Automata Theory HW3

20181593 계인혜

1.9

At first, the following conditions should be satisfied to find shortest cyclic tour of a graph that crosses every edge at least once.

1. Between any two nodes no more than one edge is added.
2. In any cycle of the extended graph, the total length of the added edge is not greater than half of the total length of the cycle.

The following are steps to solve the problem.

1. If the graph is an eulerian, that is, if the degree of all vertices is even, the Euler circuit will be the shortest cyclic tour.
2. If the graph is not an eulerian, that is, if the graph has vertices with an odd degree, then find all vertices with odd degrees.
    ① In a set S of vertices with odd degrees, list all possible pairs of odd vertices. However, except for pairs that are not represented by edges in the graph.
    ② For remaining all pairs, find the shortest path including this pair.
    ③ Find the miminum shortest path that includes the pair.
    ④ This pair is the edge that passes twice.

1.10

There is no way to find a Hamiltonian cycle in time proportional to any polynomial function of n yet.

Because Hamiltonian cycle is both, a NP-problem and NP-hard. That means NP-complete problem.

2.1

Suppose that it takes T years to do the simulation at this point.

Then, depending on the conditions in question, it takes t/2 + 2 years to execute the simulation after 2 years. This comes from the situation where the speed doubled and Pebble waited for 2 years.

To graduate earlier when waiting for the next upgrade, the following inequation has to be satisfied.

$$T > 2 + \frac{T}{2} \iff T > 4$$

Therefore, if the simulation takes longer than 4 years, Pebble can graduate faster when you wait for an upgrade.

2.18

- Given a graph G with n vertices and m edges, how many bits do we need to specify the adjacency matrix?

  Adjacent matrices are square matrices representing which vertices in the graph are connected to, and are represented by n^2 elements when there are n vertices. And, each element of the matrix consists of only 0 and 1, so 1 bit is required to each element.

  $$\therefore \ \boldsymbol{n^2} \textbf{ bits are required to specify the adjacency matrix.} \ / \ \boldsymbol{O(n^2)}$$

- How many do we need to specify a list of edges?

  The edges of the graph are expressed in the form of (a,b), where a and b mean a start point and an end point, respectively. Since $1 \leq a, b \leq n$, $\log n$ bits are required to represent each as given in the problem. Therefore, to express one edge, we need $2\log n$ bits. And we have m edges.

  $$\therefore \ \boldsymbol{2m \log n} \textbf{ bits are required to specify a list of edges.} \ / \ \boldsymbol{O(\text{nlog } n)}$$

- When are each of these two formats preferable?

  Adjacency matrix is advantageous for telling whether an edge exists between any two vertices and getting its weight. But there is a disadvantage that it consumes lot of memory on sparse graphs, and it has redundant information for undirected graphs.

  Edge list is advantageous for exploring all edges. However, there is a disadvantage that it is difficult to determine whether there is a path from a specific point to another point, and it is also difficult to determine the degree of each vertex.

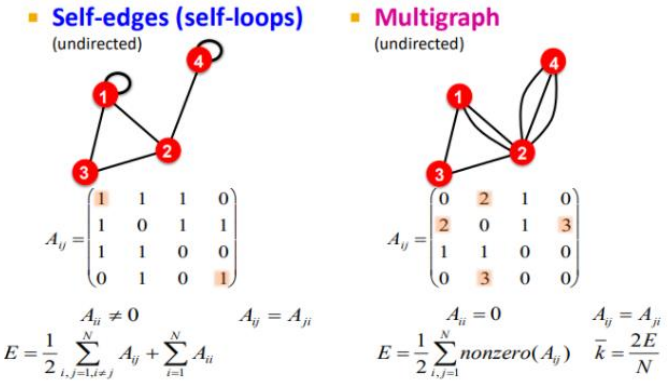
- Compare sparse graphs where $m = O(n)$ with dense graphs where $m = \theta(n^2)$.

Sparse graph is a graph in which the number of edges is much less than the possible number of edges. Sparse graph is advantageous for using precise storage requirements and simple to use and manage. But there is a disadvantage that it can't support topology changes, and it's inflexible.

On the other hand, **dense graph** can support topology changes and updates. But there is a disadvantage that it needs massive storage requirements, and it has a characteristic with poor locality.

- How do thing change if we consider a multigraph?

If the multigraph is represented as an adjacency matrix, each element of the adjacent matrix may have a value greater than 1 as shown in the figure below. If the maximum number of edges between two vertices is n, n^2 elements are required to construct an adjacent matrix, and $n^2 \log n$ bits are required because each element is represented using $\log n$ bits.



On the other hand, if the multigraph is represented as an edge list, then the number of bits used will vary greatly depending on the number of edges. When comparing the spatial complexity of the edge list and the spatial complexity of the adjacent matrix, If there is no limit to the maximum number of edges connecting the two vertices, it is advantageous to represent this multigraph as an adjacent matrix. However, if there is a limit to the maximum number, implementing it as an edge list could use fewer bits.

2.20

- Show that any *quasipolynomial* $f(n) = 2^{\theta(\log^k n)}$ with k>1 is $\omega(g(n))$ for any polynomial func g(n).

At first, $\log \frac{f(n)}{g(n)} = \theta(\log^k n) - O(\log n)$.

If you put the limit on both sides, then $\lim_{n\to\infty} \log\frac{f(n)}{g(n)} = \lim_{n\to\infty}\{\theta(log^k\, n) - O(\log\ n)\}$.

It's natural that $log^k\, n$ increases much faster than $\log\ n$, so the limit converges to ∞.

So, $\lim_{n\to\infty}\frac{f(n)}{g(n)} = \infty$.

On the other hand, there are positive constants C and k s.t $|f(n)| \geq C|g(n)|$ whenever n>k when f(n) is $\omega(g(n))$. Putting limits on both sides and transforming the expression is as follows.

$$\lim_{n\to\infty}\left|\frac{f(n)}{g(n)}\right| = \infty\ > \lim_{n\to\infty} C = C$$

$$\therefore f(n)\ is\ w(g(n)).$$

- Show that $f(n) = o(h(n))$ for any exponential function $h(n) = 2^{\theta(n^c)}$ for any c>0.

At first, $\log\frac{f(n)}{h(n)} = \theta(log^k\, n) - \theta(n^c)$.

If you put the limit on both sides, then $\lim_{n\to\infty} \log\frac{f(n)}{h(n)} = \lim_{n\to\infty}\{\theta(log^k\, n) - \theta(n^c)\}$.

It's natural that n^c increases much faster than $log^k\, n$, so the limit converges to $-\infty$.

So, $\lim_{n\to\infty}\frac{f(n)}{h(n)} = 0$.

On the other hand, there are positive constants C and k s.t $|f(n)| \leq C|h(n)|$ whenever n>k when f(n) is $O(h(n))$. Putting limits on both sides and transforming the expression is as follows.

$$\lim_{n\to\infty}\left|\frac{f(n)}{h(n)}\right| = 0\ < \lim_{n\to\infty} C = C$$

$$\therefore f(n)\ is\ O(h(n)).$$

- Show that the set of quasipolynomial functions is closed under composition.

Suppose that $f(n) = 2^{(\log n)^{c1}}, g(n) = 2^{(\log n)^{c2}}$.

Then, $f(g(n)) = f\left(2^{(\log n)^{c2}}\right) = 2^{(\log 2^{(\log n)^{c2}})^{c1}}$.

Based on the characteristic of the log, $f(g(n)) = 2^{((\log 2)\,(\log n)^{c2})^{c1}} = 2^{(\log 2)^{c1}(\log n)^{c1\cdot c2}}$.

Since $(\log 2)^{c1}$ is constant, it can be substituted to c3. So, $f(g(n)) = 2^{c3(\log n)^{c1\cdot c2}}$.

It's the form of quasipolynomial. Therefore, quasipolynomial functions is closed under composition.