exists l, m \in \mathbb{N}$ such that $x_{k+1}^{(l)} \leq x_{k+1}^{(m)}$. In this case $x^{(l)} \leq x^{(m)}$, contradicts the fact that AC is an antichain. 2) $x_{k+1}^{(1)} > x_{k+1}^{(2)} > x_{k+1}^{(3)} > \cdots$. But this is impossible since there is no infinite strict descending sequence in $\mathbb{N}_0$. So either way, we have a contradiction. So $(\mathbb{N}_0^{k+1}, \leq)$ has no infinite antichain.

By induction the partial order $(\mathbb{N}_0^n, \leq)$ has no infinite antichain for all $n \geq 2$. ■

Problem 6. Show the following statement which is even stronger than the one before: Every infinite set $S \subseteq \mathbb{N}_0^n$ contains an infinite chain $C \subseteq S$.(try to prive for $n = 2$. then use induction on n.

Proof. Case $n = 2$.

Let $S \subseteq \mathbb{N}_0^2$. We will construct an infinite chain. Without lose of generatliy, we suppose $S \cap \{x = (x_1, x_2) | x_1 \leq x_2\}$ is infinite. Let M_1 be the set of minimals of S, then we pick $c^{(1)}$ such that $c^{(1)} \in M_1$ and $c_1^{(1)} \leq x$ for all $x \in M_1$. Then $S_1 = \{x \in S | x \succeq x_1\}$ is not empty since S is infinite. Then note by M_2 the set of the minimals of $S \backslash M_1$. We pick $c^{(2)}$ such that $c^{(2)} \in M_2$ and $c_1^{(2)} \leq x$ for all $x \in M_2$. Then we have $c^{(1)} \preceq c^{(2)}$. And we can preced to get an ascending chain which is infinite.

Suppose that every $S \subseteq N_0^k$ contains an infinite chain $C \subseteq S$. Now we prove that $S \subseteq N_0^{k+1}$ contains an infinite chain.

Let S^{k+1} be an infinite subset of N_0^{k+1}. By the induction hypothese, the restricition of S^{k+1} to N_0^k contains an infinite chian C^k. Pick an infinite sequence $D = \{d^{(1)}, d^{(2)}, \cdots\} \subseteq S^{k+1}$ with C^k as its restriction to N_0^k. Then if we could find an infinite non-decreasing subsequence of $\{d_{k+1}^{(i_1)}, d_{k+1}^{(i_2)}, \cdots\}$ such that $1 \leq i_1 \leq i_2 \leq \cdots$, and $d_{k+1}^{(i_1)} \leq d_{k+1}^{(i_2)} \leq \cdots$, then we are done. Indeed, as there is no infinite strict descending sequence in $\mathbb{N}_0$, such non-decreasing subsequence always exists. Thus we have found an infinite chain $C^{k+1} = \{d_{k+1}^{(i_1)}, d_{k+1}^{(i_2)}, \cdots\}$.

By induction on n, we conclude the proof. ∎

Problem 7. Consider the partial order $(\{0,1\}^n, \leq)$, where $\{0,1\}^n$ is the set of all length-n-strings over $0, 1$. For $x, y \subseteq \{0,1\}^n$ we define $x \leq y$ if $x_i \leq y_i$ for all $1 \leq i \leq n$. For example, $(1,0,0) \leq (1,0,1)$ but$(1,0,0) \not\leq (0,1,0)$.

What is the largest chain of$(\{0,1\}^n, \leq)$. Justify your answer!

Proof. The minimal of $P = (\{0,1\}^n, \leq)$ is a string of all 0 , $s_1 = (0, 0, \cdots, 0)$. The immediate succesor of s_1 is $S^2 = \{s \in \{0,1\}^n |$only one charater being 1, others all being 0$\}$. Then $|S^2| = C_n^1$. Pick $s_2 \in S^2$,its immediate succesor is$S^3(s_2) = \{s \in \{0,1\} |$ only 2 characters being 1, others all being 0. One of the 1s is at the same poistion of the 1 of $s_2\}$.

And then we could progress unitil we reach $s_{n+1} = (1, 1, \cdots, 1)$, which is the maximal of $(\{0,1\}^n, \leq)$. Note the length of the chain is $n + 1$.

The justification is by Mirsky's theorm that n+1 is the smallest number of t such that P can be partitioned into t antichains. First lets see what these antichains are: $\forall i \in \{1, \cdots, n+1\}$ $A_i = \{x \in \{0,1\}^n | \sum_{k=1}^n 1_{x_i=1} = i - 1\}$. Clearly $\{A_i\}$ is an antichain partition of $(\{0,1\}^n, \leq)$.

Suppose there exist a n antichain partition $\uplus_{i=1}^n B_i$ of P. But we could find a chain C with lenghth $n+1$. By the peigon hole principle, $\exists i, j, l$ such that $c_i, c_j \in C \cap B_l$. This contradict the fact B_l being an antichain. So by contradiction n+1 is the smallest number of t such that P can be partitioned into t antichains. ∎

Problem 8. What is the largest antichain of $(\{0,1\}^n, \leq)$? Justify your answer!

Proof. The largest anitichain of $P = (\{0,1\}^n, \leq)$ is

$$A_{\lfloor n/2 \rfloor + 1} = \{x \in \{0,1\}^n | \sum_{k=1}^n 1_{x_i=1} = \lfloor n/2 \rfloor\}.$$

Note if n is even, $\lfloor n/2 \rfloor = n/2$, if n is odd, $\lfloor n/2 \rfloor = (n-1)/2$. As same notation in the previous question, $\uplus_{i=1}^{n+1} A_i$ is a smallest partition of the poset. By binomial theorem, $|A_i| = C_n^{i-1}$. So we know that C_n^{i-1} attains its maximum for $i = \lfloor n/2 \rfloor + 1$. Now we need to prove $A_{\lfloor n/2 \rfloor + 1}$ is indeed a largest antichain. Note that $|A_{\lfloor n/2 \rfloor + 1}| = C_n^{\lfloor n/2 \rfloor}$

By Dilworth's theorem, if P could be partitioned into at least $C_n^{\lfloor n/2 \rfloor}$ chains, then we are done. Since if P can be partitioned into less than $C_n^{\lfloor n/2 \rfloor}$ chains, say $\uplus_{i=1}^{k} B_i$ with $k < C_n^{\lfloor n/2 \rfloor}$. Then by pigeon hole principle $\exists i, j, l$ such that $b_i, b_j \in A_{\lfloor n/2 \rfloor + 1} \cap B_l$. But this could not be true since two element of an antichain b_i and b_j are comparable(in the same chain). ■

Problem 9. An ordering $(X, \preceq)$ is linear if for every two$x, y \in X$ at least one of $x \preceq y$ and $y \preceq x$ holds. For example, $(\mathbb{N}_0, \leq)$ is linear but $(\mathbb{N}_0^2, \leq)$ is not.

Let$(X, \preceq)$be an ordering. We say $(X, \preceq')$ is an extension of $(X, \preceq)$ if $\preceq \subseteq \preceq'$. Less formally, if $x \preceq y$ implies $x \preceq' y$ for every$x, y \in X$. For example, on $X = \mathbb{N}$) the " $\leq$ " is an extension of "$|$" (here, $a|b$ means a divides b).

We say$(X, \preceq')$ is a linear extension of $(X, \preceq)$ if it is an extension and it is a linear ordering. Show that every finite ordering $(X, \preceq)$ has a linear extension.

Proof. Let $P = (X, \preceq)$ where X is a finite set. If P is a linear order, then P is its own linear extension. Assume, now, P is not a total order. Suppose x and y are incomparable in P. We define a new relation,$\preceq'$ on X as follows: Let $s, t \in X$, we have $s \preceq' t$ provided either of the following conditions holds: 1) $s \prec t$ or 2) $s \prec x$ and $y \prec t$. Then clearly $P' = (X, \preceq^{'})$ is a extension of P and $x \preceq^{'} y$. If P' is not linear, we could continue and eliminate incomparable paris. And since the X is finite, we will eventually result in a poset that is a linear extension to P. ■

Problem 10. Let $(A, \preceq)$ be a finite partial order. Show that $\preceq$ can be written as the intersection of linear extensions on $\preceq$. That is, there are linear extensions$\leq_1$ $, \cdots, \leq_N$ of $\preceq$ such that$\preceq = \bigcap_{i=1}^{N} \leq_i$.

Using less awfully formal notation, this means that for every $x, y \in A$, $x \preceq y$ holds if and only if $x \leq_i y$ for every $1 \leq i \leq N$.

Proof. Let $\preceq$ be a partial order on A, and let $\{\leq_i\}$ be the set of all linear orders which extend $\preceq$. The intersection of the orders in $\{\leq_i\}$ certainly contains $\preceq$,i.e. $\preceq \subseteq \left(\bigcap_{i=1}^{N} \leq_i\right)$

We show $\preceq \supseteq \left(\bigcap_{i=1}^{N} \leq_i\right)$. So suppose that a and b are incomparable in $\preceq$. By the previous question, there is a total order extending $\preceq$ in which $a \leq b$, and another in which $b \leq a$. So in the intersection of these total orders, a and b are still incomparable. This means that $\preceq^{\mathrm{C}} \subseteq \left(\bigcap_{i=1}^{N} \leq_i\right)^{\mathrm{C}}$, so $\preceq \supseteq \left(\bigcap_{i=1}^{N} \leq_i\right)$. And we are done. ■

Problem 11. Let $(A, \preceq)$ be a partial order. The dimension of this partial order is the smallest number N such that $\preceq$ can be written as the intersection of N linear extensions of it. In the previous exercise you showed that every finite partial order has finite dimenion. Show that the partial order $\leq$ on $0, 1^n$ has dimension at most n. That is,$\leq$ can be written as the intersection of n linear extensions of $\leq$.

Proof. This is an application of embedding theorem. We note x_i the i-th element of $x \in \{0, 1\}^n$. We define n linear orders L_i on $\{0, 1\}^n$ by $x <_i y$ if$x_i < y_i$, and if

$x_i = y_i$, then there must be a lowest index $j \neq i$ such that $x_j \neq y_j$ and we define $\begin{cases} x \leq_i y & \text{if } x_j < y_j \\ x \geq_i y & \text{if } x_j > y_j \end{cases}$. The L_i is a linear order on $\{0,1\}^n$.

claim 1: L_i is a linear extension of $P = (\{0,1\}^n, \leq)$. Suppose $x <_P y$ in P, then $x_i \leq y_i$. If $x_i < y_i$ then $x <_i y$. If $x_i = y_i$, then since $x \leq_P y$, there exist an index j such that $x_j < y_j$, so $x \leq_i y$. So $\leq_P \subseteq (\bigcap_{i=1}^n \leq_i)$.

claim 2: if x and y are incomparable in P, then there are indices i and j with $x <_i y$ and $x >_j y$. Since x and y are incoparable in P, we know there exist i, j such that $x_i < y_i$ and $x_j > y_j$. So $\leq_P \supseteq (\bigcap_{i=1}^n \leq_i)$.

So we are done. ■

Problem 12. Let $(A, \preceq)$ be a finite partial order. Show that the dimension of $\preceq$ is at most $|A|$. That is, $\preceq$can be written as the intersection of $|A|$ linear extensions.

If$|A| = n$, then there exists an injection $f : (A, \preceq) \to P = (\{0,1\}^n, \leq)$ that preserves the orders. Therefore, since P has dimension n, $(A, \preceq)$ must have dimension no greater than n.

Relation and functions

By $[n]$ we denote the set $\{1, \ldots, n\}$. A function $f : [m] \to [n]$ is called monotone if $f(i) \leq f(j)$ whenever $i < j$. Let $S(m, n)$ be the number of monotone injective functions from $[m]$ to $[n]$. Find an explicit formula for $S(m, n)$.

Solution. Since f is injective, f is then strictly monotone,i.e.,$f(i) < f(j)$ whenever $i < j$.

1: (Boundary condition) If $m = 1$, then $S(1, n) = n$.

2: if $m > n$, then there will be no injective function. So $S(m, n) = 0$, if $m > n$.

3: if $m = n$, then $S(m, n) = 1$

4. (Recurrence) if $m < n$. Two case:

a) $f(1) = 1$. Then there will be $n-1$ numbers left in $[n]$ viables for $\{f(2), \ldots, f(m)\}$,the number of choices equals to $S(m-1, n-1)$.

b) $f(1) \neq 1$. Then there will be $n-1$ numbers left in $[n]$ viables for $\{f(1), f(2), \ldots, f(m)\}$, the number of choices equals to $S(m, n-1)$.

So $S(m, n) = S(m-1, n-1) + S(m, n-1)$. Inspired by the well known combinatorial equality $C_{n-1}^k + C_{n-1}^{k+1} = C_n^{k+1}$, we claim that $S(m, n) = C_n^m$. This could be proven by an double induction on m.

Base case: if $m = 1$, $S(m, n) = n = C_n^1$.

Induction hypo: Suppose it if true for $m = k$, that $S(k, n) = C_n^k$.

Induction phase:If $n < k+1$, then $S(k+1, n) = 0$.$S(k+1, k+1) = 1$. For $n > k+1$,we have $S(k+1, n) = S(k, n-1) + S(k+1, n-1)$. Then we use another induction from $n = k+2$, it is easy to show $S(k+1, n) = C_{n-1}^k + C_{n-1}^{k+1} = C_n^{k+1}$.

Then we are done.

Note there is an combinatorial argument for this: The number of choice of monotone injective functions from $[m]$ to $[n]$ equals to the the number of combination to choose m from n. Indeed, the image of f in $[n]$ contains m elements. Once a subset of $[n]$ with m elements is fixed, there will be only 1 way to construct the monotone function.

3. Counting basic objects

Problem 13. By $[n]$ we denote the set $\{1, \ldots, n\}$. A function $f : [m] \to [n]$ is called monotone if $f(i) \leq f(j)$ whenever $i < j$. Let $S(m, n)$ be the number of monotone injective functions from $[m]$ to $[n]$. Find an explicit formula for $S(m, n)$.

Solution. Since f is injective, f is then strictly monotone,i.e.,$f(i) < f(j)$ whenever $i < j$.

1: (Boundary condition) If $m = 1$, then $S(1, n) = n$.

2: if $m > n$, then there will be no injective function. So $S(m, n) = 0$, if $m > n$.

3: if $m = n$, then $S(m, n) = 1$

4. (Recurrence) if $m < n$. Two case:

a) $f(1) = 1$. Then there will be $n-1$ numbers left in $[n]$ viables for $\{f(2), \ldots, f(m)\}$,the number of choices equals to $S(m - 1, n - 1)$.

b) $f(1) \neq 1$. Then there will be $n-1$ numbers left in $[n]$ viables for $\{f(1), f(2), \ldots, f(m)\}$, the number of choices equals to $S(m, n - 1)$.

So $S(m, n) = S(m - 1, n - 1) + S(m, n - 1)$. Inspired by the well known combinatorial equality $C_{n-1}^{k} + C_{n-1}^{k+1} = C_{n}^{k+1}$, we claim that $S(m, n) = C_{n}^{m}$. This could be proven by an double induction on m.

Base case: if $m = 1$, $S(m, n) = n = C_{n}^{1}$.

Induction hypo: Suppose it if true for $m = k$, that $S(k, n) = C_{n}^{k}$.

Induction phase:If $n < k + 1$, then $S(k + 1, n) = 0.S(k + 1, k + 1) = 1$. For $n > k + 1$,we have $S(k+1, n) = S(k, n-1) + S(k+1, n-1)$. Then we use another induction from $n = k + 2$, it is easy to show $S(k + 1, n) = C_{n-1}^{k} + C_{n-1}^{k+1} = C_{n}^{k+1}$.

Then we are done.

Note there is an combinatorial argument for this: The number of choice of monotone injective functions from $[m]$ to $[n]$ equals to the the number of combination to choose m from n. Indeed, the image of f in $[n]$ contains m elements. Once a subset of $[n]$ with m elements is fixed, there will be only 1 way to construct the monotone function.

Problem 14. Let $T(m, n)$ be the number of monotone functions That is, $f : [m] \to [n]$ such that $1 \leq i \leq j \leq m$ implies $f(i) \leq f(j)$. Find an explicit formula to $T(m, n)$.

Solution. Here we will use an combinatorial argument . Denote by for $i \in [n]$, k_i the number of elements in the set $f^{-1}(i)$, k_i could be 0. The the solution fo the following integer equation $k_1 + k_2 + \ldots + k_n = m,\ k_i \geq 0 \forall i \in [n]$ $(*)$ uniqely determines a monotone functions f . We claim there is a bijection between the set of monotone functions f from $[m]$ to $[n]$ and the solution set of the equation (*).

Given a solution $\{k_i\}$ to equation $(*)$. We construct our f. We start with k_1. Suppose $k_1 \neq 0$, we set $f(1) = f(2) = \cdots = f(k_1) = 1$. If $k_1 = 0$, then 1 is not in the image set $f([n])$. Also we note by l_i the minimum of free numbers in $[n]$(x in $[n]$ that have not been given an value $f(x)$ at step 1). And we procede to $k_2, k_3, \cdots, k_n$, setting $f(l_{k_{p-1}}) = f(l_{k_{p-1}} + 1) = \cdots = f(l_{k_{p-1}} + k_p - 1) = p$ for those $k_p \neq 0$ and keeping track l_p. The resulting f is monotone.

Conversley, given a monotone function f, by setting $k_i = |f^{-1}(i)|$, $\{k_i\}$ is a solution to equation $(*)$.

Then we need to count the number of solutions to equaiton $(*)$. This is an establish question of stars and bars. $T(m, n) = C_{m+n-1}^{n-1}$. Why this works? Let's consdier the problems of placing $n - 1$ bars and m stars into $m + n - 1$ bins. Then

from left to right, k_1 = numbers of stars fall at the left side of first bar. k_i = numbers of stars fall between the $(i-1)$-th and the i-th bars. Clearlly, $k_1+k_2+\ldots+k_n = m$.

So $T(m,n) = C_{m+n-1}^{n-1}$.

Note there is also a combinatoric argument:

Task A : We have m identical candies, and we want to distribute them among n kids, with some kid(s) possibly getting no candy.

Task B : Distribute $m+n$ candies among the kids, with each kid getting at least 1candy. Then take away a candy from each kid.

It is clear that there are just as many ways to carry out Task B as there are to carry out Task A.

And task B become finding the number of solutions for $\sum_{i=1}^{n} p_i = m+n$ with $p_i \geq 1(**)$. $(**)$ is conceptually easier than $(*)$ since we do not need to care the empty sets. The answer is C_{m+n-1}^{n-1}. Why, C_{m+n-1}^{n-1} is the ways of selecting $n-1$ spaces between $m+n$ adjacent objects as the dividing points to form n nonempty subsets.

Problem 15. A function $\{0,\ldots,n\} \to \mathbb{Z}$ is a walk function if consecutive values differ by excatly 1, i.e., $|f(i)-f(i-1)| = 1$ for all $i = 1,\ldots,n$. What is the number of walk function from $\{0\ldots,n\}$ to $\mathbb{Z}$ with $f(0) = 0$? Find such formula.

Solution. Consider the graph of our walk in $\mathbb{Z}^2$. We have $f(0) = 0$. From $i-1$ to i, f either goes up 1 unit or goes down 1 unit. Since $f(n)$ have travelled all together n steps, the number of walk is 2^n.

Problem 16. What is the number of walk functions from $\{0,\ldots,n\}$ to $\mathbb{Z}$with $f(0) = f(n) = 0$? Try to find an explicit formula.

Solution. For $f(0) = f(n) = 0$ to be true, there must be equal numbers of ups and downs in the trajectory of f. So for n odd, the number of walk is 0.

For n even, this means that there are $n/2$ ups and $n/2$ downs. The number of walks equals to the ways of putting without considering the order $n/2$ red balls and $n/2$ blue balls into n bins. The anwer is $C_n^{n/2}$.

Problem 17. Recall our definition of $M(n)$, the number of almost injective monotone functions $f : [n] \to [n]$. Almost injective means that for each $j \in [n]$ there are at most two $i \in [n]$ with $f(i) = j$. We define $L(n)$ as the number of functions from $0,1,\ldots,n$ to $0,1,\ldots,n$ with $|f(i)-f(i-1)| \leq 1$ for all$i \in [n]$ and $f(0) = f(n) = 0$. Prove that $L(n) = M(n)$.

Solution. First consider the graph of functions from $0,1,\ldots,n$ to $0,1,\ldots,n$ with $|f(i)-f(i-1)| \leq 1$. So for $i \in [n]$, we have $f(i)-f(i-1) \in \{-1,0,1\}$ and $\sum_{i=1}^{n}(f(i)-f(i-1)) = 0$.

Secondly, we note by $AIM(n,n)$ the set of all almost injective monotone functions $f : [n] \to [n]$. Inspired by the solution of question 2, $|AIM(n,n)|$ equals the number of the solution set of the integer equation $k_1+k_2+\ldots+k_n = n(*)$, with a restriction that $\forall i \in [n], k_i \in \{0,1,2\}$. With a change of variable $p_i = k_i - 1$, $(*)$ becomes $\sum_{i=1}^{n} p_i = 0$, with $p_i \in \{-1,0,1\}$. So there is a natural bijection between $AIM(n,n)$ and the set of functions from $0,1,\ldots,n$ to $0,1,\ldots,n$ with $|f(i)-f(i-1)| \leq 1$ for all$i \in [n]$ and $f(0) = f(n) = 0$. (indeed, $f(0) = f(n)$ will suffice.)

So $L(n) = M(n)$.

Problem 18. Find a closed formula for $L(n)$.

$M(n)$ is number of the solutions of constrainted integer equation $k_1 + k_2 + \ldots + k_n = n, k_i \in \{0, 1, 2\}$ $(Eq1)$. Using a generating function trick, $M(n)$ equals to the coefficient of x^n in the polynomial $A(x) = \prod_{i=1}^{n}(\frac{1-x^{n+1}}{1-x})$. But i am not able to come to an explicit formula. Also we can use an inclusion-exclusion argument, but this wont help to find an explicit formula either.

Also, in the Lipschitzien walk problem $L(n)$, there are same number of ups and dowsn, and the number of paris of up and down are in the set $\{0, 1, \ldots, \lfloor n/2 \rfloor\}$. Given the number of pairs of up and down equals to i then the problem become choose $2i$ in n and choose i in $2i$. So $L(n) = \sum_{i=0}^{\lfloor n/2 \rfloor} C_n^{2i} \times C_{2i}^{i}$.

4. Basic Combinatorial Identity

Problem 19. find a closed formlua for the sum $\sum_{r=0}^{k} \binom{n}{r}\binom{n-r}{k-r}$

Solution. The answer is $2^k\binom{n}{k}$.

Analytical approch:

$$\begin{aligned}\sum_{r=0}^{k} \binom{n}{r}\binom{n-r}{k-r} &= \sum_{r=0}^{k} \frac{n!}{(n-r)!r!} \times \frac{(n-r)!}{(n-k)!(k-r)!} \\ &= \frac{n!}{(n-k)!}\sum_{r=0}^{k} \frac{1}{r!(k-r)!} = \frac{n!}{(n-k)!}\sum_{r=0}^{k} \frac{1}{k!}\binom{k}{r} \\ &= \binom{n}{k}\sum_{r=0}^{k}\binom{k}{r} = 2^k\binom{n}{k}.\end{aligned}$$

$\sum_{r=0}^{k}\binom{k}{r} = 2^k$ since given an subset A of k elements, $\binom{k}{r}$ is the number of its subsets containing r elements, so summing over $r = 0$ to $r = k$ we get the number elements in the power set of A.

Combinatorial thinking:

We have a set X with n elements. We claim $\binom{n}{r}\binom{n-r}{k-r}$ it the number of choices of doing the following 2 procedures :

Proc I:

1. In X choose an r-elements subset $B \subset X$,
2. In $X \backslash B$ choose a subet C with $k - r$ elements.

Note $D = B \cup C$, then $|B \cup C| = k$. So in procedure I we firstly choose a subset B of r-elements and then choose a subest D of k-elements that $B \subset D$. The order of selection could be reversed and the number of choices is still the same. To be more precise, consider Procedure II as the following.

Proc II:

1.In X choose a subset D of k elements .

2. In D choose an r-elements subset B.

Now the first part of Procedure II does not depend on r, so doing Proc II over $r \in \{0, 1, \ldots, k\}$ is equal to do the following:

Proc III:

1.In X choose a subset D of k elements .

2. Choose $r \in \{0, 1, \ldots, k\}$.

3.In D choose an r-elements subset B.

But the step 2 and 3 of Proc III is just enumerating all the subset of D, i.e., the power set of D. So the number of choices of doing Proc III is $\binom{n}{k} \times 2^k$.

Also doing Proc I over $r \in \{0, 1, \ldots, k\}$ is $\sum_{r=0}^{k} \binom{n}{r}\binom{n-r}{k-r}$. Since the number of choices of Proc I and Proc II are the same, we have $\sum_{r=0}^{k} \binom{n}{r}\binom{n-r}{k-r} = \binom{n}{k} \times 2^k$.

Problem 20. Consider the "grid graph" on $\{0, 1, \ldots, m\} \times \{0, 1, \ldots, n\}$. A monotone path is a path from$(0, 0)$ to (m, n) that in every step (i) goes one to the right or (ii) one up, but never down, left, or diagonal.

What is the number of monotone paths from (0,0) to (m,n)?

Solution. So there are m right moves and n up moves. So the number of path is $\binom{m+n}{n}$.

Problem 21. What is the number of monotone paths from (0,0) to (m,n) that pass through the point (i,j)?

Solution. The answer is the product of the number of paths form $(0, 0)$ to (i, j) and the number of paths from (i, j) to (m, n). Using the previous result, the answer is $\binom{i+j}{i} \times \binom{m+n-i-j}{m-i}$.

Problem 22. Note that the monotone paths in the picture below passes through the edge from (i−1,j) to (i,j)

What is the number of monotone paths passing through the edge from (i−1,j) to (i,j)?

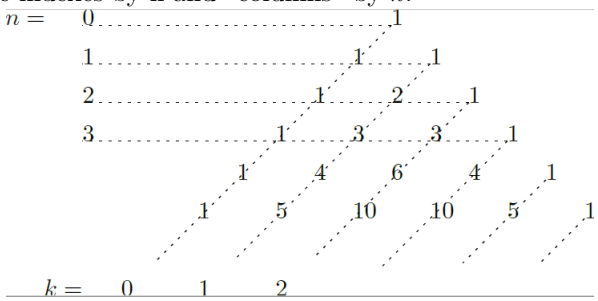
Solution. The answer is the product of the number of paths form $(0, 0)$ to $(i - 1, j)$ and the number of paths from (i, j) to (m, n). The answer is then $\binom{i+j-1}{i-1} \times \binom{m+n-i-j}{m-i}$

Problem 23. Give a closed formula for the sum$\sum_{j=0}^{n} \binom{i+j-1}{i-1} \times \binom{m+n-i-j}{m-i}$

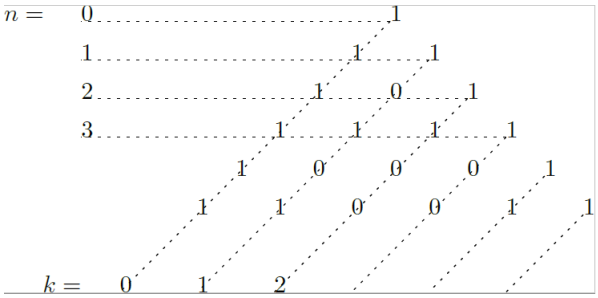
The answer is $\binom{m+n}{m}$.

Inspired by the previous solution, $\sum_{j=0}^{n} \binom{i+j-1}{i-1} \times \binom{m+n-i-j}{m-i}$ is the number of the paths connecting $(0, 0)$ and (m, n) passing through the edge from $(i - 1, j)$ to (i, j) for some $j \in \{0, 1, \ldots, n\}$. Then the sum is just the number of the paths connecting $(0, 0)$ and (m, n) since all paths must some edge from $(i - 1, j)$ to (i, j) for some $j \in \{0, 1, \ldots, n\}$.

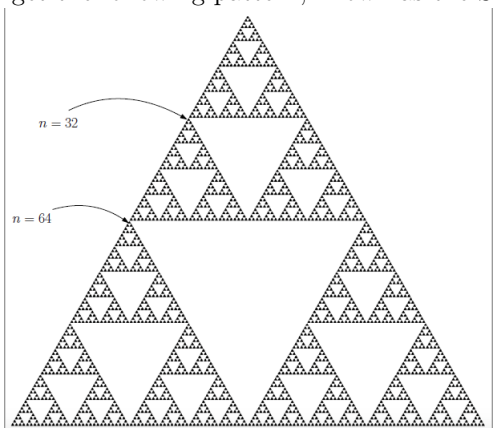
Pascal's Triagle. Here is my version of "Pascal's triangle", indicating that rows are indexes by n and "columns" by k.



Something interesting happens when we take the triangle modulo 2, that is, we replace even numbers by 0 and odd numbers by 1:



If we draw a black dot for every 1 and look at a larger section of this triangle, we get the following pattern, known as the Sierpinski triangle:



Note that the rows $n = 32$ and$n = 64$ are empty expect for the leftmost and rightmost position. Your task now is to explain this phenomenon of the Sierpinski's triangle.

To be more precise, show that for$n = 2^d$ being a power of two, the binomial coefficient $\binom{n}{k}$ is even except for $k = 0$ and $k = n$ (where it is 1, of course). I will walk you through it.

Suppose n is a power of 2. Prove that $\binom{n}{k}$ is even unless $k = 0$ or $k = n$. Do this by counting how often the factor 2 appears in the denominator and the numerator.

Proof. $\binom{n}{k} = \frac{n\cdot(n-1)\cdot(n-2)\cdot\cdots\cdot(n-k+2)(n-k+1)}{k\cdot(k-1)\cdot(k-2)\cdot\cdots\cdot2\cdot1}$.

For $k = 0$ or $k = n$, clearly $\binom{n}{k} = 1$.

If k is even, in the numerator there are $k/2$ even factors : $n, (n-2), \cdots, (n-k+2)$. And in the denominator, there are also $k/2$ even factors : $k, (k-2), \cdots, 2$. So we can write $\binom{n}{k} = \frac{n\times(n-2)\times\cdots\times(n-k+2)}{k\times(k-2)\times\cdots2} \times K$ for some K containing no factors of 2. Now we study $\frac{n\times(n-2)\times\cdots\times(n-k+2)}{k\times(k-2)\times\cdots2}$. We observe that $(n-2)\times\cdots\times(n-k+2)$ and $2\times4\times\cdots\times(k-2)$ share the same factors of 2. So $\frac{n\times(n-2)\times\cdots\times(n-k+2)}{k\times(k-2)\times\cdots2} = \frac{n}{k} \times K^{'}$ with $K^{'}$ containing no factors of 2. For n/k, there exist a postive integer b such that $\frac{n}{k} = 2^b \times K^{''}$ with $K^{''}$ containing no factors of 2.

So for an even number k suth that $0 < k < 2^d = n$, there exist a postive integer b such that $\binom{n}{k} = 2^b \times L$ with L containing no factors of 2. But since $\binom{n}{k}$ is an integer, by the unique decompositon of integer, we conclude that $\binom{n}{k}$ is even.

If k is uneven, the analysis is the same, and we can prove that $\binom{n}{k}$ is also even.

So we are done. ■

Let me now walk you through a more "combinatorial" proof of this fact. This is also valuable because it lets you practice with notions of sets and functions. Consider the set $\{0,1\}^d$. You can view this as the set of all binary strings of length d. For $1 \leq i \leq d$ and $x \in \{0,1\}^d$ let $f_i(x)$ be x with the ith position flipped. For example, $f_3(11011) = 11111$.

Let $d \geq 1$ and $1 \leq i \leq d$. Show that f_i is an involution without a fixed point. That is, $f(f(x)) = x$ and $f(x) \neq x$ for all $x \in \{0,1\}^d$.

Proof. Clearly $f_i(f_i(x)) = x$ since we flipped the ith position twice. Note x_i the value in the ith position of the binary string.

Supose $\exists x \in \{0,1\}^d$ such that $f_i(x) = x$. Then for this x , $(f_i(x))_i = x_i + 1 \bmod 2 = x_i$. But this is not possible. So by contradiction, f_i does not have a fixed point. ■

Let $S \subseteq \{0,1\}^d$. We define $f_i(S)$ as the set arising from applying f_i to every element of S. Formally, $f_i(S) = \{f_i(x) | x \in S\}$. Given a set $S \subseteq \{0,1\}^d$, we call an index $i \in [d]$ active for S if $f_i(S) \neq S$.

Let $d = 3$ and $S = \{000, 100\}$. Which of the indices $1, 2, 3$ are active?

Solution. Index 2,3 are active. Index 1 is not since $f_1(000) = 100, f_1(100) = 000$.

Let$S \subseteq \{0,1\}^d$. Show that if $S \neq \emptyset$ and$S \neq \{0,1\}^d$ then S has at least one active index.

Proof. We call the S satisfies P_i if $\forall x \in S, f_i(x) \in S$. That means all elements in S can be partitioned into S_1 and S_2 with same number of elements and the paris $(a, b | a \in S_1, b \in S_2)$ only differs in its ith position.

Suppose S does not have active index. Then then S satisfies $P_1, P_2, \ldots, P_n$, which means that for each $x \in S$, and for each $i \in [d]$, we can find one string with different values in i th position. But this means that $S = \{0,1\}^d$.

So by contradiction, S has at least one active index. ■

Given$S \subseteq \{0,1\}^d$, define $f(S)$ as follows:if $S \neq \emptyset$or $S \neq \{0,1\}^d$ define $f(S) = S$. Otherwise, let $f(S) := f_i(S)$ where i is the smallest active index of S (which exists by the previous exercise).

Show that f is an involution. That is, $f(f(S)) = S$. Furthermore, show that the only fixed points of f are $\emptyset$ and $\{0,1\}^d$.

Proof. 1. Now for a given $S \neq \{\emptyset, \{0,1\}^d\}$, we have for some i such that $f(S) = f_i(S)$. By def, i is the smallest active index of S.

Claim A:i is an active index of $S' = f(S)$. Proof: $f_i(S) \neq S$. Therefore $f_i(f_i(S)) \neq f_i(S)$. So $f_i(f(S)) \neq f(S)$. This proves Claim A.

Claim B : Any $j < i$ is not an active index for $f(S) = f_i(S)$. Proof: We note that the order in which we flip digits does not matter, so $f_i(f_j(x)) = f_j(f_i(x))$. Let $x \in S$. So $f_j(x) \in S$ since j is not an index of S. Then $f_i(f_j(x)) \in f_i(S) = f(S)$. This means that $f_j(f_i(x)) \in f(S)$. So $f_j(f(S)) \subseteq f(S)$. But since f_j is a bijection and $f(S)$ is finite, by pigenhole, we must have $f_j(f(S)) = f(S)$, i.e., $j < 1$ is not an active index of $f(S)$. This proves claim B.

Putting together claim A, and claim B, we can say that i is the smallest active index of $f(S)$. This means that $f(f(S)) = f_i(f(S)) = f_i(f_i(S)) = S$. Therefore, f is an involution.

2. For $S \neq \{\emptyset, \{0,1\}^d\}$, $f(S) := f_i(S)$ for some i. Then by definition of f_i, $f(S) \neq S$. But for $S \neq \{\emptyset, \{0,1\}^d\}$, $f(S) = S$ by definition. ■

Let $\mathcal{S} = \binom{\{0,1\}^d}{k}$. This is a set of sets, and each set $S \in \mathcal{S}$ consists of exactly k strings from $\{0,1\}^d$.

1. Show that f defines a bijection on $\mathcal{S}$.

2. Furthermore, if $1 \leq k \leq 2^d - 1$ then this bijection is an involution without fixed points.

3. Show that $\mathcal{S}$ has even size, if $1 \leq k \leq 2^d - 1$. Conclude that for $n = 2^d$, the binomial coefficient $\binom{n}{k}$ is even unless $k = 0$ or $k = n$.

Proof. 1. One to one: if $f(S_1) = f(S_2)$, then we have $S_1 = f(f(S_1)) = f(f(S_2)) = S_2$.

Onto: For all $S \in \mathcal{S}$, we have $f^{-1}(S) = f(S) \in \mathcal{S}$ since $f(f(S)) = S$.

So f is bijective.

2. Since the only fixed point of f are $\emptyset = \binom{\{0,1\}^d}{0}$ and $\{0,1\}^d = \binom{\{0,1\}^d}{2^d}$, f is a bijection without fixed points for $1 \leq k \leq 2^d - 1$.

3.Then for $1 \leq k \leq 2^d - 1$, f defines an equivlant relation $\mathcal{R}$ in $\binom{\{0,1\}^d}{k}$, $S_1 \mathcal{R} S_2$ iff $S_2 = f(S_1)$. And since $\forall S \in \binom{\{0,1\}^d}{k}$, $f(S) \neq S$, there are exactly 2 equivlant partitions of $\binom{\{0,1\}^d}{k}$ defined by $\mathcal{R}$. This means that the number of element of set $\binom{\{0,1\}^d}{k}$ is even. Now $\left|\binom{\{0,1\}^d}{k}\right| = \binom{2^d}{k}$. So $\binom{2^d}{k}$ is even for $1 \leq k \leq 2^d - 1$. ■

Let $n = p^d$ where p is a prime number. Show that p divides $\binom{n}{k}$ unless $k = 0$ or $k = n$. Do this by counting how often the factor p appears in the enumerator and denominator of $\binom{n}{k}$.

Proof. We follow the same steps as the case $p = 2$. First seperate the terms in to terms contain p as factor and terms do not contain p as factor.

For $k \geq p$, there exist $b \geq 1$, K, K', K'' do not contain p as factor such that $\binom{n}{k} = \frac{n \times (n-p) \times \cdots \times (n-p(\lfloor k/p \rfloor - 1))}{\lfloor k/p \rfloor p \times 2p \times \cdots p} \times K = \frac{n}{\lfloor k/p \rfloor p} \times K' = p^b K''$.And since $\binom{n}{k}$ is an integer, p divides $\binom{n}{k}$.

For $1 \leq k < p$, there exist K not containing p as factor such that $\binom{n}{k} = nK = p^d K$. Then p also divides $\binom{n}{k}$. ■

Prove the same statement as in the previous exercise, but now by generalizing the "combinatorial"proof above for $p = 2$.

Proof. The key for a "combinatorial" proof for any prime $p \neq 2$ is to consider the set $\{0, 1, \cdots, p-1\}^d$, and its string form. Then we define the function f_i as $(f_i(x))_i = x_i + 1 \bmod p$ and $(f_i(x))_j = x_j$ for $j \neq i$. Then $x \neq f_i(x) \neq f_i^2(x) \neq \cdots \neq f_i^{p-1}(x)$ and $f_i^p(x) = x$. So we could define active index $i \in [d]$ for S if $f_i(S) \neq S$. Also it is true that apart from $\emptyset$ and $\{0, 1, \cdots, p-1\}^d$, all $S \subset \{0, 1, \cdots, p-1\}^d$ has at least one active index. And we could define f as $f(\emptyset) = \emptyset, f(\{0, 1, \cdots, p-1\}^d) = \{0, 1, \cdots, p-1\}^d$,and otherewise $f(S) = f_i(S)$ where i is the smallest active index of S. Now f define a bijection on the set $\binom{\{0,1,\cdots,p-1\}^d}{k} = \mathcal{S}$. And for $k \neq 0$ and $k \neq p^d$, $\forall S \in \mathcal{S}, f^p(S) = S$ and $S \neq f(S) \neq f^2(S) \neq \cdots \neq f^{p-1}(S)$. So for $k \neq 0$ and $k \neq p^d$, f define an equivalent relationship $\mathcal{R}$ on $\mathcal{S}$ such that for each $s \in \mathcal{S}$, the equivalant class $[s]$ contains p elements. That means $\left|\binom{\{0,1,\cdots,p-1\}^d}{k}\right|$ is divisible by

p. And since $\left|\binom{\{0,1,\cdots,p-1\}^d}{k}\right| = \binom{p^d}{k}$, this conclude the proof for $k \neq 0$ and $k \neq p^d$. And for $k = 0$ or $k = p^d$, $\binom{p^d}{k} = 1$. This conclude the proof. ■

5. Asymptotics and the $O()$ notation

Problem 24. Prove formally the staments from the video lecture:

1. If $g_1 \in O(f_1)$ and $g_2 \in O(f_2)$ then $g_1 + g_2 \in O(f_1 + f_2)$.
2. If $g_1 \in O(f_1)$ and $g_2 \in O(f_2)$ then $g_1 \cdot g_2 \in O(f_1 \cdot f_2)$.
3. If $g \in O(f)$ then $\sum_{k=1}^{n} g(k) = O(\sum_{k=1}^{n} f(k))$.
4. If $g \in O(f)$ then $f + g \in O(f)$.
5. $n^a \in O(n^b)$ whenever $a \leq b$.
6. $n^a \in O(c^n)$ whenever $c > 1$.
7. $(\ln n)^C \in O(n^a)$ whenever $a > 0$.

Proof. 1. If $g_1 \in O(f_1)$ and $g_2 \in O(f_2)$ then $g_1 + g_2 \in O(f_1 + f_2)$.

There exist $c_1, c_2 > 0$,$n_1, n_2 \in \mathbb{N}$ such that for $n > n_1$,$g_1 \leq c_1 f_1$ and for $n > n_2, g_2 \leq c_2 f_2$. So by setting $c = \max(c_1, c_2), n_0 = \max(n_1, n_2)$ we have for $n > n_0, g_1 + g_2 \leq c(f_1 + f_2)$, i.e.,$g_1 + g_2 \in O(f_1 + f_2)$.

2. If $g_1 \in O(f_1)$ and $g_2 \in Of(_2)$ then $g_1 \cdot g_2 \in O(f_1 \cdot f_2)$.

There exist $c_1, c_2 > 0$,$n_1, n_2 \in \mathbb{N}$ such that for $n > n_1$,$g_1 \leq c_1 f_1$ and for $n > n_2, g_2 \leq c_2 f_2$. So by setting $c = c_1 \times c_2, n_0 = \max(n_1, n_2)$, we have for $n > n_0$, $g_1 \cdot g_2 \leq c \times (f_1 \cdot f_2)$, i.e., $g_1 + g_2 \in O(f_1 + f_2)$.

3.If $g \in O(f)$ then $\sum_{k=1}^{n} g(k) = O(\sum_{k=1}^{n} f(k))$.

There exist $c_1 > 0$ and $n_1 \in \mathbb{N}$ such that for $n > n_1$, $g \leq cf$. So for $n > n_1$, $\sum_{k=1}^{n} g(k) = \sum_{k=1}^{n_1} g(k) + \sum_{k=n_1+1}^{n} g(k) = \sum_{k=1}^{n_1} g(k) + O(\sum_{k=n_1+1}^{n} f(k))$. But $\sum_{k=1}^{n_1} g(k)$ is a constant, clearly $\sum_{k=1}^{n_1} g(k) = O(\sum_{k=1}^{n_1} f(k))$ (it suffice to choose $c = \frac{\sum_{k=1}^{n_1} g(k)}{\sum_{k=1}^{n_1} f(k)} + 1$). So $\sum_{k=1}^{n} g(k) = O(\sum_{k=1}^{n_1} f(k)) + O(\sum_{k=n_1+1}^{n} f(k)) = O(\sum_{k=1}^{n} f(k))$.

4.If $g \in O(f)$ then $f + g \in O(f)$.

There exist $c_1 > 0$ and $n_1 \in \mathbb{N}$ such that for $n > n_1$, $g \leq c_1 f$. Then choose $c = c_1 + 1$, for $n \geq n_1, f + g \leq c(f)$, i.e., $f + g \in O(f)$.

5.$n^a \in O(n^b)$ whenever $a \leq b$.

We have $n^a \leq n^b$ for $n \geq 1$, since $\log(n^a) = a \log n \leq b \log n = \log(n^b)$.

6.$n^a \in O(c^n)$ whenever $c > 1$.

Suppose $a > 0$,we havelog $n^a = a \log n$, $\log c^n = n \log c$, and $\log n \leq n \times \frac{\log c}{a}$ for some $n \geq n_0 > 1$. Since log is monotone, we have $n^a \leq c^n$ for $n \geq n_0$. The case for $a < 0$ is trivial since $n^a < 1$then.

7.$(\ln n)^c \in O(n^a)$ whenever $a > 0$.

Suppose also $c > 0$. We haveln$(\ln n)^c = c \ln \ln n$, and $\ln n^a = a \ln n$. We also have $\ln \ln n \leq \frac{a}{c} \ln n$ for some $n \geq n_0 > 1$. So we have $(\ln n)^c \leq n^a$ for $n \geq n_0$.The case for $c < 0$ is trivial since $(\ln n)^c < 1$ for large enough n if $c < 0$. ■

Problem 25. Ture of false: If $g \in O(f)$ then $g^2 \in O(f^2)$.

Ture. Since this is the consequence of property 2 in the Problem 1.

Problem 26. Ture of false: If $g \in O(f)$ then $\log g \in O(\log f)$.

In genearl false.

It is ture as long as $f(n) \geq 1$ for sufficiently large n. Suppose $g \in O(f)$, then there exist $c_1 > 0$ and $n_1 \in \mathbb{N}$ such that for $n \geq n_1$, $g \leq c_1 \cdot f$ and $f(n) > 1$, which implies $\log g \leq \log c_1 + \log f$. It suffice to set $c = (1 + \log c_1)$.

But for the case f is below 1, for example $g(n) = 1$ and $f(n) = \frac{1}{2} + \frac{1}{n}$, then $\log g = 0$ and $\log f = \log(\frac{1}{2} + \frac{1}{n}) < 0$ for $n > 2$. So $\log g > \log f$ for sufficiently large n and there could be no positive c such that $\log g \leq c \log f$.

Problem 27. Ture of false: If $g \in O(f)$ then $\log g = \log(f) + O(1)$

True. As in the previous problem , with the same notation, we have $\log g \leq \log c_1 + \log f$. And for $\log c_1$ is $O(1)$ with a $c = |\log c_1| + 1$. In your case, then $\log g - \log f < 0 < 1$, so $O(1)$. Maybe my understanding is not correct? But if we use the normed version, then I agree the statatement is not true.

Problem 28. Ture of false: If $g \in O(f)$ then $2^g \in O(2^f)$.

False. Take $g(n) = 2n$ and $f(n) = n$, then $2^g = 4^n$ and $2^f = 2^n . \frac{2^g}{2^f} = 2^n$, which is exponetial. So there does not exist c such that for large enough n, $2^g \leq c \times 2^f$.

Classes that often occur in complexity theory.

Problem 29. We define some sets of functions $\mathbb{N} \to \mathbb{N}$.
poly $:= \bigcup_{c \geq 1} O(n^c)$
quasipol:$= \bigcup_{c \geq 1} O(2^{\log^c(n)})$
stronglySubExp:$= \bigcap_{\varepsilon > 0} O(2^{n^\varepsilon})$
weaklySubExp:$= \bigcap_{\varepsilon > 0} O(2^{\varepsilon n})$
E:$= \bigcup_{c \geq 1} O(2^{cn})$
Exp:$= \bigcup_{c \geq 1} O(2^{n^c})$
For example, let A be an algorithm and let $f(n)$ be its worst-case running time on instances of bit-size n. If $f \in$ poly we say that A is a polynomial-time algorithm. Note that f need not be a polynomial. For example, $f(n) = n^2 \log(n)$ is not a polynomial, but surely $f \in$ poly since $f \in O(n^3)$. Similarly, if $f \in$ quasipol we say that A is a quasi-polynomial algorithm. If somebody states that A is a subexponential algorithm, it is not clear what they mean. Depending on the research community or context they might mean $f \in$ stronglySubExp or $f \in$ weaklySubExp. If $f \in$ E we call A a simply exponential algorithm; finally, if $f \in$ Exp we say that A is an exponential algorithm. For example, if the running time of A is $\Theta(n!)$ then A is exponential (since $n! \leq n^n \leq 2^{n^2}$) but not simply exponential, since $n!$ grows faster than 2^{cn}, for any $c \in N$.

Show that the above hierarchy is strict. That is, poly $\subsetneq$ quasipol $\subsetneq$ stronglySubExp $\subsetneq$ weaklySubExp $\subsetneq$ E $\subsetneq$ Exp.

Proof. 1) poly $\subsetneq$ quasipol.

Consider for $a \geq 1, b \geq 1$,$\log \frac{2^{\log^b(n)}}{n^a} = \log^b(n) \log 2 - a \log n$. Then this value is never bounded from above for n sufficently large. So we have proven quasipol is strictly slower than , i.e., poly $\subsetneq$ quasipol.

2) quasipol $\subsetneq$ stronglySubExp.

Now consider for arbitarly $\varepsilon > 0$ and $b \geq 1$, we have $\log \frac{2^{n^\varepsilon}}{2^{\log^b(n)}} = \log 2 \times (n^\varepsilon - \log^b(n))$ not bounded from above for n sufficentily large since $\log^b(n) = o(n^\varepsilon)$. This means that quasipol $\subseteq$ stronglySubExp.

Lef $f = 2^{(\log n)^{(\log(\log n))}}$, then for all $c \geq 1$, we have

$$\log \frac{2^{(\log n)^{(\log(\log n))}}}{2^{\log^c n}} = \log 2 \left((\log n)^{\log(\log n)} - (\log n)^c \right)$$

not bounded from above. So $f \notin$ quasipol.

But for any $\varepsilon > 0$, $\log \frac{n^\varepsilon}{(\log n)^{\log(\log n)}} = \varepsilon \log n - (\log\log n)^2$ not bounded from above, so $f = 2^{(\log n)^{(\log(\log n))}} \in$ stronglySubExp.

Thus quasipol $\subsetneq$ stronglySubExp.

3)stronglySubExp $\subsetneq$ weaklySubExp.

The inclusion is obvious. And $f = 2^{\sqrt{n}} \in$ weaklySubExp but $\notin$ stronglySubExp.

4)weaklySubExp $\subsetneq$ E.

The inclusion is also obvious. And $f = 2^n \in$ E but $\notin$ weaklySubExp.

5)E $\subsetneq$ Exp.

The inclusion is also obious. And $f = 2^{n^2} \in$ Exp but $\notin$ E.

So the above hierarchy is strict. ∎

Problem 30. Let us call a set S of functions $\mathbb{N} \to \mathbb{N}$ closed under multiplication $f \cdot g \in S$ whenever both f and g are in S . It is closed under powers if $f^c \in S$ for every $f \in S$ and $c \in \mathbb{N}$.

1. Show that if S is closed under multiplication then it is closed under powers.
2. Which of the six classes in the above hierarchy are closed under powers?

Proof. 1. This could be proved by induction.

Suppose $f \in S$ closed under multiplication. We claim for all c, $f^c \in S$.

The base case is just power of 2, $f^2 = f \cdot f \in S$ by closure under multiplication.

Suppose $f^{c-1} \in S$. Then $f^c = f^{c-1} \cdot f \in S$.

2.

All of the classes are closed under powers. I will only study the closure under multiplication. The closure under powers is directy result.

1) poly $:= \bigcup_{c\geq 1} O(n^c)$ is closed under multiplication.

This is because for an $c_1, c_2 \geq 1$, $n^{c_1+c_2} \in \bigcup_{c\geq 1} O(n^c)$.

2) quasipol:$= \bigcup_{c\geq 1} O(2^{\log^c(n)})$ is closed under multiplication.

This is because for an $c_1, c_2 \geq 1$, $2^{\log^{c_1}(n)+\log^{c_2}(n)} \leq 2^{\log^{c_1+c_2}(n)} \in \bigcup_{c\geq 1} O(n^c)$.

3) stronglySubExp:$= \bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$ is closed under multiplication.

Suppose $f_1, f_2 \in \bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$. Take an arbitary $\varepsilon > 0$, there exist $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$ we have $f_1 \leq c_1 2^{n^{\varepsilon/2}}, f_2 \leq c_2 2^{n^{\varepsilon/2}}$. Then for $n \geq n_0$, $f_1 f_2 \leq c_1 c_2 2^{2n^{\varepsilon/2}} \leq c_1 c_2 2^{n^\varepsilon}$. The latter inequality is because $\log \frac{2n^{\varepsilon/2}}{n^\varepsilon} = \log 2 - \frac{\varepsilon}{2} \log n < 0$ for large enough n.

So by setting $c = c_1 c_2$, we have proved $f_1 f_2 \in O(2^{n^\varepsilon})$ for any $\varepsilon > 0$. Thus $f_1 f_2 \in \bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$.

4) weaklySubExp:$= \bigcap_{\varepsilon>0} O(2^{\varepsilon n})$ is closed under multiplication.

Suppose $f_1, f_2 \in \bigcap_{\varepsilon>0} O(2^{\varepsilon n})$. Take an arbitary $\varepsilon > 0$, there exist $c_1, c_2 > 0$ and n_0 such that for all $n \geq n_0$ we have $f_1 \leq c_1 2^{n\varepsilon/2}, f_2 \leq c_2 2^{n\varepsilon/2}$. Then for $n \geq n_0$, $f_1 f_2 \leq c_1 c_2 2^{n\varepsilon} \in O(2^{\varepsilon n})$. Since this is true for any $\varepsilon > 0$, we have $f_1 f_2 \in \bigcap_{\varepsilon>0} O(2^{\varepsilon n})$.

5) E:$= \bigcup_{c\geq 1} O(2^{cn})$ is closed under multiplication.

The is because for any $c \geq 1$, 2^{2cn} is still in the set $\bigcup_{c\geq 1} O(2^{cn})$.

6) Exp:= $\bigcup_{c\geq 1} O(2^{n^c})$ is closed under multiplication.

This because for any $c \geq 1$, $2^{2\times n^c} \leq 2^{n^{2c}}$ for $c > 1$. ■

Problem 31. A set S of functions $\mathbb{N} \to \mathbb{N}$ is called closed under argument powering if $f(n^c) \in S$ for all $f \in S$ and $c \in N$. Which of the above classes (poly,quasipol,...) is closed under argument powering?

[Remark.] The notion of being closed under argument powering becomes important when we combine algorithms. Suppose A is an algorithm with worst-case running time $f(n)$ and B is an algorithm of worst-case running time$O(n^2)$ which reads an n-bit input string, does some computation, and outputs a n^2-bit string. Now combine A and B into a new algorithm $A \circ B$ by first running B on the input x and then A on the output of B. That is, compute $A(B(x))$. You might think of B as some kind of preprocessing step. S being closed under argument powering allows us to state "if the running time of A is in S then that of $A \circ B$ is, too."

Proof. 1) poly := $\bigcup_{c\geq 1} O(n^c)$ is closed under argument powering.

This is because for any $c_1, c_2 \geq 1$, $(n^{c_1})^{c_2} = n^{c_1 c_2} \in \bigcup_{c\geq 1} O(n^c)$.

2) quasipol:= $\bigcup_{c\geq 1} O(2^{\log^c(n)})$ is closed under argument powering.

This is because for any $c_1, c_2 \geq 1$, $2^{\log^{c_2}(n^{c_1})} = 2^{C_1^{C_2}\log^{c_2}(n)} \leq 2^{(\log(n))^{c_2^{c_1^{c_2}}}} \in \bigcup_{c\geq 1} O(n^c)$.

3) stronglySubExp:= $\bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$ is closed under argument powering.

Suppose $f \in \bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$. Then for any $c > 0$, for any arbitary $\varepsilon > 0$, there exist $c_1 > 0$ and n_0 such that for all $n \geq n_0$ we have $f \leq c_1 2^{n^{\varepsilon/c}}$. Then $f(n^c) \leq c_1 2^{(n^c)^{\varepsilon/c}} = c_1 2^{n^\varepsilon} \in O(2^{n^\varepsilon})$. $f(n^c) \in \bigcap_{\varepsilon>0} O(2^{n^\varepsilon})$.

4) weaklySubExp:= $\bigcap_{\varepsilon>0} O(2^{\varepsilon n})$ is not closed under multiplication.

Let $f = 2^{\sqrt{n}} \in \bigcap_{\varepsilon>0} O(2^{\varepsilon n})$. We have $f(n^2) = 2^n \notin \bigcap_{\varepsilon>0} O(2^{\varepsilon n})$.

5) E:= $\bigcup_{c\geq 1} O(2^{cn})$ is not closed under multiplication.

Let $f = 2^n \in \bigcup_{c\geq 1} O(2^{cn})$. We have $f(n^2) = 2^{n^2} \notin \bigcup_{c\geq 1} O(2^{cn})$.

6) Exp:= $\bigcup_{c\geq 1} O(2^{n^c})$ is closed under multiplication. This because for any $c_1, c_2 \geq 1$, $2^{(n^{c_1})^{c_2}} = 2^{n^{c_1 c_2}} \in \bigcup_{c\geq 1} O(2^{n^c})$. ■

6. Graph and Isomorphisms

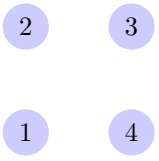
Problem 32. A minute of thought shows that there are $2^3 = 8$ graphs on the vertex set $V = a, b, c$. Many of them, however, are isomorphic. For example, all such graphs that have two edges are isomorphic. A closer look reveals that there are four isomorphism classes:

Now do the same for four vertices. Draw all non-isomorphic graphs on four vertices. To be more precise, there are $2^6 = 64$ graphs on the vertex set a, b, c, d, but only 11 isomorphism classes (oops, I gave it away). Anyway. Just draw one graph for each of these 11 isomorphism classes.

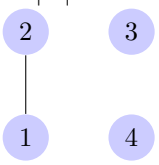
Solution. Since max degree for each vertex is 3, the max number of edges is $4 \times 3/2 = 6$.

Now we draw the 11 classes.

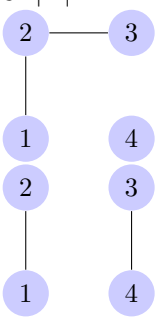
1: $|E| = 0$. There is only 1 class.



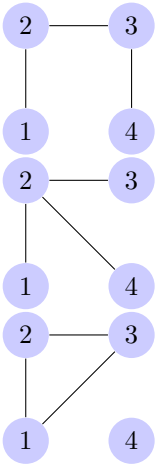
2: $|E| = 1$. There is only 1 class.



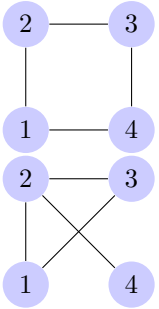
3: $|E| = 2$. There is only 2 class.



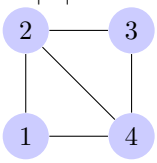
4: $|E| = 3$. There is only 3 class.



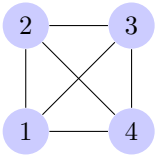
5: $|E| = 4$. There is 2 classes.



6: $|E| = 5$. There is only 1 class.



7:$|E| = 6$. There is only 1 class.



Problem 33. As I have stated above, there are 11 isomorphism classes of graphs on four vertices. How many are there on 50 vertices? It is probably very difficult to compute the exact number. However, the following task is definitely doable: Show that the number of isomorphism classes of graphs on 50 vertices is an even number!

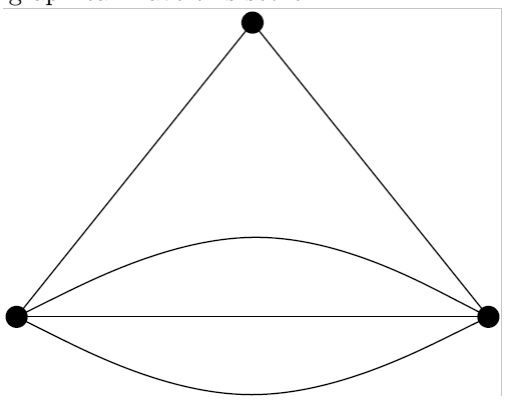
Proof. Take a arbitary graph on n unlabeled vertices V, there are maximum $n(n-1)/2$ edges. We note by $\mathcal{C}$ the set of all distinct unlabeled graphs on V. We can partition $\mathcal{C}$ into $n(n-1)/2+1$ parts, $\{C_0, C_1, \ldots, C_{n(n-1)/2}\}$ with $C_k = \{\text{isomorphism classes of graphs}|\text{contains } k \text{ edges}\}$.

Now we claim $\forall k \in \{0, 1, \ldots, n(n-1)/2\}$ there are same number of distinct unlabeled graphs in C_k and $C_{n(n-1)/2-k}$. This is shown by a complement graph argument. Formally, for any graph $G = (V, E)$, we define the complement of G to be $H = (V, K \backslash E)$, with K the set of all possible edges linking two vertices in V. We observe that for each $G \in C_k$, its complement H is in $C_{n(n-1)/2-k}$. So $|C_k| = |C_{n(n-1)/2-k}|$.

This means for all $k \in \{0, 1, \ldots, n(n-1)/2\}$, $|C_k \cup C_{n(n-1)/2-k}|$ is even.

For any graph on 50 vertices, the maximum number of edges is $n(n-1)/2 = 1225$. So there are 1226 elements in the partition $\mathcal{C}$. Then we can write $\mathcal{C} = \uplus_{k=0}^{612} \left(C_k \cup C_{n(n-1)/2-k}\right)$. Which leads to $|\mathcal{C}| = \sum_{k=0}^{612} |C_k \cup C_{n(n-1)/2-k}|$. Since the summants are all even, $|\mathcal{C}|$ is even. So we are done. ∎

The graph score theorem. This is a multigraph with score $(4, 4, 2)$. Obviously no graph can have this score.



Recall that in a multigraph we can have several parallel edges between two vertices. We cannot, however, have self-loops. That is, there cannot be an edge from u to u itself.

Problem 34. Find a graph score theorem for multigraphs. That is, a theorem of the form "there exists a multigraph with score $(d_1, ..., d_k)$ if and only if $(d_1, ..., d_k)$ satisfies P. Here, P should be some property that is easy to check.

Prove your theorem.

Solution. Score theorem for multigraph: a nondecresaing sequence $(d_1, ..., d_k)$ is a realisable score for a multigraph $G = \{V, E\} \iff$ 1) $\sum_{i=1}^{k} d_i$ is even and 2)$d_k \leq \sum_{i=1}^{k-1} d_i$.

Now we prove this score theorem.

The $\Longrightarrow$part.

Since $\sum_{i=1}^{k} d_i = 2|E|$, so $\sum_{i=1}^{k} d_i$ is even. Also, every edge incident to the vertex of largest degree has its other end counted among the degrees of the other vertices, so the inequality holds.

The $\Longleftarrow$ part.

Suppose (a) $\sum_{i=1}^{k} d_i$ is even and (b) $d_k \leq \sum_{i=1}^{k-1} d_i$, we claim we can construct a multigraph.

We do an induction on $\sum d_i = 2n$.

The base case $\sum d_i = 0$ is realized by an isolated vertices. For $\sum_{i=1}^{k} d_i > 0$, there are two cases.

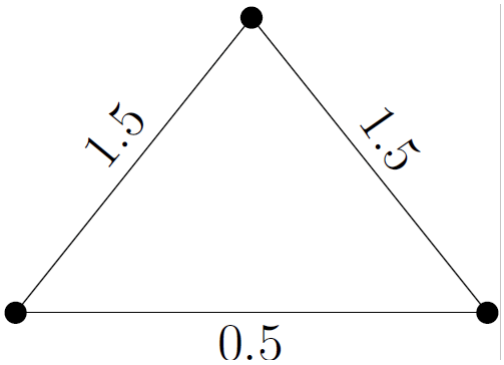
Case 1: $d_k = \sum_{i=1}^{k-1} d_i$. Then the multigraph realises $(d_1, ..., d_k)$ is with vertices $(v_1, v_2, \cdots, v_k)$ and for all $i = 1, \ldots, k-1$, there are d_i edges from v_i to v_k .

Case 2: $d_k < \sum_{i=1}^{k-1} d_i$.

The induction hypothesis is that for $\sum d_i = 2n$ and $d_k < \sum_{i=1}^{k-1} d_i$, there exist a multigraph $G = \{V, E\}$ with degree sequence $(d_1, ..., d_k)$.

Now suppose $\sum d_i = 2n + 2$ and $d_k < \sum_{i=1}^{k-1} d_i$. Since the total sum $\sum_{i=1}^{k} d_i$ is even, the difference between the two sides must be at least 2. Since d_k is the largest value, there exists at least be two nonzero degrees before d_k. Now subtract 1 from the two lowest nonzero degrees, say d_i and d_j, then by the induction hypothesis, there exists a multigraph $G = \{V, E\}$ associated with the resulting degree sequence$(d_1, \ldots, d_i - 1, \ldots, d_j - 1, \ldots, d_k)$. Now if we add an edge $\{v_i, v_j\}$ to the resulting multigraph $G = \{V, E\}$, the multigraph G' has $(d_1, \ldots, d_k)$ as degree sequence. So we are done.

Problem 35. A weighted graph is a graph in which every edge e has a nonnegative weight w_e. In such a graph the weighted degree of a vertex is $\text{wdeg}(u) = \sum_{v \neq u} w_{\{u,v\}}$.



This is an example of a weighted graph, which has score $(3, 2, 2)$. Obviously no graph and no multigraph can have this score.

Find a graph score theorem for weighted graphs and prove it!

Solution. Score theorem for weighted graph: a nondecresaing sequence $(d_1, ..., d_k)$ is a realisable score for a weighted graph $G = \{V, E\} \iff d_k \leq \sum_{i=1}^{k-1} d_i$.

The argument is almost the same as Score theorem for multigraph as multigraph can be seen as a special case of weighted graph for which the weightx can only be non-nagative integers. Now by relaxing the constraints on weights, we lose the even property of the sum of degrees. Now we prove this score theorem.

The $\Longrightarrow$part.

Also, every edge incident to the vertex of largest degree has its other end counted among the degrees of the other vertices, so the inequality holds.

The $\Longleftarrow$ part.

Suppose $d_k \leq \sum_{i=1}^{k-1} d_i$, we claim we can construct a weighted graph. However we could not do induction here since the set of sum of degrees is non-negative reals. Still, there are two cases.

Case 1: $d_k = \sum_{i=1}^{k-1} d_i$. Then the weighted graph realises $(d_1, ..., d_k)$ is with vertices $(v_1, v_2, \cdots, v_k)$ and for all $i = 1, \ldots, k-1$, there are d_i edges from v_i to v_k .

Case 2: $d_k < \sum_{i=1}^{k-1} d_i$.

We do a strong induction on k.

The base case is $k = 3$. (If $k = 2$, we must have $d_2 = d_1$, and if $k = 1$, we must have $d_1 = 0$). For $k = 3$, if $d_3 \geq d_2 \geq d_1$,we can set the weights to be $w_{\{1,2\}} = \frac{d_1+d_2-d_3}{2} \geq 0, w_{\{1,3\}} = \frac{d_1+d_3-d_2}{2} \geq 0, w_{\{2,3\}} = \frac{d_2+d_3-d_1}{2} \geq 0$ to construct the desired graph.

The induction hypothesis: for $k = n-1$,for non-decreasing degree sequence such that $d_{n-1} < \sum_{i=1}^{n-2} d_i$ we can construct a weighted graph.

We suppose that $k = n$ and $d_n < \sum_{i=1}^{n-1} d_i$. Then the sequence $(d_2, d_3, \ldots, d_{n-1}, d_n - d_1)$ admits either $d_n - d_1$ or d_{n-1} as its largest value.

If $d_n - d_1$ is the largest value and we suppose for now $d_n - d_1 \neq 0$, then we have $d_n - d_1 < \sum_{i=1}^{n-1} d_i - d_1 = \sum_{i=2}^{n-1} d_i$. And by the induction hypothesis, we can construct a weighted graph on $\{v_2, \ldots, v_n\}$ with $(d_2, d_3, \ldots, d_{n-1}, d_n - d_1)$ as its weighted degree sequence. Now just add vertex v_1 and connect v_1 and v_n with an edge of weight d_1 will construct a desired weighted graph on $\{v_1, \ldots, v_n\}$.

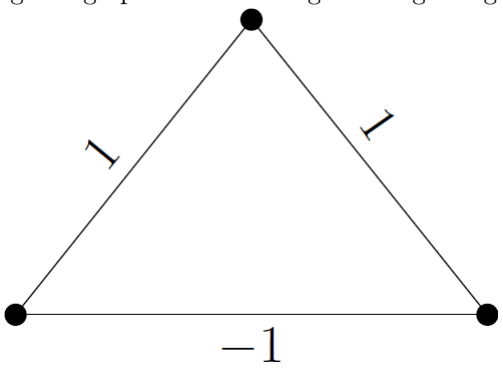
If $d_n - d_1 = 0$, then it must be the case $d_1 = d_2 = \cdots = d_n$. Consider the sequnce $(d_2, \ldots, d_{n-1})$, now also satisfies the induction hypothese for $k = n-2$. So

we have a graph for $(v_2, v_3, \ldots, v_{n-1})$, and we add two vertices v_1 and v_n with d_1 times edge connecting v_1 and v_n will give us a desired graph.

If d_{n-1} is the larest, then $d_{n-1} < d_n - d_1 + d_2 \leq d_n - d_1 + \sum_{i=2}^{n-2} d_i$. And follow the same reasoning as above, we can also construct a desired weighted graph on $\{v_1, \ldots, v_n\}$.

So we are done.

Problem 36. Suppose now we allow negative edge weights. Then we can get a graph of score $(2, 0, 0)$, which is of course impossible for graphs, multigraphs, and weighted graphs with non-negative edge weights.



Find a graph score theorem for weighted graphs where negative edge weights are allowed and prove it!

Solution. If we allow negatvie edge weights, then for $k \geq 3$ any non-decresasing sequence of real numbers is realisble. The case $k \leq 2$ is just trival.

The non-decresasing is just the defintion of degree sequence. And given any real sequnce, we can relable them to be non-decreasing.

For $k = 1$, only possilbilty is $d_1 = 0$.

For $k = 2$, only possilbilty is $d_1 = d_2$.

And for $k = 3$, suppose 3 reals such that $d_1 \leq d_2 \leq d_3$(possible after relableing) then we can set the weights to be $w_{\{1,2\}} = \frac{d_1+d_2-d_3}{2}, w_{\{1,3\}} = \frac{d_1+d_3-d_2}{2}, w_{\{2,3\}} = \frac{d_2+d_3-d_1}{2}$ to construct the desired graph.

The induction hypothesis: it is possible for $k = n-1$ to construct a graph with any real degree sequence $d_1 \leq d_2 \leq \ldots \leq d_{n-1}$.

Now for the case $k = n$. Given a sequence of non-decreasing sequence. Then consider the sequnce $(d_2, d_3, \ldots, d_n - d_1)$.

Two cases :

Case 1: $d_n - d_1 \neq 0$. Then with possible relabling, by inducton hypothese, we have a graph on vertices $(v_2, \cdots, v_n)$ and we connect v_1 and v_n with an edge with weight d_1 will give us the required graph.

Case 2: $d_n = d_1$. Then $d_1 = d_2 = \cdots = d_n$. One possible graph is given by the circle with $w_{\{i,i+1\}} = \frac{d_1}{2}$ for $1 \leq i \leq n-1$ and $w_{\{1,n\}} = \frac{d_1}{2}$.

7. Cycles and Trees

Notations: For a graph $G = (V, E)$, we note $V(G) = V$ its vertices and $E(G) = E$ its edges.

For $u, v \in V(G)$, we define the **distance** between u, v, $d(u, v)$ as the least lengt of a $u, v-$path. If such path does not exists, then $d(u, v) = \infty$.

Problem 37. Let T be a tree and $e \in \binom{V}{2}$ be an edge not in T. Show that $T + e$ contains a unique cycle, and if e' is on that cycle then $T + e - e'$ is again a tree.

Proof. We firstly prove a lemma that would be useful.

Lemma(*): If there are two paths connecting vertices u and v in a graph G, then there exist a cycle in G.

Proof of Lemma(*): This is because that the two paths must diverge at some vertex $w \in V(G)$ and re-converge at some another vertex $w' \in V(G)$, otherwise the 2 paths are the same. Thus we can construct a cycle in G starting from w and following the first path to w' and come back to w following the second path.

Now we can prove the first part the problem:

$T + e$ **contains a unique cycle.**

Let $e = \{u, v\} \in \binom{V}{2} \backslash E(T)$.

Existence of a cycle in $T + e$:

Since T is a tree, T is connected. There is a path $P_1 = u \to x_1 \to \ldots \to x_n \to v$ for some n and each edges is in $E(T)$.

Now, e is not in $E(T)$, e constitue another $u - v$ path. Connecting the two paths, we can create a cycle $C_1 = \{u \to x_1 \to \ldots \to x_n \to v \to u\}$ in $T + e$.

Uniqueness of the cycle in $T + e$:

Suppose there exists another cycle C_2 in $T + e$. Two cases:

Case I: $u, v \in V(C_1) \cap V(C_2)$. Then there exists another path $P_2 \neq P_1$ from u to v in $C_2 \backslash e$. But $C_2 \backslash e$ is in T. So with P_1 and P_2 both in T, **Lemma(*)** implies that there exist a cycle in T. Contradiction.

Case II: At least one of u, v is not in $V(C_2)$. This is enough to ensure that e is not within C_2, which means C_2 is within T. But such C_2 could not exisit as T is acyclic. Another contradiction.

Now we prove the second part of the problem:

if e' **is on that cycle then** $T + e - e'$ **is again a tree.**

Claim1: $T + e - e'$ is connected. This is because the removal of any edge in a cycle within a graph does not break the connectivity of that graph.

Claim2: $|E(T + e - e')| = |V(T + e - e')| - 1$. This is because $|E(T)| = |V(T)| - 1$, $|E(T + e - e')| = |E(T)|$ and $|V(T)| = |V(T + e - e')|$.

Then combining **Claim 1** and **Claim 2**, $T + e - e'$ is again a tree.

■

Problem 38. A graph $G = (V, E)$ is called *bipartite* if there is a partition $V = V_1 \uplus V_2$ such that no edge of G is within V_1 or V_2. That is, for every edge $e = \{u, v\} \in E$ we have $|e \cap V_1| = |e \cap V_2| = 1$. Prove that G is bipartite if and only if it has no cycle of odd length.

Proof. **The $\Longrightarrow$ part:**

Lemma 1: if G is bipartite, then for any $u \in V_1$, any path of odd length starting from u will end in V_2 and any path of even length will end in V_1.

Proof of lemma 1: We do this by induction.

The base case:

Path of length 1 is of the form $u \to v$. Now since no other vertex in V_1 is adjacent to $u \in V_1$, v must be in V_2.

Path of length 2 is of the form $u \to v \to u'$. Since v is adjacent to u, $v \in V_2$. Since there are no other vertex in V_2 adjacent to v, u' must be in V_1.

Induction Hypo(IH):

Path of length $2n$ with one end in V_1 must have another end also in V_1.

Path of length $2n+1$ with one end in V_1 must have another end also in V_2.

Prove the case $n+1$ works:

Path of length $2(n+1)$. If we number the vertices in the path as

$$u_1, u_2, \ldots, u_{2n+1}, u_{2(n+1)},$$

then by **IH** we have u_{2n+1} is in V_2. And since $u_{2(n+1)}$ is adjacent to u_{2n+1}, it is in V_1.

By the same argument, the path of length $2(n+1)+1$ is with one end in V_1 and another end in V_2.

By the same argumen,we have

Lemma 1bis: if G is bipartite, then for any $u \in V_2$, any path of odd length starting from u will end in V_1 and any path of even length will end in V_2.

Since any path is either of odd length or of even length, **Lemma 1** and **Lemma 1bis** implies

Lemma2 : If G is bipartite, any path with both ends in V_1 or V_2 is of even length and any path with one end in V_1 and another end in V_2 is of odd length.

Now, a cycle C is a closed path from any $u \in C$ to itself, so any cycle must a path with both ends in V_1 or V_2. If G is bipartite, by **Lemma 2**, C is of even length .

The $\Longleftarrow$ part: Suppose G has no cycle of odd length. Then we can partition $V(G)$ into two subsets $V_1 \uplus V_2$ such that no edge of G is within V_1 or V_2.

We firstly prove it is feasible if G is connected.

Take any $u \in G$, we set $V_1 = \{v | d(u,v) \text{ is odd}\}$ and $V_2 = \{v | d(u,v) \text{ is even}\}$. (0 is even here). Clearly V_1 and V_2 is a partition of $V(G)$. Note here V_1 and V_2 are conditional on the choice of u.

We claim that **if G dose not contain cycle of odd length, then no edge in G is within V_1 or V_2 associated with any u.**

Choose $x, y \in V_1$ and suppose $e = \{x, y\} \in E(G)$. By the construction of V_1, $d(u, x)$ and $d(u, y)$ are odd.

Claim: The path realizing $d(u, x)$ and $d(u, y)$ could not contain any same vertex.

Why? Suppose there exisits u' as the last common vertex on the path of $d(u, x)$ and $d(u, y)$, then $d(u', x)$ and $d(u', y)$ are both even or odd since $d(u'x) = d(u, x) - d(u, u')$ and $d(u'x) = d(u, x) - d(u, u')$. By connecting the path realizing $d(u', x)$, the edge e and the path realizing $d(u', y)$, we can create a cycle of odd length. Contradiction.

But the **Claim** is impposible at least u is on the two paths.

Therefore, there can be no edge connecting two vertices in V_1. Same argument apply to V_2.

So we have proven no edge in G is within V_1 or V_2 associated with any u.

So G is bipartite for the connected case.

Now we prove the general case. G can be partitioned into countable many connected components G_i such that $V(G_i)$ and $V(G_j)$ is not link by any path for $i \neq j$. Then for each G_i we have a partition $V(G_i) = V_1(G_i) \uplus V_2(G_i)$ with no edge of G_i is within $V_1(G_i)$ or $V_2(G_i)$. We claim that

$$V = [\uplus V_1(G_i)] \uplus [\uplus V_2(G_i)]$$

with no edge e of G is within $\uplus V_1(G_i)$ or $\uplus V_2(G_i)$. This is because if such e exists, it must be in some G_i. But by the construction of $V_1(G_i)$ and $V_2(G_i)$, e could not be within $V_1(G_i)$ or $V_2(G_i)$ which leads to a contradiction.

■

Problem 39. Consider the *Hamming cube* H_n, which is the graph on vertex set $\{0, 1\}^n$ where two vertices are connected by an edge if they differ in exactly one coordinate.

(1) Is H_n bipartite? If so, define a partition $\{0, 1\}^n = V_1 \uplus V_2$ that certifies this; if not, give an odd cycle!
(2) For which values of k does H_n contain a cycle of length k?

Solution. We note $x \in H_n$ by its binary representation $(x_1, x_2, \cdots, x_n)$ with $x_i = 0$ or 1 for $i \in [n]$.

1. H_n is bipartite. $V_1 = \{x = (x_1, x_2, \cdots, x_n) | \sum x_i \text{ is odd}\}$ and $V_2 = V_1 = \{x = (x_1, x_2, \cdots, x_n) | \sum x_i \text{ is even}\}$. Clearly $H_n = V_1 \uplus V_2$.

Now choose $x, x' \in V_1$ and suppose $e = \{x, x'\} \in E(H_n)$. Then x and x' only differ in exactly one coordinate. But this means that $\sum x_i$ and $\sum x'_i$ could not be both odd. So there can be no edge connecting the two vertices in V_1. Same argument applies to V_2. So H_n is bipartite.

2. H_n contains all the cycles of length $4, 6, ..., 2^n$.

Cycle of length 2 is not possible since a cycle is path from some vertex u back to itself u, and no edge and vertex are repeated except for u.

In general, the length of any cycle must be larger than 3.

H_n could not contain odd cycles as H_n is bipartite. The length of cycles in H_n is bounded from above by the $|H_n|$. So the possible length of cycles for H_n is a subset of $\{4, 6, ..., 2^n\}$.

We define the **Hamming distance** $d_h(u, v)$ for $u, v \in H_n$ as the numbers of coordinates that differ between u and v.

Lemma: Choose $u, v \in H_n$, the distance between u, v, $d(u, v)$ is given by the **Hamming distance**.

This **Lemma** is rather self explanatory. Why? The two ends of any edge in H_n can only differ in one coordinates, so the **Hamming distance** is the lower bound of the $d(u, v)$. And the **Hamming distance** is achievable since we can construct a path from u to v with changing one coordinate a time at each step.

Now we have fully characterized the distance in H_n. We are ready to prove that H_n contains all the cycles of length $l \in \{4, 6, ..., 2^n\}$. It suffices to find one cycle in H_n of length l.

This could be done by exploiting the charateristic of **partial Gray code**. We first define the **Gray code**, and then we define the **partial Gray code**.

The n**-bit Gray code** G_n could be defined recursively as follows:

$G_1 = \{0, 1\}$.

Define $G_1^R = \{1, 0\}$ be the G_1 of reverse order.

Define $0G_1 = \{00, 01\}$ by pre-pending 0 to each element in G_1 and define $1G_1^R = \{11, 10\}$ by pre-pending 1 to each element of G_1^R.

Then $G_2 = \{0G_1, 1G_1^R\} = \{00, 01, 11, 10\}$.

In general, we have $G_n = \{0G_{n-1}, 1G_{n-1}^R\}$.

We have few observations. Hope they are rather evident. (I am too tired to give a formal proof.)

Observation 1: G_n is the set of all the vertices in a **Hamming cube** of dimension n. So $|G_n| = 2^n$.

Observation 2: Each pair of adjacent elements in G_n differ in only one of the n coordinates. Thus, G_n represents a path in H_n from $u = (\underbrace{0 \cdots 0}_{\text{n zeros}})$ to $v = (1\underbrace{0 \cdots 0}_{\text{n-1 zeros}})$.

Let $2 \leq m \leq 2^{n-1}$. The $(n-1)-$**bit partial Gray code of oreder** m is defined as the sequence consisting of the first m elements of **Gray code** G_{n-1}. And define $G_{n-1}(m)^R$ to be the sequence $G_{n-1}(m)$ with reverse order. Then we can build a cylce of length $2m$ with the help of $(n-1)-$**bit partial Gray code of oreder** m. This is summarized in **Observation 3**.

Observation 3: $\{0G_{n-1}(m), 1G_{n-1}(m)^R, (\underbrace{0 \cdots 0}_{\text{n zeros}})\}$ is a cycle in H_n of length $2m$.

For example: $G_2(3) = \{00, 01, 11\}$, $G_2(3)^R = \{11, 01, 00\}$ and

$$\{0G_2(3), 1G_2(3)^R, (\underbrace{0\cdots 0}_{\text{n zeros}})\}$$
$$=\{000, 001, 011, 111, 101, 100, 000\}$$

is a cycle of length 6 in H_3.

Note since the **Observation 3** applies to $m = 2$ to 2^{n-1}, we can construct at least one cycle in H_n of length $l \in \{4, 6, ..., 2^n\}$.

So we are done.

Spanning tree exchange graph.

Problem 40. For a graph $G = (V, E)$ let us define the spanning tree exchange graph $\mathcal{T}(G)$. The vertex set of $\mathcal{T}(G)$ are the spanning trees of G , and two trees T, T' are connected by an edge in $\mathcal{T}(G)$ if they differ in one edge only, i.e., $|E(T) \setminus E(T')| = 1$(equivalently, $|E(T') \setminus E(T)| = 1$). Show that $\mathcal{T}(G)$ is connected for every graph G .

Proof. The connectivity of $\mathcal{T}(G)$ is based on the following **Exchange Property** (which is a direct result of Problem 1 of Assignment 7-1 and the fact that $V(T) = V(G)$):

(**Exchange Property** of Spanning tree) Let T be a spanning tree of G. Consider any edge e not in the tree in $\binom{V(T)}{2} \setminus E(T)$. We note the two endpoint of edge e to be u and v. Since T is connected, there is a unique $u, v-$path P in T . Consider any edge f from T which belongs to the path P, we have $T' = T + e–f$ is again a spanning tree.

We note that if $|\mathcal{T}(G)| \leq 1$, then $\mathcal{T}(G)$ is trivially connected. So we will suppose that $|\mathcal{T}(G)| \geq 2$.

The exchange property guarantees that $\mathcal{T}(G)$ to be connected. It is enough to show that any two spanning trees of G can be obtained from each other by a finite sequence of edge exchanges. Formally, let $T, T' \in \mathcal{T}(G)$, set $U = E(T) \setminus E(T')$ and $V = E(T') \setminus E(T)$. It is obvious that U and V have same numbers of element, i.e., $|U| = |V|$. Suppose that $|U| = |V| = n$, we can order the element of V as $\{v_1, \cdots, v_n\}$.

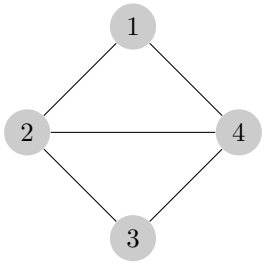
Now consider the graph $T + v_1$. Since T is a tree, $T + v_1$ contains a cycle C_1. As T' is a tree, at least one edge on C_1 is not in T'. It is clear this edge is in U. Note this edge of T by u_1. Define $T_1 = T + v_1 - u_1$, then by the Exchange Property, T_1 is still a spanning tree of G. Now T_1 and T' have one more edge in common. And by the definition of $\mathcal{T}(G)$, T_1 and T' are adjacent.

Repeating the process $n - 1$ more times, we get a sequence of spanning tress $T_0 = T, T_1, \cdots, T_{n-1}, T_n = T'$ such that T_i and T_{i+1} are adjacent in $\mathcal{T}(G)$ for $0 \leq i \leq n - 1$.

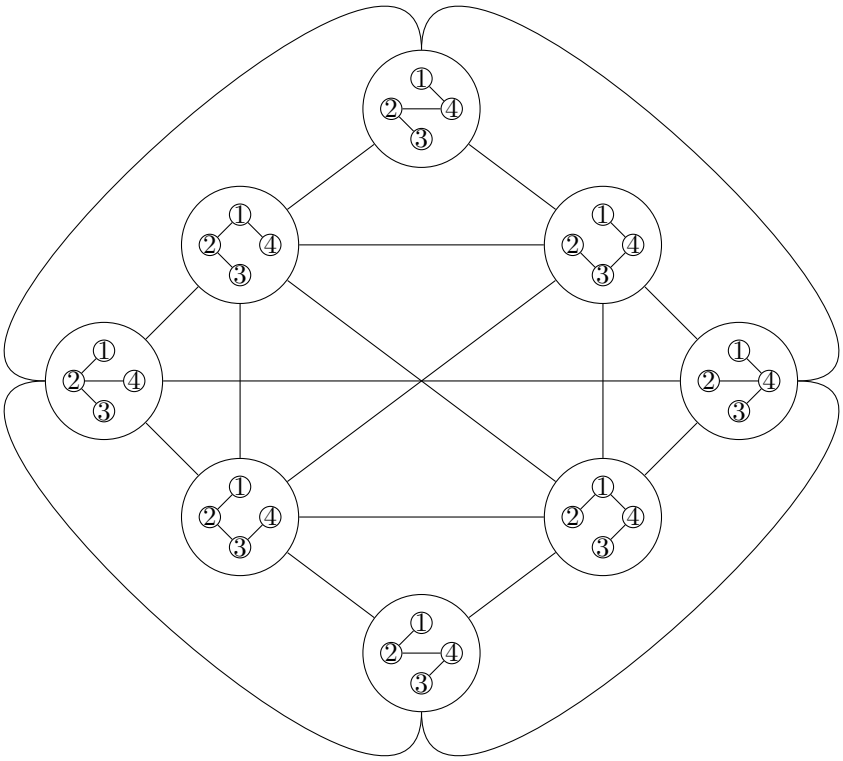
So we have constructed a path from T to T'. (It is path since each T_is are different by constructions)

Since the above statement is true for any two spanning trees $T, T' \in \mathcal{T}(G)$, $\mathcal{T}(G)$ is connected. ∎

Problem 41. For the graph G below, draw $\mathcal{T}(G)$ nicely, where the notion of "nicely" is up to you.



Solution. There are 8 nodes in the graph $\mathcal{T}(G)$.



Problem 42. True or false: if G is connected and has a cycle, then $\mathcal{T}(G)$ has a cycle?

Solution. True.

Let $C = \{e_0, e_1, \cdots, e_n\}$ be a length n cycle in G. Suppose for now this is the unique cycle in G.

We note$T_0 = (V(G), E(G)\backslash e_0)$. We claim T_0 is a spanning tree. Why? 1) T is acyclic since we cut the cycle C 2)T is connected since remove one edge in the cycle doesn't break the connectivity.

Now we set $T_i = T_{i-1} + e_{i-1} - e_i$ for $1 \leq i \leq n$. Clearly, $T_0, T_1, \ldots, T_n$ are all spanning trees and by the **Exchange Property** the sequence is a path in $\mathcal{T}(G)$ connecting T_0 and T_n. But $T_0 = T_n + e_n - e_0$. So T_0 and T_n are adjacent. Thus the sequence$T_0, T_1, \ldots, T_n, T_0$ is a cycle in $\mathcal{T}(G)$.

Now if G contains $n \geq 2$ cycles, then we can always remove some edges in G such that the remaining graph G' satisfies :1) G' is connected and 2) G' contains a unique cycle. Then there is a cycle in $\mathcal{T}(G')$. Since $\mathcal{T}(G')$ is a sub-graph of $\mathcal{T}(G)$, $\mathcal{T}(G)$ also contains a cycle. So we are done.

Problem 43. Show that if G and G' are isomorphic then so are $\mathcal{T}(G)$ and $\mathcal{T}(G')$.

Proof. True.

Let f be an isomorph from G to G'.

We need to prove the existence of a bijection $g : \mathcal{T}(G) \to \mathcal{T}(G')$ such that any $T_1, T_2 \in \mathcal{T}(G)$ are adjacent in $\mathcal{T}(G)$ if and only if $g(T_1)$ and $g(T_2)$ are adjacent in $\mathcal{T}(G')$.

A natural choice for g is the that for any $T \in \mathcal{T}(G)$ we define $g(T) = (V(G'), f(E(T)))$ where $f(E(T)) = \bigcup_{\{u,v\} \in E(T)} \{f(u), f(v)\}$. It is clearly that $g(T) \in \mathcal{T}(G')$.

It remains to prove that g is 1) bijective and 2) any $T_1, T_2 \in \mathcal{T}(G)$ are adjacent in $\mathcal{T}(G)$ if and only if $g(T_1)$ and $g(T_2)$ are adjacent in $\mathcal{T}(G')$.

1) The bijective of g is obvious as f is an isomorph from G to G'.

Injective:

Suppose for some spanning trees $T_1 \neq T_2$ we have $g(T_1) = g(T_2)$. Since we T_1, T_2 are spanning trees, it must be the case that $E(T) \neq E(T_1)$. Now since $g(T_1) = g(T_2)$, then there must be at least two distinct pairs of vertices of G such that $\{u, v\} \in E(T_1), \{u', v'\} \in E(T_2)$, $\{u, v\} \neq \{u', v'\}$ and $\{f(u), f(v)\} = \{f(u'), f(v')\}$. But since f is isomorph, no such pairs of vertices of G could exist. So we have proved that $g(T_1) \neq g(T_2)$.

Surjective:

For any $S \in \mathcal{T}(G')$, we can define $T = (V(G), \bigcup_{\{u,v\} \in E(T)} \{f^{-1}(u), f^{-1}(v)\}$ and have $g(T) = S$.

2) Suppose $T_1, T_2 \in \mathcal{T}(G)$ are adjacent, which means that there exist two edges $x, y \in \binom{V(G)}{2}, x \neq y$ such that $E(T_1)\backslash E(T_2) = x$ and $E(T_2)\backslash E(T_1) = y$.

Then it remains to show that there are $x', y' \in \binom{V(G')}{2}, x' \neq y'$, such that $E(g(T_1))\backslash E(g(T_2)) = x'$ and $E(g(T_2))\backslash E(g(T_1)) = y'$.

It suffice to let $x' = f(x)$ and $y' = f(y)$.

The other way round is also obvious by using the reverse map f^{-1}. ∎

Problem 44. "If $\mathcal{T}(G)$ and $\mathcal{T}(G')$ are isomorphic then G and G' are isomorphic." Show that this statement is false.

Proof. Suppose G and G' are two non-isomorphic trees. Then both $\mathcal{T}(G)$ and $\mathcal{T}(G')$ have only one vertex, i.e., G and G' respectfully. Clearly, $\mathcal{T}(G)$ and $\mathcal{T}(G')$ are isomorphic. But G and G' aren't isomorphic by assumption. ■

Problem 45. Suppose $\mathcal{T}(G)$ and $\mathcal{T}(G')$ are isomorphic. What does this say about G and G' ? Are they "similar" in some way?

Solution. A necessary condition for $\mathcal{T}(G)$ and $\mathcal{T}(G')$ to be isomorphic is that 1) G and G' are both connected 2)G and G' contains same number of cycles . This is just an observation and I have no idea of how to prove this.

But I can prove a weaker condition:

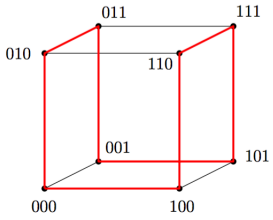
Claim: Suppose G is a connected graph containing only one $n-$circle C_n and G' another connected graph containing only one $n-$circle C'_n. Then $\mathcal{T}(G) \simeq \mathcal{T}(G')$.

Just use the **Edge Exchange Property**. Any ST(spanning tree) of G must contain $n-1$ edge of the C_n. Then by an edge exchange, each ST must be adjacent in $\mathcal{T}(G)$ to another ST of G. This means that $\mathcal{T}(G)$ is K_n a complete graph of degree n such that each vertex is adjacent to $n-1$ other vertices. Since the reasoning applies to any graph G containing only one C_n, we have also $G' \simeq K_n$. Thus $G \simeq G'$.

But for graphs with more then one circles, the structure is much more complex. I don't know how to prove it. So I just leave it here and will come back later at the end of the course.

8. Hamiltonial Cycles and Paths

Problem 46. We consider the Hamming cube, which is a graph on2^n vertices, defined as follows. Its vertex set is$0,1^n$, the set of binary strings of length n. Two vertices u, v are connected if they differ in exactly one component. If you imagine $0,1^n$ as a (finite) subset of the space $\mathbb{R}^n$, then u, v are connected by an edge if and only if their Euclidean distance is 1. We denote this graph by H^n. Here is H_3 together with a Hamilton cycle: Show that for $n \geq 2$ the graph H_n has a Hamiltonian Cycle.



Show that for $n \geq 2$ the graph H_n has a Hamiltonian Cycle.

Proof. This is the same as Problem 2 in the Assignment 1 of Week 7. The Hamiltonian Cycle in H_n is linked to the n-bit Gray Code.

We note $x \in H_n$ by its binary representation $x_1x_2\cdots x_n$ with $x_i = 0$ or 1 for $i \in [n]$.

The **n-bit Gray code** G_n could be defined recursively as follows:

1-bit Gray code: $G_1 = \{0, 1\}$.

Define $G_1^R = \{1, 0\}$ be the G_1 of reverse order.

Define $0G_1 = \{00, 01\}$ by pre-pending 0 to each element in G_1 and define $1G_1^R = \{11, 10\}$ by pre-pending 1 to each element of G_1^R.

Then **2-bit Gray code is:** $G_2 = \{0G_1, 1G_1^R\} = \{00, 01, 11, 10\}$.

In general, n**-bit Gray code** G_n is the sequence of

$$G_n = \{0G_{n-1}, 1G_{n-1}^R\}.$$

Claim 1: By construction, G_n is contains all the vertices of H_n.

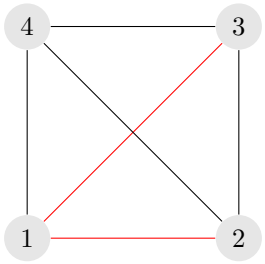
Claim 2: Each pairs of consecutive elements in G_n are adjacent in H_n, i.e., they differ in only one coordinate.

Claim 3: The first element of G_n is $\underbrace{0\cdots0}_{n \text{ zeros}}$ and the last element is $1\underbrace{0\cdots0}_{n\text{-1 zeros}}$, they are also adjacent.

So $\{0G_{n-1}, 1G_{n-1}^R, \underbrace{0\cdots0}_{n \text{ zeros}}\}$ is a Hamiltonian Cycle in H_n. And H_n is Hamiltonian.

■

Problem 47. Let $f(G)$ be the minimum number of edges you have to remove from G to destroy every Hamiltonian cycle. For example, $f(C_n) = 1$ since C_n has a Hamiltonian cycle (it is itself a cycle) and destroying one edge makes it non-Hamiltonian. As another example, $f(K_4) = 2$, as illustrated in this picture:



Find a nice formula for $f(K_n)$ and prove it.

Solution. We claim that $f(K_n) = n - 2$ for $n \geq 3$.

Proof:

For $n \geq 3$, the order n complete graph K_n is with vertices $v_1, v_2, \ldots, v_n$, and each vertex is adjacent to all the other vertices. Specially, v_1 is adjacent to $v_2, \ldots, v_n$.

Part 1: $f(K_n) \leq n - 2$.

Note $S = \{v_1v_k\text{-edges for } k = 3, 4, \ldots, n\}$. Then $|S| = n - 2$.

We claim that $K_n \backslash S$ can not be Hamiltonian.

Why? Suppose $K_n \backslash S$ is Hamiltonian, then there will be a Hamiltonian cycle that starts from v_1 and ends in v_1. But in $K_n \backslash S$, v_1 is only adjacent to v_2. Any Hamiltonian cycle that starts from v_1 and ends in v_1 must visit v_2 twice. But this is not possible since a cycle can never visit any vertex other then the origin twice.

So the minimum number of edge cuts to make K_n non-Hamiltonian is at least $n - 2$. That is $f(K_n) \leq n - 2$.

Part 2: $f(K_n) \geq n - 3$.

Choose any $n - 3$ edges in K_n and remove them. Note the remaining graph $V = K_n - U$ where U is the set of $n - 3$ chosen edges.

Then for any two vertices in V we have $\deg v_i + \deg v_j \geq 2\times(n-1)-(n-3) = n+1$. Since for the degree of any vertex is at most $n - 1$, this implies that the minimum of degree of all vertex $\delta(V) \geq 2$.

Then V is connected. It must contains a Hamiltonian path (path of length n-1).

After relabeling, we note this path $v_1v_2 \ldots v_{n-1}v_n$. Two case here:

Case 1: If v_1 and v_n are adjacent, the we have found a Hamiltonian cycle.

Case 2: Now we suppose v_1 and v_n are not adjacent. Since $\deg v_1 + \deg v_n \geq n+1$, there must be $t \in \{2, \ldots, n-2\}$ such that $v_1 v_{t+1}, v_n v_t \in E(V)$. Then the we have a Hamiltonian cycle $v_1 v_{t+1} v_{t+2} \ldots v_n v_t v_{t-1} v_2 v_1$.

Why such t exists? Note $X = \{t | (v_1 v_{t+1}) \in E(V)\}$, $Y = \{t | (v_t v_n) \in E(V)\}$. Then on one hand, $|X \cup Y| + |X \cap Y| = |X| + |Y| = \deg v_1 + \deg v_n \geq n+1$. On the other hand $|X \cup Y| \leq n-1$. So $|X \cap Y| \geq 2$.

Hence there is a Hamiltonian cycle in V. Since we choose these $n-3$ edges arbitrarily, we have $f(K_n) \geq n-3$.

Combining **2 Parts**, we have $f(K_n) = n-2$.

Problem 48. In the lecture we proved **Ore's Theorem**:

Let G be a graph on $n \geq 3$ vertices. If $\deg(u) + \deg(v) \geq n$ for all pairs of non-adjacent vertices u, v, then G has a Hamilton cycle.

Prove **Dirac's Theorem**:

if G is a graph on $n \geq 3$ vertices in which every vertex has degree at least $n/2$, then G has a Hamiltonian cycle.

Proof. This could prove the same way as the **Part 2** of Problem 2.

Let G to be a graph on $n \geq 3$ vertices in which every vertex has degree at least $n/2$.

Then for any two vertices in V we have $\deg v_i + \deg v_j \geq 2 \times (n/2) = n$. Since for the degree of any vertex is at most $n-1$, this implies that the minimum of degree of all vertex $\delta(G) \geq 1$.

Then G is connected. It must contains a Hamiltonian path (path of length n-1).

After relabeling, we note this path $v_1 v_2 \ldots v_{n-1} v_n$. Two case here:

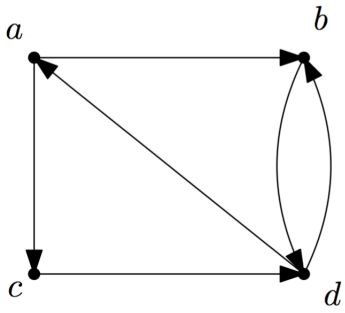
Case 1: If v_1 and v_n are adjacent, the we have found a Hamiltonian cycle.

Case 2: Now we suppose v_1 and v_n are not adjacent. Since $\deg v_1 + \deg v_n \geq n$, there must be $t \in \{2, \ldots, n-2\}$ such that $v_1 v_{t+1}, v_n v_t \in E(G)$. Then the we have a Hamiltonian cycle $v_1 v_{t+1} v_{t+2} \ldots v_n v_t v_{t-1} v_2 v_1$.

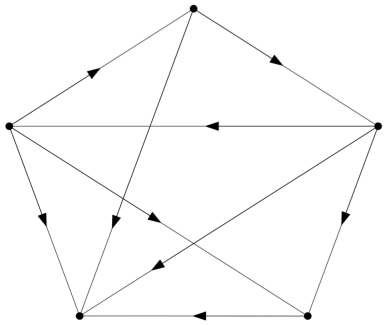
Why such t exists? Note $X = \{t | (v_1 v_{t+1}) \in E(G)\}$, $Y = \{t | (v_t v_n) \in E(G)\}$. Then on one hand, $|X \cup Y| + |X \cap Y| = |X| + |Y| = \deg v_1 + \deg v_n \geq n$. On the other hand $|X \cup Y| \leq n-1$. So $|X \cap Y| \geq 1$.

Hence there is a Hamiltonian cycle in G. ∎

Problem 49. A directed graph is a pair$G = (V, E)$ where V is a finite set of vertices and $E \subseteq V \times V$ is the set of edges. An element $(u, v) \in E$ is considered an edge from u to v. We usually assume that G has no self-loops, i.e., $(u, u) \notin E$ for all $u \in V$.

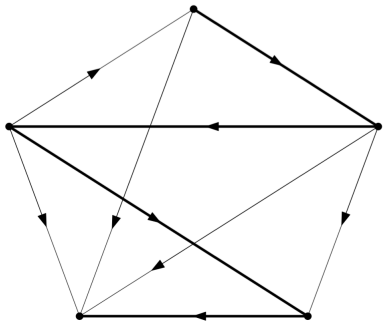


Note that the above directed graph has no edge between b and c and two edges between b and d. A directed graph $G = (V, E)$ is called a tournament if for allu, v in V, exactly one of the edges (u, v) and (v, u) is contained in E (see figure below). Imagine each vertex is a football team, and all possible matches have been played, without the possibility of a draw. Then $(u, v) \in E$ could mean "u defeated v" in the football tournament. That is where the name tournament for such graphs comes from. Here is an example of a tournament on five vertices:



Here is a surprising lemma:

[**Lemma**] Let G be a directed graph on n vertices. If G is a tournament, then it contains a directed Hamiltonian path. That is, there are vertices $v_1, \ldots, v_n$ such that $(v_i, v_{i+1}) \in E$.



Prove the lemma!

Proof. This lemma could be proved by an induction.

Base case : $n = 3$. With 3 vertiecs, there are 8 possible tournemants. And each of the 8 tournements admits a directed Hamiltonian path.

Inudction Hypothese(IH): the tournements of order n admits a directed Hamiltonian path.

Now let G to be a tournements graph of order $n+1$. If we don't consider the vertex v_{n+1}, by the IH, there is a directed path visiting each of $v_1, v_2, \ldots, v_n$ only once.

With relabeling, we note the path $v_1 \to v_2 \to \ldots \to v_n$. Now we consider the directed edge between v_{n+1} and v_1 and the directed edge between v_{n+1} and v_n.

Now three case:

Case 1: $(v_n v_{n+1}) \in E$, then $v_1 \to v_2 \to \ldots \to v_n \to v_{n+1}$ is the desired Hamiltonian path.

Case 2:$(v_{n+1} v_1) \in E$, then $v_{n+1} \to v_1 \to v_2 \to \ldots \to v_n$ is the desired Hamiltonian path.

Case 3: $(v_1 v_{n+1}), (v_{n+1} v_n) \in E$. We claim there is $s \in 1, 2, \ldots, n-1$ such that $(v_s v_{n+1}), (v_{n+1} v_{s+1}) \in E$. Then $v_1 \to \ldots \to v_s \to v_{n+1} \to v_{s+1} \to \ldots \to v_n$ is the desired Hamiltonian path.

Why such s exist? We note $X = \{s | (v_s v_{n+1}) \in E\}$ and $Y = \{s | (v_{n+1} v_{s+1}) \in E\}$ then $|X \cup Y| = n-1$, and $|X| + |Y| = n$. this means that $|X \cap Y| = |X| + |Y| - |X \cup Y| = 1$. So such s exsits.

Thus, G contains a Hamiltonian path. ■

Problem 50. Euler's Theorem gives us a simple necessary and sufficient criterion for the existence of an Euler cycle. Ore's Theorem gives us a sufficient criterion for the existence of Hamilton cycle. There most likely isn't a simple sufficient and necessary criterion for the existence of a Hamilton cycle.

However, try your best and find other sufficient criteria, beyond Ore's Theorem!

Proof. I could not find any on my own.

1) One direction of thought is inspired by Problem 4. Tha is if we replace the undirected edge with two directed edges of opposite direction, then if we can find a directed Hamilton with two endpoint adjacent, G would be Hamilton. But here G is usually not a tournament graph, so we could not use the lemme.

2) By a search on internet, there is a "Rahman-Kaykobad" condition on the existence of Hamiltonian path.

Let G be a connected graph with vertices such that for all pairs of distinct nonadjacent vertices $u, v \in V$ one has $d_u + d_v + \delta(u, v) \geq n + 1$. Then, G has a Hamiltonian path.

If the two endpoints of the Hamiltonian Path are adjacent, then G is Hamiltonian. ■

9. Minimum Spanning Trees

Problem 51. Throughout this assignment, let $G = (V, E)$ be a connected graph and $w : E \to \mathbb{R}^+$ be a weight function. A set $X \subseteq E$ is called **good** if there exists a minimum spanning tree T of G such that $X \subseteq E(T)$.

1. State the Cut Lemma from last class and sketch its proof. Draw a picture!

2. Prove the inverse of the cut lemma: If X is good, $e \notin X$, and $X \cup e$ is good, then there is a cut $S, V \setminus S$ such that

(i) no edge from X crosses this cut and

(ii) e is a minimum weight edge of G crossing this cut.

Proof. A set $X \subseteq E$ is called good if there is a MST(minimum spanning tree) T such that $X \subseteq E(T)$.

[**Cut Lemma**] Let $X \subseteq E$ be a good set. Let $V = S \cup \bar{S}$ be a cut of X. That is, no edge of X goes form S to $\bar{S}$. Let $e \in E$ be an edge of minimum cost crossing S and $\bar{S}$. Then $X \cup \{e\}$ is good too.

[**Proof of the Cut Lemma**] We can only consider the case such that (V, X) consits of only two connected components. If (V, X) consists of more than two connected components, then we can consider the sub-graph consisting only two connected components.

Let T be a MST such that $X \subseteq E(T)$. There are two case:

Case 1: $e \in T$. Then $X \cup \{e\} \subseteq E(T)$ and we are done.

Case 2: $e \notin T$. Since $G = (V, E)$ is connected, $T + e$ will contain a unique cycle. Then there will by another edge e' crossing S and $\bar{S}$ in that cycle. Then $T' = T + e - e'$ is again a tree by the edge exchange property of tree(proved in week 8's assigment). Now since e is of minimum cost, then $w(e) \leq w(e')$. But T' is a tree, its weight must be greater or equal than the weight of MST, $w(T') = W(T) + w(e) - w(e') \geq w(T)$. Then it must be the case that $w(e) \leq w(e')$. This in turn mean that $w(T') = w(T)$, i.e., T' is also a MST. Since $X \cup \{e\} \subseteq E(T')$ by construction, $X \cup \{e\}$ is good too.

[**Proof of the inverse of the Cut Lemma**]

Since $X \cup e$ is good, there is a MST T such that $(X \cup e) \subseteq E(T)$. The e is a cut edge of the MST $T = (V, E(T))$. $T - e$ consists of two connected components and we name them respectively S and $\bar{S}$. It is easy to verify (i) that no edge in $T - e$ cross S and $\bar{S}$.

It remains to prove that e is a minimum weight edge of G crossing this cut. Suppose there is e' in G such that $w(e') < w(e)$. $T - e + e'$ will be a tree with weight less then a MST. This is not possible. So e is a minimum weight edge of G crossing this cut(ii). ∎

Problem 52. For $c \in \mathbb{R}$ and a weighted graph $G = (V, E)$, let $G_c := (V, \{e \in E \mid w(e) \leq c\})$. That is, G_c is the subgraph of G consisting of all edges of weight at most c.

[**Lemma**] Let T be a minimum spanning tree of G, and let $c \in \mathbb{R}$. Then T_c and G_c have exactly the same connected components. (That is, two vertices $u, v \in V$ are connected in T_c if and only if they are connected in G_c).

1. Illustrate Lemma with an example!
2. Prove the lemma.

Proof. 1. The example of a graph on 6 vertices.

2.[**Proof of the Lemma**]

1)$u, v \in V$ are connected in $T_c \Rightarrow u, v \in V$ are connected in G_c.

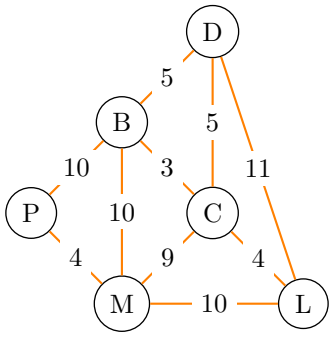
Since T_c is a subgraph of G_c, so this direction is obvious.

2)$u, v \in V$ are connected in $G_c \Rightarrow u, v \in V$ are connected in T_c.

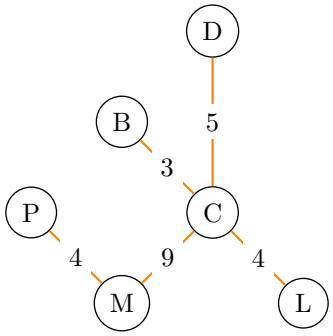
We prove by contradiction. Suppose $u, v \in V$ are connected in G_c and $u, v \in V$ are not connected in T_c. This means that the u, v-path in T contains at least one edge with weight grater than c. We note that edge e.

Now consider the the graph $X = T - e$. X is good (as in Problem 1). The graph X has at least two connected components (any edge in a tree is a cut edge). So we can divide X into two connected subgraph S and $\bar{S}$ with no edge in X crossing the S and $\bar{S}$. On the other hand, since G_c is connected, there is a edge $e' \in E(G_c)$

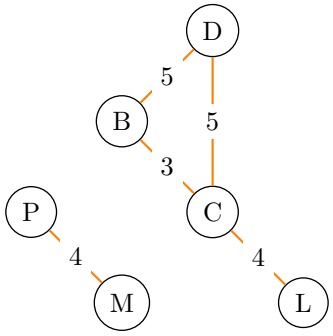
Example of Problem 2 : The example of a graph on 6 vertices.



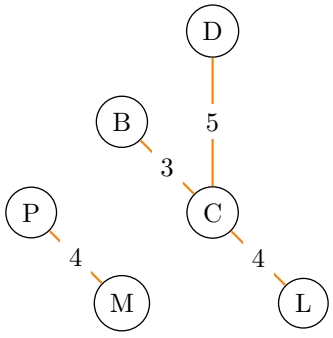
The graph $G = (V, E)$



The minimum spanning tree T



The graph G_5



The minimum spanning tree T_5

crossing S and $\bar{S}$ and $e' \neq e$. We can always choose e' so that e' has the minimum weight among all these edges. Now, by the cut lemma, $T' = T - e + e'$ is good too. Indeed, T' is another MST of G.

But there is a contradiction here. Since $e \notin E(T_c)$ and $e' \in E(G_c)$, we have $w(e) > c \geq w(e')$. This means that $w(T') = w(T) - w(e) + w(e') < w(T)$. This is not possible since both T and T' are MST.

So $u, v \in V$ are connected in $G_c \Rightarrow u, v \in V$ are connected in T_c. ■

Problem 53. For a weighted graph G, let $m_c(G) := |\{e \in E(G) \mid w(e) \leq c\}|$, i.e., the number of edges of weight at most c (so G_c has $m_c(G)$ edges).

[**Lemma**] Let T, T' be two minimum spanning trees of G. Then $m_c(T) = m_c(T')$.

1. Illustrate Lemma with an example!

2.Prove the lemma.

Proof. 1. Example where $m(T_5) = m(T_5') = 4$.

2.**[Proof of the Lemma]**

Actaully, if T and T' are two MST, then there is a sequence $T = T_0, T_1, \ldots, T_k = T'$ of MST of G such that $T_{i+1} = T_i - e'_{i+1} + e_{i+1}$ for some e'_{i+1} and e_{i+1} with $w(e'_{i+1}) = w(e_{i+1})$. It suffices to find one such step whenever T' is different from T ; the sequence then exists by using induction on the number of edges in which the two trees differ.

Choose any $e' \in E(T') - E(T)$. Deleting e' from T' creates two connected components with vertex sets S,$\bar{S}$. The path in T between the endpoints of e' must have an edge e from S to $\bar{S}$; then $T' - e' + e$ is a spanning tree. We want to show that $w(T' - e' + e) \leq w(T')$. Since e is an edge of the path in T between the endpoints of e' , the edge e belongs to the unique cycle in $T + e$'. Thus $T + e' - e$ is also a spanning tree. Because $T + e' - e$ is a spanning tree and T has minimum weight, $w(e) \leq w(e')$. Thus $T' - e' + e$ moves from T' toward T without increasing the weight. Since T' is of minimum weight, $T' - e' + e$ is also of minimum weight. Actually, $w(e') = w(e)$.

We note that in each step, we replace one edge with another edge of same weight. So we preserve the number of edges with weight smaller or equal than c. Thus $m_c(T) = m_c(T')$. ■

Problem 54. Suppose no two edges of G have the same weight. Show that G has exactly one minimum spanning tree!

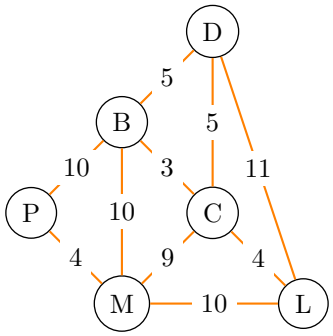
Proof. Suppose the edge weights are distince. Let T be an MST. For each edge $e \in E(T)$, consider the graph $T - e$. It has two connected components with vertex sets S,$\bar{S}$. The only edge in T that crosses the cut $S, \bar{S}$ is e.

Since T is a MST, e must be of the minimum weight among all the edges that crosses the cut. Since G has distince edge weiths, e is the unique edge of minimum weight for this cut. This means that every MST must contain e.

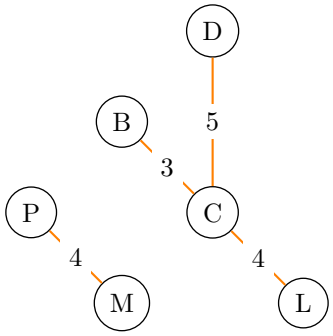
Since it is ture for every edge in $E(T)$, this means that any MST of G must be equal to T, i.e., G has exactly one minimum spanning tree. ■

Problem 55. Suppose H is a graph, not necessarily connected. A **maximal forest** is an acyclic subgraph graph with a maximum number of edges. Equivalently, it contains a spanning tree for each connected component of H. A **multigraph** is a graph that can have multiple edges, called "parallel dges". Without defining it formally, we illustrate it:

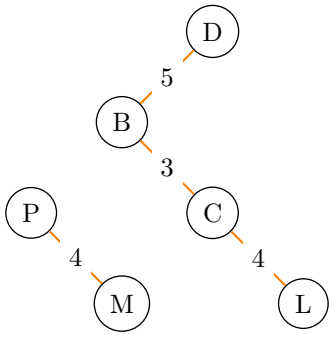
Example of Problem 3 : The minimum spanning tree T_5', $\mathbf{m}(T_5')$=4.



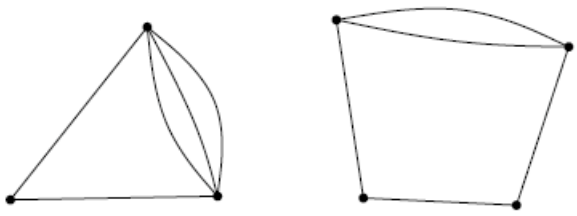
The graph $G = (V, E)$



The minimum spanning tree T_5, m(T_5)=4.



The minimum spanning tree T_5', m(T_5')=4.



A multigraph.

All other definitions, like connected components, spanning trees, spanning forests, are the same as for normal (non-multi) graphs. However, when two spanning forests use different parallel edges, we consider them different:

The same multigraph with two different spanning forests.

How many spanning forests does the above multigraph on 7 vertices have? Justify your answer!

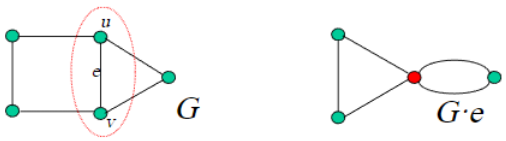
Solution. For each connected component, we first shrink the multigraph into simple graph and count the number of spanning trees. Then for each spanning trees containing the edge connecting the endpoints having multiple edges in the multigraph with order m, there are m spanning trees in the connected component of the multigraph.

For the left subgraph, there are 7 spanning trees. For the right subgraph, there are 7 spanning trees. So for the entire multigrpah, there are $7 \times 7 = 49$ spanning forests.

Problem 56. Suppose you have a polynomial-time algorithm that, given a multigraph H, computes the number of spanning forests of H. Using this algorithm as a subroutine, design a polynomial-time algorithm that, given a weighted graph G, computes the number of minimum spanning trees of G.

Solution. We need edge contraction in our algorithm to count the number of MST. We describe edge contracion informally : The edge contraction operation occurs relative to a particular edge, e. The edge e is removed and its two incident vertices, u and v, are merged into a new vertex w, where the edges incident to w each correspond to an edge incident to either u or v.

Usually an edge contraction will result in multigraph as shown by the following example.



Now we are ready for the algorithm. We note N the number of edges and n the number of vertices.

1. We partition the edges by weight in increasing order. So $E = \uplus_{i=1}^{K} E_i$ where E_i consists of edges with the i-th smallest weight. We may use the mersge sort. This time complexity of sorting is $O(NlogN)$.

2. Initial Step: We consdier $G_1 = (V, E_1)$ with E_1 consists of the edges with only the minimum weight w_1. We apply our subroutine to find the number of spanning forest for G_1, the result is M_1. Now we consider the $G_1' = (V, E_1 \cup E_2)$, the edges

now are of weight w_1 and w_2. We contract all the edges with w_1 (edges in E_1) in G'_1, delet self loop if any, and we set the resulting graph to be $G_2 = (V, E(G_2))$.

3. The general procedure it to apply the subroutine to the graph $G_k = (V, E(G_k))$, get he result M_k and contract in the graph $G'_k = (V, E(G_k) \cup E_{k+1})$ all the edges in $E(G_k)$.

4.Our procedure will end when G_k is connected. We note this k to be K'. Then we just compute $M_{K'}$ and the number $M = M_1 \times M_2 \times \cdots \times M_{K'}$ is the desired number of MST.

[**Time complexity**]

Our algorithm is obvious polynomial since maximum number of calling our subroutine is bounded from above by $N \leq \frac{1}{2}n(n-1)$. (We dont need tight bound here).

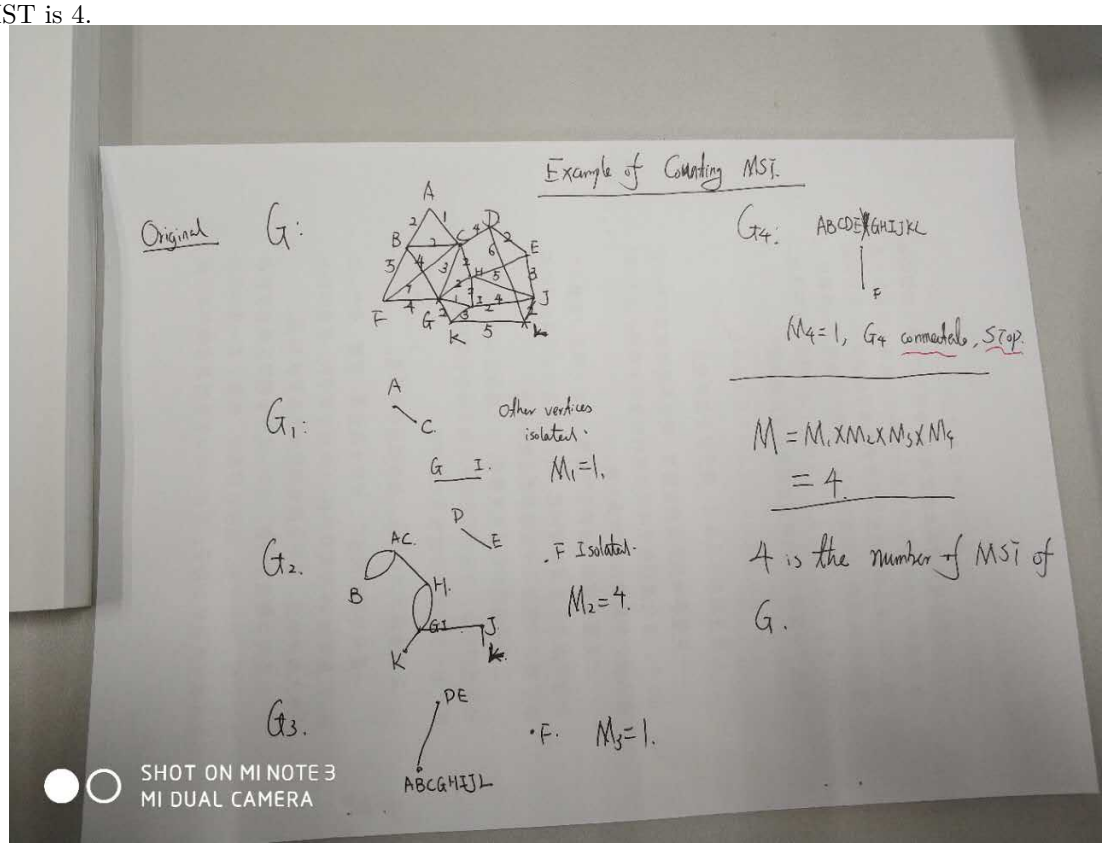
Also edge contracting is polynomial and called at most $n-1$ times.

[**Correcteness**]

We will not prove the correcteness of the our algorithm. But if we think carefully the Kruskal's algorithm, then it would be obvious that M_i is the number of choice of edges with weight w_i. Then by the multiplication rule, the number of MST $M = M_1 \times M_2 \times \cdots \times M_{K'}$.

[**One Example**]

The following is an example of our algorithm. The graph is taken from the Problems 4 of the Quzzi. The algorithms runs 4 iteration. And the number of MST is 4.



Problem 57. Let $G = (V, E)$ be a graph and $c : E \to \mathbb{R}$ be an edge cost function. We call an edge $e \in E$

(1) **necessary** if e is contained in every minimum spanning tree;

(2) **useless** if no minimum spanning tree contains e;

(3) **optional** if else.

Design an efficient algorithm which computes the set of necessary, optional, and useless edges for a given graph with edge costs.

[Remark] Do not try to find an extremely efficient algorithm. Try to find something as simple as possible!

Solution. We adapt from the algorithm in Problem 6. We just describe how it is done.

For each G_k, we assume that there are n_k connected components. Three case here:

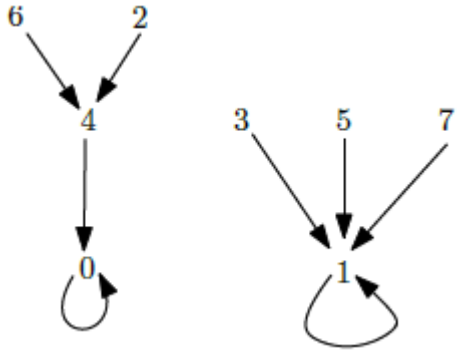
1) components contain a cycle as subgraph. Every edge in that cycle is optional.

2) components contains a tree as subgraph. Then every edge of that tree is necessary.

3) All other case, the edge is useless.

The Numbers of Trees on n Vertices.

Problem 58. Let V be a set of size n. We have learned that there are n^n functions $f : V \to V$. For such a function we can draw an "arrow diagram" by simply drawing an arrow from x to $f(x)$ for every V. For example, let $V = \{0, \ldots, 7\}$ and $f(x) := x^2 \mod 8$. The arrow diagram of f looks as follows:



The core of a function is the set of elements lying on cycles in such a diagram. For example, the core of the above function is $\{0, 1\}$. Formally, the core of f is the set

$\text{core}(f) = \{x \in V | \exists k \geq 1 : f^{(k)}(x) = x\}$ where $f^{(k)}(x) = f(f(\ldots f(x)\ldots))$, i.e., the function f applied k times iteratively to x.

Of the n^n functions from V to V, how many have a core of size 1? Give an explicit formula in terms of n and prove its correctness.

Solution. The number of the functions from V to V having a core of size 1 is $n^{n-1} = n \times n^{n-2}$. Actually, n^{n-2} is the number of trees on n vertices and n is the choice of the **absorbing state**(the element in the core).

Now we give a bijection between the function from V to V having a core of size and the tree on n vertices with one absorbing state.

On one hand, given a tree T, and one **absorbing state**, we can identify a function from V to V having a core of size 1. How to do this? Let v be the unique

element in the core. For any element u not in the core, there is a unique $u, v-$path. We note z the next vertex after u on this path. We then set $f(u) = z$. For v in the core, we set $f(v) = v$. Since tree is acyclic, for any u not in the core, there is no k such that $f^{(k)}(u) = u$. And by definition, $f(v) = v$. So the function f is well defined and the size of core is 1.

On the other hand, let f be a function from V to V having only one core $v \in V$.

1) It must be the case that $f(v) = v$. Suppose it is not the case. Let l be the smallest number such that $f^{(l)}(v) = v$. Then $k \geq 2$. Let $u = f^{(1)}(v) \neq v$. It is easy to verify that $f^{(l)}(u) = u$. So u is also in the core. This contradicts the fact that the size of core is 1.

2)For any $u \neq v$, there must be a k such that $f^{(k)}(u) = v$. Suppose it is not the case. Then by the pigeonhole principle, there must be k_1 and k_2 such that $f^{(k_1)}(u) = f^{(k_2)}(u) \neq v$. This contradicts the fact that the size of core is 1.

3)Now we can construct a graph $G = (V, E)$. We set $E = \{e = (u, u')|\exists u, u' \in V, u \neq v, f(u) = u'\}$. That is for any $u \neq v$, the edge incident to u is of the form (u, u') such that $u' = f(u)$ or of the form (u', u) such that $u = f(u')$. It is easy to verify that G is a tree.

a) G is connected since every $u \neq v$ is connected to v as there is a $k \geq 1$ such that $f^{(k)}(u) = v$.

b) G is acyclic. This is because the size of the core of f is 1, for any $u \in V, u \neq v$ there is no k such that $f^{(k)}(u) = u$.

Now we add to the set of edges E an self-loop of v. Then we have a desired tree with one absorbing state.

Problem 59. Let $f : V \to V$ be a function. An element $x \in V$ is a source of f if it has no incoming arc, i.e., $\forall y \in V : f(y) \neq x$. For example the function in the picture above has five sources. Let $s(f)$ denote the number of sources of f. Let T be a tree on vertex set V. A leaf is a vertex of degree 1. Let $\ell(T)$ denote the number of leaves in T. Recall that a vertebrate is a triple (T, a, b) where T is a tree on V and $a, b \in V$. We call a the head and b the tail of the vertebrate.

Recall that there is a one-to-one correspondence between functions $f : V \to V$ and vertebrates (T, a, b). Find the precise connection between $s(f)$ and $\ell(T)$.

Solution. 1) We label the element in V so that $V = \{x_1, x_2, \ldots, x_N\}$. In the following we will use interchangeably the numbering of an element $\{1, 2, 3, \ldots, N\}$ and $\{x_1, x_2, \ldots, x_N\}$ to represent the element in V.

Let the core of f to be $C = \{x_{a_1}, \ldots, x_{a_n}\}$ with $\{a_1, a_2, \ldots, a_n\}$ an increasing sequence.

By the bijectiong between (T, a, b) and f, we know that the spline of (T, a, b), the unique a, b-path, consists of elements in the core$\{a_1, a_2, \ldots, a_n\}$. Let $a_{\min} = \min\{a_1, a_2, \ldots, a_n\}$ and $a_{\max} = \max\{a_1, \ldots, a_n\}$. Then the head a =index of $f(x_{a_{\min}})$ and the butt b =index of $f(x_{a_{\max}})$.

2) Other than the vertices in the a, b-path, the number of source and the number of the leaves are the same. This is relative easy to see.

3) No vertices in the a, b-path could be source. Only vertices in the a, b-path that may be leaves are a and b. If there is $u \notin a, b$-path such that $f(u) = a$ or $f(u) = b$, then respectively a or b will not be a leaf.

[**Final results**] Let (T, a, b) be the corresponding vertebrate for f. Summarizing the above analysis, we have $\ell(T) = s(f) + c(T)$ where $c(T) = 1_{\{a \text{ is a leaf}\}} + 1_{\{b \text{ is a leaf}\}}$.

Problem 60. In the following we talk about random functions and trees and expectation ($\mathbb{E}$) of certain random variables. Here, a random function $f : V \to V$ is one picked uniformly at random from all n^n functions, and a random tree is one picked uniformly at random from all trees on V.

In case you are unfamiliar with the notion of probability and expectation, we re-formulate each exercise in a "combination way. Let f be a random function from V to V. What is the expectation $\mathbb{E}[s(f)]$?

[Combination version.]Fix an element $x \in V$. What is the number of functions $f : V \to V$ for which x is a source?

Give an explicit formula for $\sum_f s(f)$, where the sum is over all functions $f : V \to V$.

Solution. 1.Fix an element $x \in V$. We will show that the number of functions $f : V \to V$ for which x is a source is $(n-1)^n$.

Why? Since x is a source, the value of $f(x)$ can have $n-1$ choices (except x), and for any $u \in V\backslash\{x\}$, $f(u) \neq x$. That means f restricted on $V\backslash\{x\}$ has $V\backslash\{x\}$ as range.

So for f restricted on $V\backslash\{x\}$, there are $(n-1)^{n-1}$ choices. So the total number of function $f : V \to V$ for which x is a source is $(n-1) \times (n-1)^{n-1} = (n-1)^n$.

2. We argue that $\sum_f s(f) = n \times (n-1)^n$. Why? We recognize the right side of the equation to the $\sum_{x \in V}$ the number of functions f:V→V for which x is a source. Then f with $s(f) = 1$is counted once, ...,f with $s(f) = k$ is counted k times.

3. $\mathbb{E}[s(f)] = \frac{\sum_f s(f)}{\text{total number of functions}} = \frac{n(n-1)^n}{n^n} = n(\frac{n-1}{n})^n$.

Problem 61. Let T be a random tree on vertex set V. What is $\mathbb{E}[\ell(T)]$?

[Combination version.] Give an explicit formula for $\sum_T \ell(T)$ where the sum is over all trees T on vertex set V.

Solution. Given the answer to problem 59, we only need to find the expectation of $c(T)$

For $x \in V$, we now count the number of vertebrate such that x is head and x is leaf . The number of vertebrates such that x is head and x is leaf is equals to the number of vertebrates on set $V\backslash\{x\}$, which equals to $(n-1)^{n-1}$. Same way , the number of vertebrates such that x is tail and x is leaf is $(n-1)^{n-1}$.

So $\sum_T c(T) = 2 \times n(n-1)^{n-1}$. The reason is the same as problem 3, we sum the number of vertebrates such that x is head and x is leaf and the number of vertebrates such that x is tail and x is leaf over $x \in V$ to get the sum over $T \in \mathcal{T}$ of $c(T)$. The same trick of double counting is used.

Then $\mathbb{E}[c(T)] = \frac{\sum_T c(T)}{n^n} = 2(\frac{n-1}{n})^{n-1}$. Then $\mathbb{E}[\ell(T)] = \mathbb{E}[s(T)] + \mathbb{E}[c(T)] = (n+1)(\frac{n-1}{n})^{n-1}$.

10. Flow Networks, Flows, Cuts

Problem 62. Let$G = (V, E, c)$ be a network with capacities. That is, (V, E) is a directed graph and$c : E \to \mathbb{R}^+$ are capacities. We can generalize c to a function $V \times V \to \mathbb{R}^+$by setting$c(u, v) = 0$ whenever $(u, v) \notin E$. An s, t-flow in G is a function f: $V \times V \to \mathbb{R}$ such that

- $f(u, v) = -f(v, u)$ (**Skew symmetry**)
- $\sum_{v \in V} f(u, v) = 0 \ \forall u \in V \setminus \{s, t\}$ (**Flow conservation**)
- $f \leq c$(**Capacity constraints**)

The value of a flow is defined to be $\text{val}(f) := \sum_{v\in V} f(s,v)$.

(1) Prove that $\sum_{u,v\in S} f(u,v) = 0$ for all sets $S \subseteq V$.
(2) Prove that $\text{val}(f) = \sum_{v\in V} f(v,t)$.
(3) Let $S \subseteq V$ be a set with $s \in S$ and $t \notin S$. The flow from S to $V\setminus S$ is defined as $f(S, V \setminus S) := \sum_{u\in S, v\in V\setminus S} f(u,v)$. Prove that $f(S, V \setminus S) = \text{val}(f)$.

[Note]Don't be too informal here! The statements might sound obvious to you. Still, give a formal proof, deriving them from the "flow axioms", i.e., skew symmetry, flow conservation, and capacity constraints.

Proof. **1)** Let S be a subset of V. We observe that $(u,v) \in S \times S$ iff (v,u) is also in $S \times S$.Then $\sum_{u,v\in S} f(u,v) = \sum_{u,v\in S} f(v,u)$. Then we can write $\sum_{u,v\in S} f(u,v) = \frac{1}{2}\sum_{u,v\in S}[f(u,v) + f(v,u)] = 0$. The last equality is because of Skew symmetry.

2) We decompose V in to $\{s,t\}, V\setminus\{s,t\}$, then

$$\begin{aligned}\sum_{u,v\in V} f(u,v) &= \sum_{v\in V} f(s,v) + \sum_{v\in V} f(t,v) + \sum_{\substack{u\in V\setminus\{s,t\}\\ v\in V}} f(u,v)\\ &= \sum_{v\in V} f(s,v) + \sum_{v\in V} f(t,v) \text{ (by Flow conservation)}\\ &= \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,t) \text{ (by Skew symmetry)}\end{aligned}$$

Since by 1) we have $\sum_{u,v\in V} f(u,v) = 0$, then $\sum_{v\in V} f(s,v) = \sum_{v\in V} f(v,t)$, that is $\text{val}(f) = \sum_{v\in V} f(v,t)$.

3) The key is to notice that $f(S, V \setminus S) = f(S,V)$. Indeed, we have $f(S,V) = f(S,S) + f(S, V\setminus S) = f(S, V\setminus S)$. (The notation is $f(A,B) := \sum_{u\in A, v\in B} f(u,v)$)

Then

$$\begin{aligned}f(S,V) &= f(s,V) + f(V\setminus\{s,t\}, V)\\ &= \sum_{v\in V} f(s,v) + \sum_{u\in V\setminus\{s,t\}} \underbrace{\sum_{v\in V} f(u,v)}_{=0 \text{ by Flow conservation}}\\ &= \sum_{v\in V} f(s,v) = \text{val}(f).\end{aligned}$$

■

Problem 63. An $s-t$ -cut is a set $S \subseteq V$ such that $s \in S, t \in V \setminus S$. The capacity of the cut S is $c(S, V \setminus S) := \sum_{u\in S, v\in V\setminus S} c(u,v)$. Let f be a flow in G and S be a cut. Show that $\text{val}(f) \le c(S, V \setminus S)$.

Proof. By **the result in problem 1**, we have $\text{val}(f) = f(S, V\setminus S)$. Then it remains to prove that $\sum_{u\in S, v\in V\setminus S} f(u,v) \le \sum_{u\in S, v\in V\setminus S} c(u,v)$. This is evident since $f \le c$ on $V \times V$. ■

Problem 64. Prove that flow is "transitive" in the following sense: If there is a flow from s to r of value k , and a flow from r to t of value k , then there is a flow from s to t of value k .

Hint. The solution is extremely short. If you are trying something that needs more than 3 lines to write, you are on the wrong path.

Proof. We will use the fact that for a $s-t$ flow f, for any $s-t$ cut S and $V\backslash S$, we have $f(S, V \setminus S) = \text{val}(f)$.

Suppose there was no flow of value k from s to t. Then there exists an $s-t$ cut $s \in S$, $t \in V\backslash S$ such that $f(S, V\backslash S) < k$. Then either $r \in S$ or $r \in V\backslash S$.

If $r \in S$,then there is a cut between r and t that is less than k and there is no flow between r and t of value k.

If $r \notin S$, then there is a cut between s and r that is less than k and there is no flow between s and r of value k.

In either case, we reach a contradiction. So there must be a flow from s to t of value k. ■

Problem 65. Suppose we have a directed graph $G = (V, E)$ but instead of edge capacities we have vertex capacities $c : V \to \mathbb{R}$. Now a flow f should observe the vertex capacity constraints, i.e., the outflow from a vertex u should not exceed $c(u)$:

$\forall u \in V : \sum_{v \in V, f(u,v)>0} f(u, v) \leq c(u)$.

Consider networks with vertex capacities.

Show how to model networks with vertex capacities by networks with edge capacities. More precisely, show how to transform $G = (V, E, c)$ with $c : V \to \mathbb{R}$ into a network $G' = (V', E', c')$ with $c' : E' \to \mathbb{R}$ such that every $s-t$ -flow f in G that respects the vertex capacities corresponds to an $s-$t -flow f' (of same value) in G' that respects edge capacities, and vice versa.

Draw a picture illustrating your solution.

Solution. [Corresponding G']

We suppose that there is no vertex capacity constraint for s and t. (it would be very easy to adapt our solution to the case where there are capacity constraints for s and t, but it will UN-necessarily complicate our discussion).

With $G = (V, E, c)$ with $c : V \to \mathbb{R}$, we do the following procedure to get a $G' = (V', E', c')$ with $c' : E' \to \mathbb{R}$.

1) Split each vertex $v \in V$ other than s, t into v^{in} and v^{out}. That is if $V = \{s, v_1, \ldots, v_n, t\}$, then $V' = \{s, v_1^{in}, v_1^{out} \ldots, v_n^{in}, v_n^{out}, t\}$.

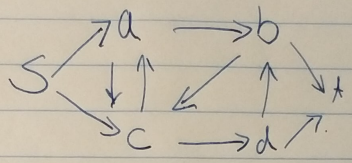
2) We construct E' such that for each edge not incident to s and t , $e = (u, v) \in E$, we have $e' = (u^{out}, v^{in}) \in E'$ and we also have for any $v \in V$, $(v^{in}, v^{out}) \in E'$. For edges of the form $(s, v) \in E$ (connecting the source), we have $(s, v^{in}) \in E'$. For edges of the form $(u, t) \in E$ (connecting the sink), we have $(u^{out}, t) \in E'$.

3) For each $v \in V$, let $e' = (v^{in}, v^{out})$, then the edge capacity of the edge e' is equal to the vertex capacity of v,i.e., $c'(e') = c(v)$. For other edge in E', the edge capacity c' is infinite. (In practice, we can replace infinite with a very large number) For other pairs of vertices(not in the edge set E'), the edge capacity is zero.

[Example]

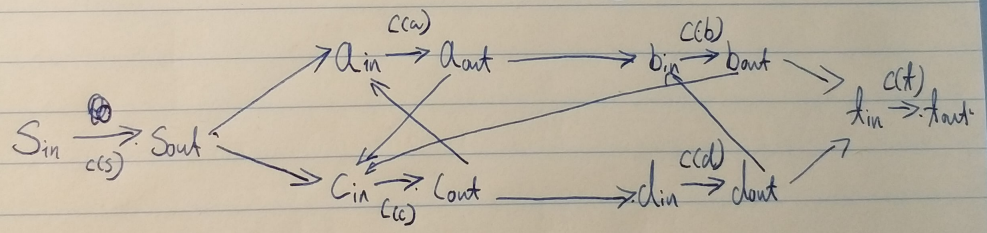
The example on 6 vertices if given below. (N.B. The following graph show an example when there are vertices capacity constraints on s, t. Just ignore the $s^{in}, s^{out}, t^{in}, t^{out}$ parts in the example. Thanks!)

$G = (V, E)$.



- Vertex capacity: $c(s) = 10$, $c(a) = 6$, $c(b) = 7$, $c(c) = 8$, $c(d) = 9$, $c(t) = 10$

- Equivalent edge capacity.:
  - Split the vertices into in vertices and out vertices.
  - The vertex capacity → edge capacity between in and out.



Other edge with infinite capacity.

In practice, can set their capacity to a very large number.

[Define the Flow f' in G']

Given an $s-t$ -flow f in $G=(V,E,c)$ we now show how to construct an $s-$t -flow f' (of same value) in (V,E,c).

It is relatively easy, for any $e'=(u,v)\in V'\times V'$, e' is either of the form (u^{out},v^{in}) for some $e=(u,v)\in E$ or of the form (u^{in},u^{out}) for $u\in V$ or of other form.

If $e'=(u^{out},v^{in})$ for some $e=(u,v)\in E$, then we set $f'(u^{out},v^{in}))=f(u,v)$. Also, we set $f'(v^{in},u^{out})=-f(u^{out},v^{in})$.

If $e'=(u^{in},u^{out})$ for some $u\in V$, and $u\neq t$, then we set $f'(u^{in},u^{out})=\sum_{v\in V,f(u,v)>0}f(u,v)$. Also, we set $f'(u^{out},u^{in})=-f(u^{in},u^{out})$.

For all the pairs of vertices containing the source s or the sink t, we just set $f'=f$ (here we confound the splitted vertices and the original vertices.)

If other case, $f'=0$ since there is no flow.

It remains to verify that f' is a $s-t$ flow with value val(f). We verify the following three flow conditions one by one:

1)(Skew symmetry) $f'(u',v')=-f(v',u')$ for $u',v'\in V'\times V'$. This is automatic by construction.

2)(Flow conservation) $\sum_{v'\in V'}f(u',v')=0\ \forall u'\in V'\setminus\{s,t\}$.

Two case here.

If u' is one of the u^{out}, then $\sum_{v'\in V'}f(u',v')=\sum_{v\in V,f(u,v)>0}f(u,v)+f(u^{out},u^{in})=\sum_{v\in V,f(u,v)>0}f(u,v)-f(u^{in},u^{out})=0$.

If u' is one of the u^{in}, then $\sum_{v'\in V'}f(u',v')=\sum_{v\in V,f(u,v)<0}f(u,v)+f(u^{in},u^{out})=\sum_{v\in V,f(u,v)<0}f(u,v)+\sum_{v\in V,f(u,v)>0}f(u,v)=\sum_{v}f(u,v)=0$.

3)(Capacity constraints) $f'\leq c'$. This is also automatically satisfied by construction.

[Value of f' equal that of f]

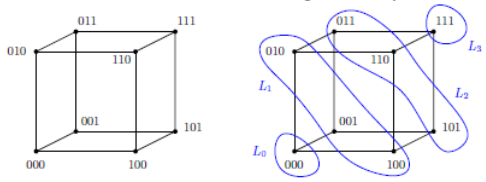
Now we prove that val$(f')=$ val(f). This is easy.

In G we consider cut $S=\{s\},\bar{S}=V\setminus\{s\}$. And in G' we consider cut $S'=\{s\},\bar{S'}=V'\setminus\{s\}$. It is clear that val$(f)=f(S,\bar{S})$ and val$(f')=f'(S',\bar{S'})$. Now by the construction of f', for all the pairs of vertices containing the source s we have $f'=f$ (here we confound the split-ted vertices and the original vertices.) So $f(S,\bar{S})=f'(S',\bar{S'})$. So val$(f')=$ val(f).

Problem 66. Let $V=\{0,1\}^n$. The n-dimensional Hamming cube H_n is the graph (V,E) where $\{u,v\}\in E$ if u,v differ in exactly one coordinate. Define the i-th level of H_n as

$L_i:=\{u\in V\mid |u|_1=i\}$

i.e., those vertices u having exactly i coordinates which are 1 .



The 3-dimensional Hamming cube and the four sets L_0, L_1, L_2, L_3.

Let H_n be the n -dimensional Hamming cube. For$i<n/2$ consider L_i and L_{n-i} . Note that $|L_i|=\binom{n}{i}=\binom{n}{n-i}=L_{n-i}$, so L_i and L_{n-i} have the same size.

Show that there are $\binom{n}{i}$ paths $p_1,p_2,\ldots,p_{\binom{n}{i}}$ in H_n such that

(i) each p_i starts in L_i and ends in L_{n-i} ;

(ii) two different paths p_i, p_j do not share any vertices.

Solution. For a given $i < n/2$ we construct our flow network.

The set of vertex $V = \{s, L_i, L_{i+1}, \ldots, L_{n-i}, t\}$. The edges crossing the set L_j and L_{j+1} is the same as in H_n for $j \in [i, n-i-1]$. There are no edges within L_js for $j \in [i, n-i]$. And all the vertices in L_i is incident to the source s, all the vertices in L_{n-i} is incident to the sink t.

As in the problem 4, we assume the the $s-t$ flow in $G = (V, E)$ obeys the vertex capacity constraints. And in this case, the constraints is unit, that is for all $u \in V \backslash \{s, t\}$, $c(u) = 1$.

Now we have defined a flow network with vertex capacity $G = (V, E, c)$.

It is clear the value of the maximum $s-t$ flow in G is the number of distinct paths in H_n connecting L_i and L_{n-i}.

We note the value of maximum flow A.

Now we consider the corresponding flow network with edge capacity $G' = (V', E', c')$. (As in problem 4)

We note that $c'(e') > 0$ and finite iff e' connect the in-replica and out-replica for some $u \in V$, or formally, $e' = (u', v')$ with $u' = u^{in}$ and $v' = u^{out}$ for some $u \in V$.

For other $e' \in E'$, $c'(e')$ is infinite.

Now we solve the problem of finding a minimum cut of G'. We note the vertices set$V' = \{s, L_i^{in}, L_i^{out}, \ldots, L_{n-i}^{in}, L_{n-i}^{out}, t\}$. With $L_j^{in} = \{u^{in}, |u \in V\ , |u|_1 = j\}$ and $L_j^{out} = \{u^{out}, |u \in V\ , |u|_1 = j\}$ for $j \in [i, n-i]$.

With this notation, for all $e' \in E'$,$c'(e') > 0$ is finite $\Rightarrow$ e' cross L_i^{in} and L_i^{out} for some $j \in [i, n-i]$. (The notation should be obvious in the example graph.)

We claim that any minimum cut must be in some L_j for $j \in [i, n-i]$. Otherwise, the set of edges crossing the cut must contain at least one $e' \in E'$ for which $c'(e')$ is infinite. Then the capacity of the cut would also be infinite.

Now consider the cut index by j, $S_j = \{s, L_i^{in}, L_i^{out}, \ldots, L_j^{in}\}$ and $\bar{S}_j = \{L_j^{out}, \ldots, L_{n-i}^{in}, L_{n-i}^{out}, t\}$.

Clearly the capacity of the cut S_j, $c(S_j) = |L_j^{in}| = \binom{n}{j}$ because there are $\binom{n}{j}$ edges crossing L_j^{in} and L_j^{out} and the edge capacity for each edge is 1.

Then we claim the minimum cut is realized by S_i or S_{n-i}.

1) As the binomial number $\binom{n}{i} = \binom{n}{n-i} \leq \binom{n}{j}$ for $j \in [i, n-i]$, $c(S_i) = c(S_{n-i}) \leq c(S_j)$ for $j \in [i, n-i]$.

2) Actually any cut with finite capacity is of the form $S = \{s, L_i^{in}\} \cup \big(\cup_{j \in J \subseteq [i, n-i-1]} \{L_j^{out} L_{j+1}^{in}\}\big)$. It is not difficult to verify that S_i is of the minimum capacity.

Then by the Maxflow-Mincut Theorem, $A = \binom{n}{i}$. That is there are $\binom{n}{i}$ distinct paths connecting L_i and L_{n-i}.

[Example]

Case $n = 3$ and $i = 0$.

We can enumerate all the cut sets with finite capacity and theire capacities. There are 8 such cut, with is identical to the set of subset of $[i, n-i-1] = \{0, 1, 2\}$. Note that we do not have backward flow in the sense that $c(L_{j+1}^{in}, L_j^{out}) = 0$ for $j \in [i, n-i-1]$.

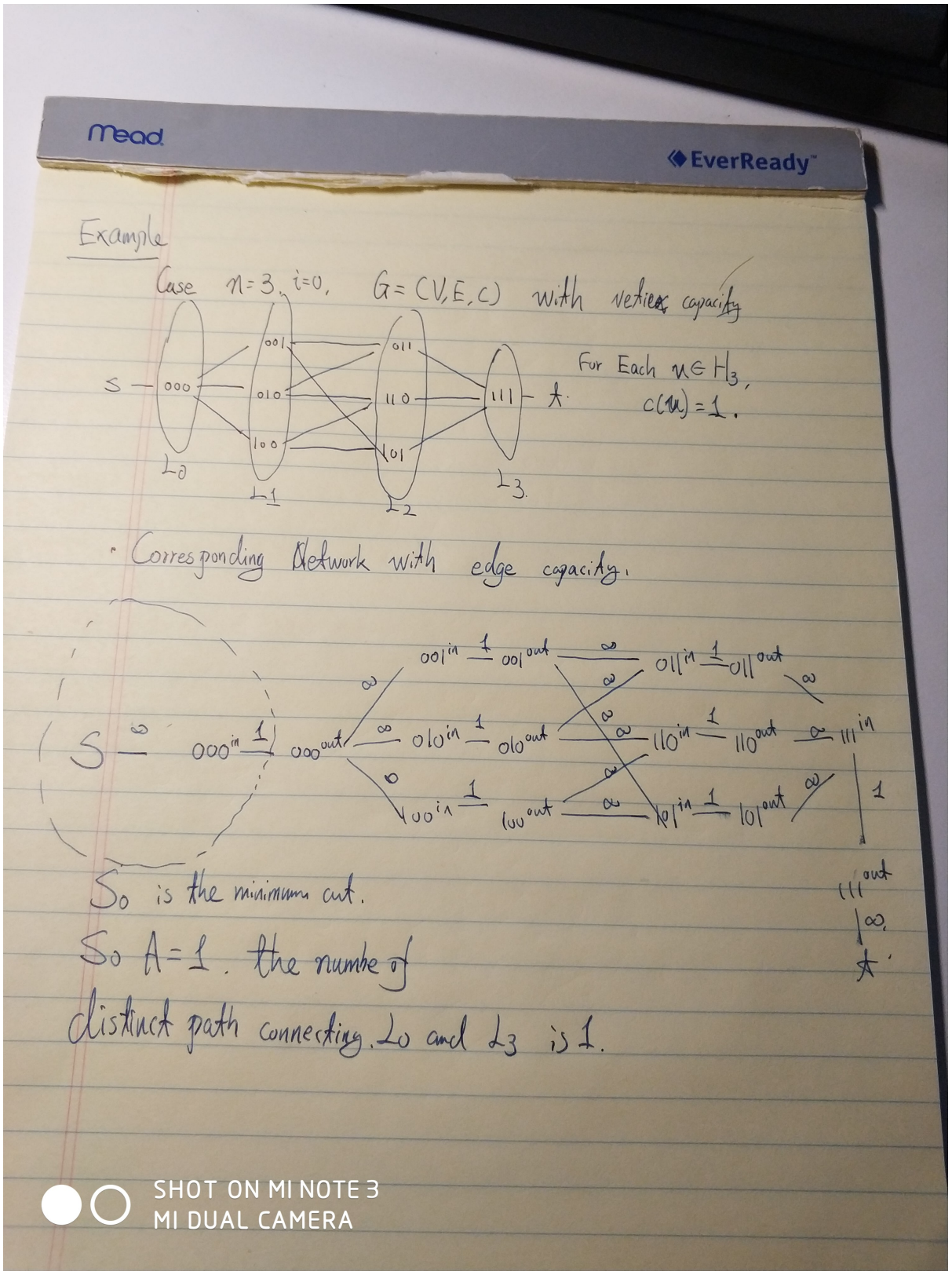
$S_0 = \{s, L_0^{in}\}$, $c(S_0) = 1$.

$S_1 = \{s, L_0^{in}, L_0^{out}, L_1^{in}\} = S_0 \cup \left(\cup_{j \in \{0\}} \{L_j^{out} L_{j+1}^{in}\}\right), c(S_1) = 3$.

$S_2 = \{s, L_0^{in}, L_0^{out}, L_1^{in}, L_1^{out}, L_2^{in}\} = S_0 \cup \left(\cup_{j \in \{0,1\}} \{L_j^{out} L_{j+1}^{in}\}\right), c(S_2) = 3$.

$S_3 = \{s, L_0^{in}, L_0^{out}, L_1^{in}, L_1^{out}, L_2^{in}, L_2^{out}, L_3^{in}\} = S_0 \cup \big(\cup_{j \in \{0,1,2\}} \{L_j^{out} L_{j+1}^{in}\}\big), c(S_3) = 1$.

$S_4 = \{s, L_0^{in}, L_1^{out}, L_2^{in}\} = S_0 \cup \left(\cup_{j\in\{1\}}\{L_j^{out}L_{j+1}^{in}\}\right), c(S_4) = 4.$
$S_5 = \{s, L_0^{in}, L_2^{out}, L_3^{in}\} = S_0 \cup \left(\cup_{j\in\{2\}}\{L_j^{out}L_{j+1}^{in}\}\right), , c(S_5) = 2.$
$S_6 = \{s, L_0^{in}, L_1^{out}, L_2^{in}, L_2^{out}, L_3^{in}\} = S_0 \cup \left(\cup_{j\in\{1,2\}}\{L_j^{out}L_{j+1}^{in}\}\right), c(S_6) = 2.$
$S_7 = \{s, L_0^{in}, L_0^{out}, L_1^{in}, L_2^{out}, L_3^{in}\} = S_0 \cup \left(\cup_{j\in\{0,2\}}\{L_j^{out}L_{j+1}^{in}\}\right), c(S_7) = 4.$


mead
EverReady
Example
Case n=3, i=0, G=(V,E,C) with vertex capacity
For Each u ∈ H3, c(u)=1.
s
000
001
010
100
011
110
101
111
t
L0
L1
L2
L3
Corresponding Network with edge capacity.
S
000in
000out
001in
001out
010in
010out
100in
100out
011in
011out
110in
110out
101in
101out
111in
111out
∞
1
S0 is the minimum cut.
So A=1. the numbe of
distinct path connecting L0 and L3 is 1.
SHOT ON MI NOTE 3
MI DUAL CAMERA

11. Matchings in Bipartite Graphs

Problem 67. Let G be a bipartite graph with left vertex set U , right vertex set V , and edge set E . For a set $X \subseteq U$ we let $\Gamma(X)$ be its neighborhood, i.e., the set $\Gamma(X) := \{v \in V \mid \exists\ u \in X : \{u, v\} \in E\}$ .The deficiency of G is $\delta(G) := \max_{X \subseteq U} |X| - |\Gamma(X)|$.

Hall's Theorem is usually stated in the following form, not mentioning $\delta(G)$ explicitly:

[Hall's Theorem, Usual Form] If $|\Gamma(A)| \geq |A|$ for every set $A \subseteq U$, then there is a matching of size $|U|$.

In the video lecture, we proved a stronger version:

[Hall's Theorem, Extended Form] If G is a bipartite graph, then the maximum matching has size $|U| - \delta(G)$.

Prove the extended form directly using the usual form,without going through flow networks.

Proof. Proof the Hall's Theorem extended from from the usual form.

Let G be a bipartite graph. Let $\delta(G) := \max_{X \subseteq U} |X| - |\Gamma(X)|$.

We note the size of maximum matching to be $\alpha(G)$.

1) We claim that $\alpha(G) \leq |U| - \delta(G)$.

Choose set $A \subseteq U$ such that $|A| - |\Gamma(A)| = \delta(G)$. Because saturated vertices of A must have distinct neighbors in any matching and only $|A| - \delta(G)$ neighbors are available, every matching leaves at least $\delta(G)$ vertices of U unsaturated. Thus $\alpha(G) \leq |U| - \delta(G)$.

2) We claim that G has a matching as large as $|U| - \delta(G)$.

We form a new graph G' by adding $\delta(G)$ vertices to the right set U and making all of them adjacent to all of vertices in U . This adds $\delta(G)$ vertices to $\Gamma(A)$ for each $A \subseteq U$, which yields$|\Gamma_{G'}(A)| \geq |A|$ for all $A \subseteq U$. By Hall's Theorem (usual form), G' has a matching of size $|U|$. Now if we delete the new vertices of G' , we will lose at most $\delta(G)$ edges of the matching. Hence what remains is a matching of size at least $|U| - \delta(G)$ in G. That is $\alpha(G) \geq |U| - \delta(G)$.

Combining 1) and 2), $\alpha(G) = |U| - \delta(G)$. ■

Problem 68. In the video lecture, we proved Kőnig's theorem:

[Kőnig's Theorem] In a bipartite graph, the size of the minimum vertex cover equals that of a maximum matching.

Prove Kőnig's Theorem using the extended form of Hall's Theorem, again without going through flow networks.

Solution. We note the size of minimum vertex cover of G to be $\beta(G)$.

1) It if clear that the size of a maximum matching is smaller than the size of a minimum vertex cover, since every edge in the matching has at least one endpoint in the vertex cover. That is $\alpha(G) \leq \beta(G)$.

2) We claim that a bipartite graph G has a matching and a vertex cover of the same size. Consider an U, V -bigraph G in which A is a subset of U with $|A| - |\Gamma(A)| = \delta(G)$. By Hall's theorem (Extended), $\alpha(G) = |U| - |A| + |\Gamma(A)|$. Let $C = (U \backslash A) \cup \Gamma(A)$. By the definition of $\Gamma(A)$, there are no edges joining A and $V \backslash \Gamma(S)$. Therefore, C is a vertex cover of G . The size of C is $|U| - |A| + |\Gamma(A)|$, which equals $\alpha(G)$. Thus G has a matching and a vertex cover of the same size, as desired.

Combining 1) and 2), for a bipartite graph G, $\alpha(G) = \beta(G)$.

Problem 69. Suppose I give you a maximum matching M for the bipartite graph G . Design a simple algorithm that finds a minimum vertex cover in $O(|E|)$ time. You can assume that $|E| \geq |V|$. By simple I mean your algorithm should re-use what you have learned from the lecture. Don't try to build something from scratch.

Solution. Algorithm to find a minimum vertex cover from a maximum matching:

Suppose that M is a perfect matching and $|U| \leq |V|$, then U is a vertex cover.

Otherwise, let U be one side of the partition that has some vertices which are unmatched. Let A be the set of unmatched vertices in U.

Otherwise, let B be the set of vertices connected to A by M-alternating paths (paths going from U to V along edges from unmatched edges, $E\backslash E(M)$ and from V to U along matched edges $E(M)$).

Let $Z = A \cup B$. Set $S = U \cap Z$ and $T = V \cap Z$. Then, every vertex in T is saturated and $\Gamma(S) = T$.

Set $C = (U\backslash S) \cup (T)$, then C is the desired vertex cover.

Proof of correctness:

The size of this vertex cover C is exactly the size of matching M. Thus C is a minimum vertex cover.

This is because of for each edge in the matching M incident to a vertex in T, we have exactly one endpoint in the cover C as the other endpoint of such edge is contained in neighborhood of MT in M, $\Gamma_M(T) \subseteq Z$.

Also, for each edge in the matching M not incident to any vertex in T, we have exactly one vertex in the set $U\backslash S$.

Running time analysis:

The running time of this algorithm is $O(|E|)$.

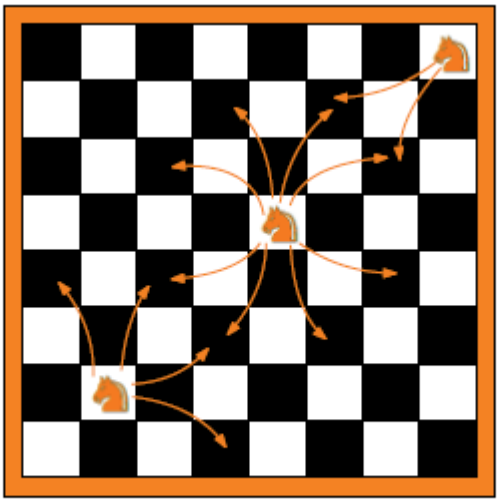
1) The cost of the operation to get A is $O(|V|)$.

2) We traverse at most all the edges in the graph G to identify the set B. So cost is $O(|E|)$.

3) And the cost operation to get $Z = A \cup B, S = U \cap Z, T = V \cap Z, C = (U\backslash S) \cup (T)$ is also $O(|V|)$.

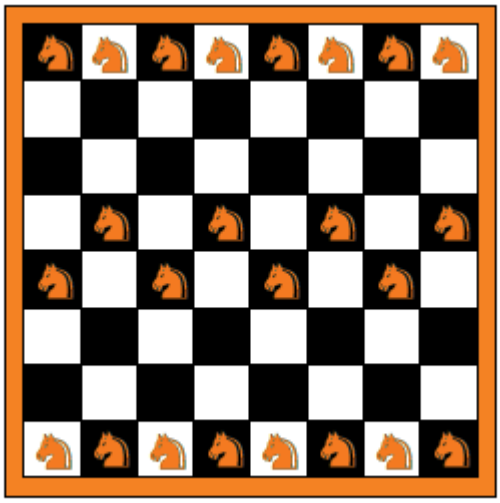
Since $|V| \leq |E|$, the total time cost of our algorithm is $O(|E|)$.

Problem 70. In chess, a popular board game, the knight is a piece which can move in a peculiar L-shaped (or Γ-shaped) fashion.



The moves of a knight. A knight in the center can perform eight different moves; closer to the boarder, only four moves might be possible; in a corner, only two.

We say an arrangements of knights on a chessboard is peaceful if no knight can attack any other. That is, if squares u, v are both occupied by a knight, then the knight on square u cannot move to v in one step (and neither vice versa).



An 8×8 chessboard with a peaceful arrangement of 24 knights.

Show that no peaceful arrangement can contain more than 32 knights.

Proof. We can build a graph $G = (V, E)$ with each vertex $v \in V$ representing one square in the 8×8 chessboard. The edges $e \in E$ represent one legal move for a knight on the chess board. That is if an u, v are two endpoints of an edge $e = (u, v) \in E$, then a knight can move from u to v. And vice versa.

Now a peaceful arrangement can be identified as an independent set in G. An **independent set** S in G is a set of vertices such that no two of which are adjacent. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two.

It suffices to prove the maximum independent set in G is of size 32. Note that if U is a maximum independent set in graph $G = (V, E)$, then $V \backslash U$ is a minimum vertex cover in G. It is well-known that G has a Hamilton cycle (the knight tour), implying that G has a perfect matching M of size 32 (take every second edge from the Hamilton cycle). And it is also well known that G is bipartite with each set in partition of size 32, then by the Hall's theorem(extended), M is a maximum matching. Then by the **Kőnig's Theorem** the minimum vertex cover $V \backslash U$ is of size 32. Which means that the maximum independent set U is also of size 32. ■

Problem 71. Let us consider non-standard chessboards. This means, width and height do not need to agree, and not even every square has to be present. Below is an example of a quite non-standard chessboard:

Design an algorithm which, given a non-standard chessboard, determines the maximum number of knights in a peaceful arrangement.

[Note] Give a simple algorithm that re-uses algorithms you have learned in class. Do not give a detailed from-scratch solution.

Solution. As in the problem 4, there problem equals to find the size of the minimum vertex cover for the associated graph $G = (V, E)$. Since the chess board is non-standard, the associated graph $G = (V, E)$ is not likely be either Hamiltonian nor bipartite. It is well known that the algorithm of finding a minimum vertex cover for an arbitrary graph is NP-hard.

But the remaining board is still bipartite with black squre and white square. Then by Konig's theorem, the maximum number of knights in a peaceful arrangement equals to the minimum vertex cover which equals to the maximum matching.

Now the size of maximum matching equals to the maximum flow in the following graph $G' = (V', E')$: add a source adjacent to all black and a sink adjacent to all white and make all the capacities 1.

Then the problem could be solved by a Ford–Fulkerson algorithm which cost $O(|E'|^2)$ in our case.

Problem 72. Suppose $G = (U, V, E)$ is our bipartite graph and $|U| = |V| = n$. A matching of size n is called a perfect matching. Of course you know by now how to search for a perfect matching. However, here is a cute algorithm which is faster and works whenever G has a unique perfect matching:

```
M := ∅
while G has a vertex u of degree 1 do
    let v be the unique neighbor of u
    add {u, v} to M
    remove {u, v} from G
end
return M
```

(1) Prove the following statement: If G is a bipartite graph with a unique perfect matching, then G has a vertex of degree 1.
(2) Prove that the algorithm is correct, i.e., if G has a unique perfect matching then the algorithm finds it.

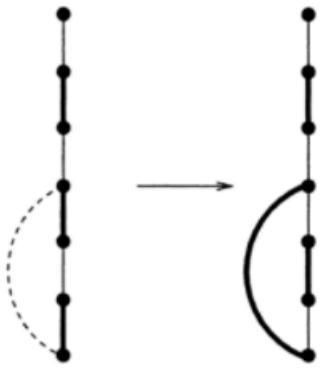
Solution. 1) If G is a bipartite graph with a unique perfect matching, then G has a vertex of degree 1.

We prove the conterpositive: If every vertices in G is of degree larger than 2 and if G has a perfect matching, then G will have at least one another perfect matching.

Let's reconsider our algorithm for matching. We start from the null match. Let u be any un-matched vertex of G, then there will an M-alternating path (one edge not in the match M and one edge in the match M) from u.

We can grow DFS a path P from u two edges at a time, repeatedly adding an unmatched edge (x, y) and the matched edge containing y.

So if we suppose every vertex in G is of degree larger than 2, than in the above path, there will an M-alternating cycle C of even length. Then there will be another perfect matching by using two alternating part of the cycle as well as other edges in M. Formally, M and $(M \backslash (C \cap M)) \cup C \cap M^c$ are two perfect matching.



2) We will prove the algorithm by showing each edge $\{u, v\}$ is in the unique perfect matching. Let M be the unique perfect matching in G. Let v_1 be the vertex of degree 1 and let u_1 be the unique vertex adjacent to v_1. Then by 1) we know that the edge (u_1, v_1) is in the matching. Now we consider the graph induced by $V \backslash \{u_1, v_1\}$, we call the induced graph G_2.

Since M is the unique perfect matching in G, $M \cap G_2$ is also the unique perfect matching of G_2. So the above algorithm will give use an edge (u_2, v_2) which is in

the matching. The process will proceed until there are two vertices left. And the last two vertices will form the last edge in the match.

Problem 73. Suppose I give you a minimum vertex cover C for the bipartite graph G. Design an algorithm that finds a maximum matching in $O(|E|)$ time. You can assume that $|E| \geq |V|$.

Solution. could not find.....

Email address: gabrielcai@126.com