第七章

7.6

Let L1 and L2 be two languages in P, and M1, M2 such that L(M1) = L1 and L(M2) = L2.

Union.

Construct TM M as follows.

On input (w):

1. Run M1 on w. If M1 accepts, M accepts.
2. Run M2 on w. If M2 accepts, M accepts.
3. reject.

Note that M accepts if either M1 or M2 does, thus deciding the union of the two languages. Also, Step 1 decides in polynomial time. Likewise Step 2 will be decided in polynomial time. Trivially, so will Step 3. Thus M decides in polynomial time.

Concatenation.

Construct TM M as follows:

On input (w):

1. for i = 0 to |w|
    1. Run M1 on w[0,i]
    2. Run M2 on w[i+1,|w|]
    3. If M1 and M2 accept, then accept
2. reject

Note that M accepts only if both halves of w, as split at some point i, are in their respective languages. That is, w[0,i], the substring of w from 0 to i is in L1 and the remainder of the string, w[i+1,|w|] is in L2. Only if there is no split of w in w1,w2 will TM M reject. Thus M decides the concatention of L1 and L2.

To show that M is in P, not that the inner loop steps each decide in polynomial time by construction. The summation of their execution times is thus in polynomial time, and the for loop will only increase this execution time by a factor. Thus M is in P.

Complement.

Construct TM M as follows:

On input(w):

1. Run M1 on w.
2. If M1 accepts w, reject. If M1 rejects, accept.

Here, M is outputs the opposite decider M1 does. Thus M decides the complement of M1. Almost trivially, Step 1 and 2 are both executed in polynomial time, and thus M is in P.

7.7 证明 NP 在并和连接运算下封闭。

(1) 并：

对任意 $L_1, L_2 \in NP$，设分别有 n^a 时间非确定图灵机 M_1 和 n^b 时间非确定图灵机 M_2 判定它们，且 c=max{a,b}。

构造判定 $L_1 \cup L_2$ 的非确定图灵机 M:

M=“对于输入字符串 w：

1) 在 w 上运行 M_1，在 w 上运行 M_2。

2) 若有一个接受则接受，否则拒绝。”

对于每一个非确定计算分支，第一步用时为 $O(n^a)+O(n^b)$，因此总时间为 $O(n^a+n^b)=O(n^c)$。 所以 $L_1 \cup L_2 \in NP$，即 NP 在并的运算下封闭。

(2) 连接：

对任意 $L_1, L_2 \in NP$，设分别有 n^a 时间非确定图灵机 M_1 和 n^b 时间非确定图灵机 M_2 判定它们，且 c=max{a,b}。

构造判定 $L_1 \circ L_2$ 的非确定图灵机 M:

M=“对于输入字符串 w：

1) 非确定地将分成两段 x,y，使得 w=xy。

2) 在 x 上运行 M_1，在 y 上运行 M_2。

3) 若都接受则接受，否则拒绝。”

对于每一个非确定计算分支，第一步用时 O(n),第二步用时为 $O(n^a)+O(n^b)$，因此总时间为 $O(n^a+ n^b)=O(n^c)$。 所以 $L_1 \circ L_2 \in NP$，即 NP 在连接运算下封闭。

7.9

To show that TRIANGLE is in P, consider the graph G represented as an ajacency matrix. The verticies become the column and row entries, and a 1 is marked if the two verticies are adjacent, a 0 if they are not.

For example, if <G> = {V,E} = { {a,b,c}, {ab, bc} }, then the adjacency matrix would look like:

	a	b	c
a	0	1	0
b	1	0	1
c	0	1	0

Now, given a matrix, representation, we can decide if the graph G has a triangle:

By construction, the matrix multiplication AxA = A^2 amounts to the dot products of each row in A with each column in A. Specifically, $A^2[x, y] = A[x, *] \cdot A[*, y]$. Additionally, by construction, a dot product is the sum of pair-wise multiplications. Given that the values of A contain only 1 (for edge presence) or 0 (for a missing edge), the multiplication of two cells in A amounts to the binary AND operation indicating the presence of the two edges corresponding to the two cells. The dot product $A[x, *] \cdot A[*, y]$ amounts to the logical sum (OR) of all of the combinations of edges containing x and y as endpoints, and occupies cell $A^2[x, y]$. Therefore, A^2 gives the combinations of 2 edges that form angles, with vertices x and y at the unclosed ends; $A^2[x, y]$ is read as "an angle with unclosed ends x and y".

Similarly, the matrix multiplication $AxA^2 = A^3$ combines each edge each angle. The dot product of A[x, *] with $A^2[*, y]$ gives the sum of all combinations of an edge containing vertex x with an angle with vertex y as an end point. Dot products including A[x, y] and $A^2[x, y]$ indicate the pairing of an edge ending at vertexes x and y with an angle having endpoints at vertexes x and y, which by definition is a triangle. According to the definition of matrix multiplication, such combinations occur only at $A^3[x, x]$, which is along the diagonal. All other pairings involve A[x, s] and $A^2[x, y]$, or A[s, x] and $A^2[x, y]$, or or A[t, u] and $A^2[x, y]$ … all of which are not triangles.

Groups of three edges that begin with one vertex, and end with the same vertex are by definition a triangle and are found at each $A^3[x, x]$. A cell on the diagonal having a value of 1 or more indicates the presence of a triangle.

Thus, to create a TM S that decides TRIANGLE,

S := on input (<G>)

1. create the adjacency matrix for the edges in G.
2. compute A^2 by multiplying A and A and write it after A on the tape.
3. compute A^3 by multiplying A and A^2 and write it after A^2 on the tape.
4. check the "diagonal" entries in A^3, if any are 1, accept.
5. otherwise, reject.

Each step above can be performed in polynomial time. Step 1 in O(n), Step 2 and 3 each in $O(n^3)$ and Step 4 O(n) in linear time. Thus S decides TRIANGLE in Ptime.

7.10 Show that ALLdfa is in P.

ALLdfa is the set of every language that is deterministically computed, in other words, there is always a finite number of steps to be computed and the machine never has to make any decisions, it always knows its next state. The class P is defined as a class of languages that are decidable in polynomial time on a deterministic single tape Turing machine. So, given a TM where we run an input string w, if it is accepted as decidable in polynomial time, then it is Deterministic and therefore in Class P. Modification of Theorem 4.1

M= “On input<B,w>, where B is a DFA and w is a string:

1.Simulate B on input w.

2.If the simulation ends in an accept state, accept. If it ends in a non-accepting state,reject.”

If input is accepted, then it is in class P, else it is not in class P.

7.11

To show that ISO is in NP, show that candidate isomorphic graphs G and H can be verified in polynomial time.

Thus, we are given some reordering of the nodes in G to G'so that they are identical to H. To verify that G' identical to H, verify each edge as follows:

Let L be a list of nodes that have been visited. Originally, this list is empty.

1. Let g be the first vertex on the tape in G
2. Let h in H be the candidate node equivalent to g.
3. If the degree of g is not equal to the degree of h, reject.
4. Add g and h to the list.
5. For each child g' of g not in the list.
    1. Let h' in H be the isomorphic vertex.
    2. If hh' is not in H, reject.
    3. Otherwise, execute Step 1 with the context of g'.
6. If each child of g is isomorphic to some child of h, then g and h are isomorphic.

Note that in the worst case, the recursion depth is the number of vertices in G. However, because only children not visited are considered at further depths, the algorithm halts and verifies that G is isomorphic to H in polynomial time.

7.12 证明 MODEXP∈P。

设二进制整数 b= $k_m k_{m-1}...k_1 k_0$，则 b=$k_0 2^0+k_1 2^1+k_2 2^2+...+k_m 2^m$($k_i$ =0,1)。构造判定 MODEXP 的图灵机如下：

M=“对于输入<a,b,c,p>，a,b,c,p 为二进制整数，

(1) 以 $k_0,k_1,...,k_n$ 记 b 的从低位到高位的 1 至 n+1 位。

(2) 令 d_0=a，对于 i=1,2,...,n 计算 $d_i=d_{i-1}\times d_{i-1}$ mod p。

(3) 对于 i=0,1,...,n，若 k_i=1，则令 $c_i=d_i$；若 k_i=0，则令 c_i=1。

(4) 计算 $e=c_0\times c_1\times ...\times c_n$ mod p。

(5) 若 e=c mod p，则接受，否则拒绝。”

设对于两个 n 位二进制整数，相乘再模一个至多 n 位的二进制整数，需要运行的时间为 T。则第二步的时间为 nT，第四步为 nT，所以总的运行时间为 O(nT)。这意味着 MODEXP∈P。

7.14 证明 P 在星号运算下封闭。

证明：设 M 为判定 A 的时间 f(n)图灵机，不妨设 f(n)≥n。设计如下 TM：

D=“对于输入 $y=y_1y_2...y_n$，

1) 若 y= ε ，则接受；
2) 对于 i,j＝1,2,...,n 重复(3)。
3) 在 $y_iy_{i+1}...y_j$ 上运行 M。若接受，则令 T(i, j)=“yes”。
4) 重复下面步骤直到表 T 不再改变。
5) 对于 i,j＝1,2,...,n 重复下面步骤。
6) 若 T(i,j)=“yes”，转(5)。否则继续。

7) 对于 k = i,i+1,…,j-1, 若 T(i,k)=T(k+1,j)=“yes”,则令 T(i,j)=“yes”。

8) 若 T(1, n)=yes，则接受；否则拒绝。

运行时间：设 $O(n^2f(n))+O(n^3)=O(n^2f(n))$。

7.15 证明 NP 在星号运算下封闭。

证明：设 M 为判定 A 的时间 f(n)非确定图灵机，不妨设 $f(n)\geq n$。设计如下 NTM：

D= “对于输入 $y=y_1y_2\ldots y_n$，

1) 若 $y=\varepsilon$ ，则接受；
2) 非确定地选择 y 的一个划分 $y=x_1x_2\ldots x_k$，其中是 $x_1,x_2,\ldots,x_k$ 字符串。
3) 对于 i=1,2,…,k，重复下一步骤：
4) 在 x_i 上运行 M，若 M 拒绝，则拒绝。
5) 若都接受，则接受。”