Automata Theory HW4

20181593 계인혜

3.19

Prove that for any l and any m>=1 we have $F_l = F_{m+1}F_{l-m} + F_mF_{l-m-1}$.

Base case is the following: If m = 1, $F_2F_{l-1} + F_1F_{l-2} = F_{l-1} + F_{l-2} = F_l$.

And we seek $F_l = F_{k+2}F_{l-k-1} + F_{k+1}F_{l-k-2}$ **when m = k+1.**

Suppose that for some $k \in \mathbb{N}$, where k≤ l-2 $F_l = F_{k+1}F_{l-k} + F_kF_{l-k-1}$.

Observe that $F_l = F_{k+1}F_{l-k} + F_kF_{l-k-1} = F_{k+1}(F_{l-k-1} + F_{l-k-2}) + F_kF_{l-k-1}$

$$= F_{k+1}F_{l-k-1} + F_{k+1}F_{l-k-2} + F_kF_{l-k-1} = F_{l-k-1}(F_k + F_{k+1}) + F_{k+1}F_{l-k-2}$$

$$= F_{k+2}F_{l-k-1} + F_{k+1}F_{l-k-2} \text{ .}$$

Thus, by principle of complete Induction $F_l = F_{m+1}F_{l-m} + F_mF_{l-m-1}$ for any l and any m>=1.

Prove that $F_{2l} = {F_l}^2 + 2F_lF_{l-1}$.

We will use $F_l = F_{m+1}F_{l-m} + F_mF_{l-m-1}$ to prove that $F_{2l} = {F_l}^2 + 2F_lF_{l-1}$.

Observe that $F_{2l} = F_lF_{l-1} + F_{l+1}F_l = F_lF_{l-1} + (F_{l-1} + F_l)F_l = {F_l}^2 + 2F_lF_{l-1}$.

So simply we can prove that $F_{2l} = {F_l}^2 + 2F_lF_{l-1}$.

Prove that $F_{2l+1} = {F_l}^2 + {F_{l+1}}^2$.

We will use $F_l = F_{m+1}F_{l-m} + F_mF_{l-m-1}$ again to prove that $F_{2l+1} = {F_l}^2 + {F_{l+1}}^2$.

Put m = k, l = 2k+1.

Observe that $F_{2k+1} = F_{k+1}F_{k+1} + F_kF_k = {F_k}^2 + {F_{k+1}}^2$.

So simply we can prove that $F_{2l} = {F_l}^2 + 2F_lF_{l-1}$.

Show that if l and p are n-bit numbers, we can calculate $F_l \, mod \, p$ in poly(n) time.

We can use a divide-and-conquer method here. Let T_l be the time for computing F_l. Then, by (3.21) we can derive the following when we ignore additions and multiplication : $T_{2l} = 2T_l$. So, T(l) is θ(l).

But as you know when we use recursion to get Fibonacci sequence, we have to calculate same things over and over. However, we can calculate $F_l \; and \; F_{l+1}$ simultaneously with the followings:

Using (3.21) we can reduce the problem $F_{2l} \; and \; F_{2l+1}$ **to** $F_l \; and \; F_{l+1}$ using some additions and multiplications with n-bit integers. Let f(n) be the time when we do this. Then, we can derive the equation that $f(n) = f(n-1) + o(n^2)$ because multiplication with n -bit integers is $o(n^2)$ and $F_{n-1} = F_{n+1} - F_n$. Therefore, T(n) = $o(n^3)$.

3.22

It is known by LIS(Longest Increasing Subsequence) algorithm. We can find the LIS in polynomial time using dynamic programming.

At first, we have to break down the problem into smaller sub-problems. And we need to store the solutions of the sub-problems and use them later.

We have two arrays, 'DP' and 'arr'. 'DP' is used for storing the sub-problems result and 'arr' is used for storing a given sequence.

And DP[i] means LIS value when arr[i] is the last element of an increasing sequence.

```
int LIS(int n){
    int i,j;
    int ans = 1;
    for(i=0;i<n;i++){
        dp[i] = 1;
        for(j=0;j<i;j++){
            if(arr[j] < arr[i] && dp[j]+1 > dp[i]){
                dp[i] = dp[j]+1;
                if(ans < dp[i]){
                    ans = dp[i];
                }
            }
        }
    }
    return ans;
}
```

There is a LIS code above. For n times, we fill the DP[i]. And the second For loop is finding a maximum DP[i] value.

For condition, 'arr[j] < arr[i]', it's the part that checks whether the value before i is less than the current i. And the 'dp[j]+1 > dp[i]' condition is necessary because the last incremental sequence with the jth element is DP[j], and when the last ith value is added at the end, it should be greater than the current maximum value.

So, if these two conditions are satisfied, then DP[i] is updated. It can be executed in polynomial time, **O(n^2).**

3.30

Let $G = (V, S, \Sigma, R)$ be a context-free grammar in Chomsky normal form. And specification of G is as follows :

V = {S,T,X},

Σ = {a,b},

R : S -> ε | AB | XB

T -> AB | XB

X -> AT

A -> a

B -> b

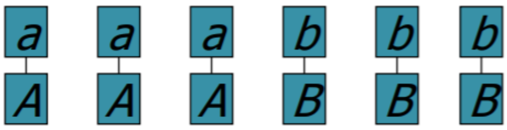
We can show this problem is in P using dynamic programming with CYK algorithm.

At first, pseudo code of CYK algorithm is as follows :

```
function CKY(word w, grammar G) returns table
  for i <- from 1 to LENGTH(W) do
     table[i-1,i] <- {A | A -> Wi∈G}
  for j <- from 2 to LENGTH(W) do
    for i <- from j-2 down to 0 do
      for k <- from i+1 to j-1 do
         table[i,j] <- table[i,j] ∪ {A | A->BC∈P, B∈table[i,k], C∈table[k,j]}
     |
   If the start symbol S ∈ table[0,n] then w ∈ L(G)
```

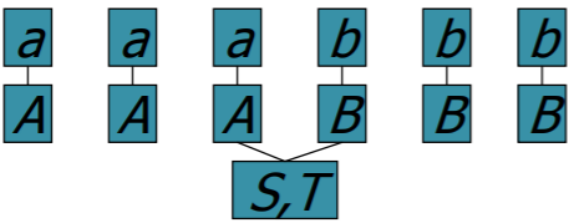
Let’s consider whether w = aaabbb is in L(G) starting with bottom of the tree.

1. Write variables for all length 1 substrings.



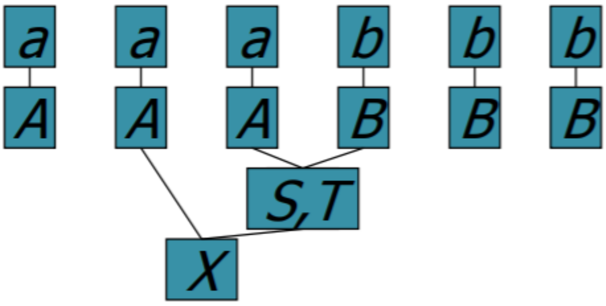
We use the rules **A->a and B->b.**

2. Write variables for all length 2 substrings.



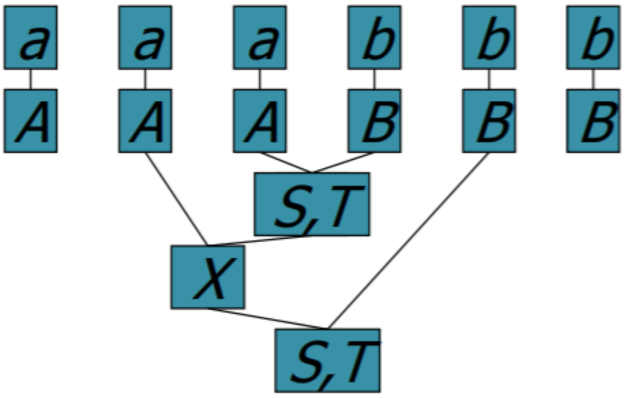
We use the rules **S->AB and T->AB.**

3. Write variables for all length 3 substrings



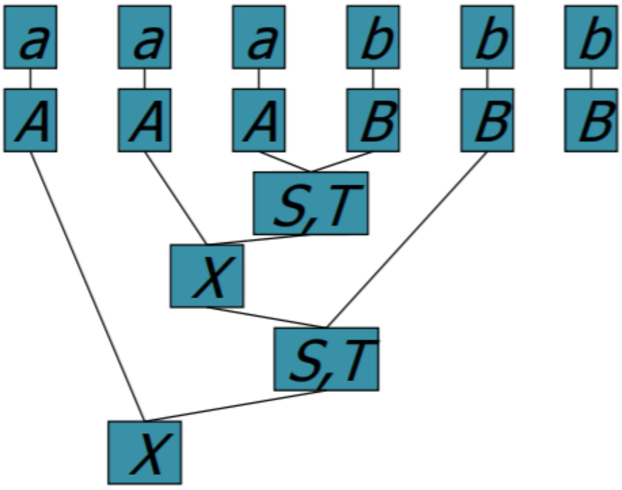
We use the rule **X->AT.**

4. Write variables for all length 4 substrings



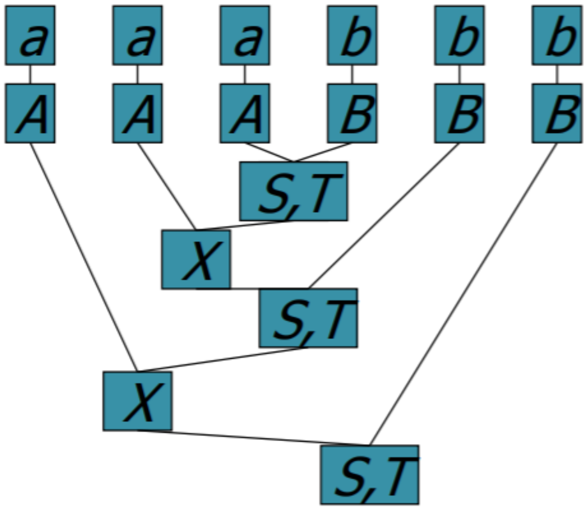
We use the rules **S->XB and T->XB.**

5. Write variables for all length 5 substrings



We use the rule **X->AT.**

6. Write variables for all length 6 substrings



We use the rule **S->XB and T->XB.**

Therefore, aaabbb is accepted.

Now let’s check the table chart used by the algorithm;

j / i	1 a	2 a	3 a	4 b	5 b	6 b
0	A	-	-	-	X	S,T
1		A	-	X	S,T	-
2			A	S,T	-	-
3				B	-	-
4					B	-
5						B

As you can see from above table, we keep the results for every w_{ij} in a table.

Note that we only need to fill in entries up to the diagonal.

And every element of the table can contain up to r=|N| symbols where N is the size of the non-terminal set.

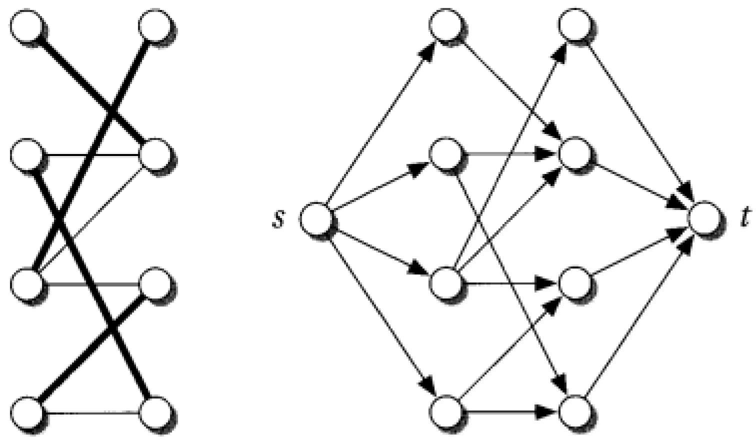
Finally we can calculate the time complexity of the algorithm.

Three nested for loop each one of O(n) size. And we have to lookup for N rules at each step.∴ $\boldsymbol{O(r^2n^3)}$ = $\boldsymbol{O(n^3)}$

Therefore, the problem is in P.

3.47

Prove Hall's theorem using the duality between Max Flow and Min Cut and the reduction in the picture below.



- At first, Hall theorem is as follows :

  A bipartite graph with n vertices on each side has a perfect matching ↔ Every subset S of the vertices on the left is connected to a set T of vertices on the right, where $|T| \geq |S|$.

<u>Pf. (→) Proof by contraposition</u>.

If there is a subset S on the left which is connected to a set T of vertices on the right, where $|T| < |S|$, then by the pigeonhole principle there is no way to find partners for all elements of S. And there is contradiction with the original proposition.

Therefore, if there is A bipartite graph with n vertices on each side has a perfect matching then every subset on the left is connected to a subset on the right, which is at least at large.

<u>Pf. (←)</u>

Suppose that there is no perfect matching and consider the above figure. Take the graph G, orient all of its edges so that they go from V1 to V2, add a source vertex s, a sink vertex t, edges from s to all of V1, from all of V2 to t, and let c be a capacity function that's identically 1 on all of the edges s->V1, t->V2, and infinite on all of the original edges in G. Then the MAX FLOW from s to t is less than n. And, by duality the MIN CUT is also less than n, so there is some cut consisting of c<n edges which separates s from t.

We claim that these edges might as well be among those leading out of s or into t in the figure, instead of in the middle layer of edges from the original bipartite graph. To see this, suppose the cut includes and edge (i,j) where i is a vertex of the left and j is a vertex of the right. This edge only blocks one path from s to t.(s->i->j->t) If we replace (i,j) with (s,i) or (j,t), we still block this path, and perhaps others as well.

So, there is a cut consisting of x edges leading from s to some set of vertices on the left, and y edges leading from some set of vertices on the right to t, where $x+y<n$. Call these sets X and Y respectively, where $|X|=x$, $|Y|=y$.

Now for these edges to block all paths from s to t, there must be no edges from $\bar{X}$ to $\bar{Y}$. Thus if $S=\bar{X}$, the set T of vertices that S is connected to is a subset of Y. But since $y<n-x$, we have $|T| \leq |Y| = y < n - x = |S|$. Hence, if there is no perfect matching, there is a subset on the left connected to a smaller subset on the right.