20211507 컴퓨터공학과 김동건

1.2 Pigeons and holes. Properly speaking, we should call our representation of Königsberg a *multigraph*, since some pairs of vertices are connected to each other by more than one edge. A *simple* graph is one in which there are no multiple edges, and no self-loops.

Show that in any finite simple graph with more than one vertex, there is at least one pair of vertices that have the same degree. Hint: if n pigeons try to nest in $n-1$ holes, at least one hole will contain more than one pigeon. This simple but important observation is called the *pigeonhole principle*.

Assume that simple graph has n vertices.

Let d_i is degree of i-th vertices. $(1 \le i \le n)$

Then, $0 \le d_i \le n-1$. Because graph is simple.

But, If there is j that $d_j = 0$, then there are no k that $d_k = n-1$. And If there is k that $d_k = n-1$, then there are no j that $d_j = 0$

Because $d_j = 0$ means there are no vertices directly connected with j.

And $d_k = n-1$ means k is directly connected with all other vertices.

It is contradict. And so, there are at most $n-1$ degree value.

So, at least two vertices have same degree.

(pigeon : vertices (n)

pigeon hole : degree $(n-1)$

2.2 Euclid extended. Euclid's algorithm finds the greatest common divisor $\gcd(a,b)$ of integers a and b. Show that with a little extra bookkeeping it can also find (possibly negative) integers x and y such that

$$ax + by = \gcd(a,b). \tag{2.7}$$

Now assume that $b < a$ and that they are mutually prime. Show how to calculate the multiplicative inverse of b modulo a, i.e., the y such that $1 \le y < a$ and $by \equiv 1 \bmod a$.

Hint: the standard algorithm computes the remainder $r = a \bmod b$. The extended version also computes the quotient q in $a = qb + r$. Keep track of the quotients at the various levels of recursion.

[1]

$a = bq_2 + r_3$ ⋮ It will finish when $r_{i+1} = 0$ and then

$b = r_3 q_3 + r_4$ ⋮ $\gcd(a,b) = r_i$

⋮ Let's Assume $r_1 = a, r_2 = b$ ... ①

⋮ And we know all r, q By doing Euclid method

⋮

$r_{i-2} = r_{i-1} q_{i-1} + r_i$ ⋮ $r_3 = a - bq_2$

$r_{i-1} = r_i q_i + r_{i+1}$ ⋮ $r_4 = b - r_3 q_3 = b - (a - bq_2) q_3$

⋮

$r_i = r_{i-2} - r_{i-1} q_{i-1}$

$r_{i+1} = r_{i-1} - r_i q_i$

$r_{i+1} = 0$, $r_i = \gcd(a,b)$

So, we can represent r_i to $s_i a + t_i b$. $(r_i = s_i a + t_i b)$

Substitute this to $r_{i+1} = r_{i-1} - r_i q_i$,

$$s_{i+1} a + t_{i+1} b = (s_{i-1} a + t_{i-1} b) - (s_i a + t_i b) q_i$$

$$= (s_{i-1} - s_i q_i) a + (t_{i-1} - q_i t_i) b$$

So, $s_{i+1} = s_{i-1} - s_i q_i$, $t_{i+1} = t_{i-1} - t_i q_i$

And, by ①, $s_1 = 1, t_1 = 0, s_2 = 0, t_2 = 1$. So, we can iterate and calculate s_i and t_i that satisfy $s_i a + t_i b = r_i$. And r_i is $\gcd(a,b)$. So, we can calculate x, y that satisfy $ax + by = \gcd(a,b)$

2 Now $\gcd(a,b)=1$, so we can calculate s_i, t_i with same method in 1.

So, $s_i a + t_i b = 1$

Take both side modulo a, we can yet

$$t_i b \equiv 1 \mod a$$

we can add ka to t_i

so that $t_i + ka$ is in $[0, a)$.

(k is integer).

And that is multiplicative inverse.

2.18 How to mail a matrix. Given a graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, how many bits do we need to specify the adjacency matrix, and how many do we need to specify a list of edges? Keep in mind that it takes $\log n$ bits to specify an integer between 1 and n. When are each of these two formats preferable?

In particular, compare *sparse* graphs where $m = O(n)$ with *dense* graphs where $m = \Theta(n^2)$. How do things change if we consider a *multigraph*, like the graph of the Königsberg bridges, where there can be more than one edge between a pair of points?

Adjacency matrix need n^2 bits. Because there are n^2 pairs.

List of edges need $m \log n$ bits. Because each edge need to represent two nodes. And we need $\log n$ bits t specify integer $1 \sim n$.

	matrix(n^2)	list($m \log n$)
$m = O(n)$	n^2	$> n \log n$
$m = O(n^2)$	n^2	$< n^2 \log n$

If multigraph, matrix should represent number of edges. So, we need $\log m$ bits for each element. Because number of edge is at most m.

But, list nothing changes, so it is still $m \log n$

In multigraph

matrix: $n^2 \log m$, list: $m \log n$

3.22 Increasing subsequences. Given a sequence of integers $s_1, s_2, \ldots, s_n$, an *increasing subsequence* of length k is a series of indices $i_1 < i_2 < \cdots < i_k$ such that $s_{i_1} < s_{i_2} < \cdots < s_{i_k}$. For instance, the sequence

$$6, 3, 4, 8, 1, 5, 7, 2, 9$$

has an increasing subsequence of length 5, namely

$$3, 4, 5, 7, 9.$$

Even though there are an exponential number of possible subsequences, show that the problem of finding the longest one can be solved in polynomial time.

Let's use Dynamic Programming.

Let,

dp_i = Longest Increasing Subsequence ($\Rightarrow$ L.I.S) that last one is S_i

bef_i = index of directly before element of L.I.S that last one is S_i

And Let $dp_0 = 0$, $bef_0 = 0$. $S_0 = -\infty$

Now, $dp_i = \max_{j<i \text{ and } s_j<s_i} (1 + dp_j)$.

and $bef_i = j$, $j<i$ & $s_j < s_i$ & $dp_i = 1 + dp_j$

Now, By iterating i from 1 to n, compute dp_i and bef_i

and by tracing back from idx with the largest dp_{idx} to 0, using bef_i

We can obtain L.I.S with $O(N^2)$

3.25 Prune and conquer. Suppose I have a graph $G = (V, E)$ where each vertex v has a weight $w(v)$. An *independent set* is a subset S of V such that no two vertices in S have an edge between them. Consider the following problem:

MAX-WEIGHT INDEPENDENT SET
Input: A graph $G = (V, E)$ with vertex weights $w(v)$
Question: What is the independent set with the largest total weight?

Use dynamic programming to show that, in the special case where G is a tree, MAX-WEIGHT INDEPENDENT SET is in P.

Hint: once you decide whether or not to include the root of the tree in the set S, how does the remainder of the problem break up into pieces? Consider the example in Figure 3.26. Note that a greedy strategy doesn't work.

Many problems that are easy for trees seem to be very hard for general graphs, and we will see in Chapter 5 that MAX-WEIGHT INDEPENDENT SET is one of these. Why does dynamic programming not work for general graphs?

Let $dp(i,0)$:= value of max-weight independent set that does not contain i in the subtree that root is i

$dp(i,1)$:= value of max-weight independent set that contains i in the subtree that root is i

Let's start dfs from some node r, compute $dp(i,0)$ & $dp(i,1)$ $(1 \le i \le n)$

$$dp(i,0) = \sum_{j \in \text{child of } i} \max\left(dp(j,0), dp(j,1)\right)$$

$$dp(i,1) = w(i) + \sum_{j \in \text{child of } i} dp(j,0)$$

Answer is $\max\left(dp(r,0), dp(r,1)\right)$.

And If we recorded choices when compute dp, we can also find element of set.

Total Time complexity is $O(n+m) = O(n)$ $(\because m = n-1)$

Finally, In general graph, there might be cycle. So, we can't use dp approach as used in trees.

3.31 Faster Hamiltonian paths. Even when dynamic programming doesn't give a polynomial-time algorithm, it can sometimes give a better exponential-time one. A naive search algorithm for HAMILTONIAN PATH takes $n! \sim n^n e^{-n}$ time to try all possible orders in which we could visit the vertices. If the graph has maximum degree d, we can reduce this to $O(d^n)$—but d could be $\Theta(n)$, giving roughly n^n time again. Use dynamic programming to reduce this to a simple exponential, and solve HAMILTONIAN PATH in $2^n \operatorname{poly}(n)$ time. How much memory do you need for your solution? If the memory consumption worries you, check out Problem 3.32.

Let's define dp(i, state) := now we are node i, and state is bitmask that represent set of visited vertices. And value of dp(i, state) is 0 or 1. If 1, there is Hamiltonian path that satisfy definition of dp(i, state).

We know that there is n vertices and 2^n states, so we need $O(n \times 2^n)$ memory.

When we calculate each dp(i, state) we need to iterate n times to decide next destination. So, Time complexity is $O(n^2 \times 2^n)$.

$$dp(i, state) = \bigvee_{j \notin state}^{(bitwise)} dp(j, state | j)$$

And If state is $2^n - 1$, dp(i, state) is 1.

Finally, If dp(0, 0) is 1 there is hamiltonian path, else there isn't hamiltonian path.