Name: WENTAI,CUI

CS381

Homework 7

Question 1:

1. For this questions, we need used DFS to go through the graph and check whether the graph is satisfying the requirement that is a dag having exactly one source and one sink.

Thus, we can create three Boolean values(one for check whether it is a dag graph one for check it has one source and one for check it has one sink) and firt one to true and the others to false.

We have three situation to check when we reach the new node:

First: whether it is not a dag graph

Second: whether it has one source

Third: whether it has one sink

After we finish the DFS, we just check whether these Boolean variances are true. For DFS are O(n) and three times check are O(3). Thus the total cost is O(n+3)—linear time.

ii)

1. We should first sort these adjacency lists by alphabetical. For this one, we should create 4 different sets for these vertices for 1-4, use the DFS it is very easy to finish the set 1 and set 3 just take O(1). And once we reach a new node we will check whether it match the condition 2, or 4 use the binary search and add it to the correct sets.

Question 2:

For this one we can use AVL tree. For AVL tree insert just take O(logn), we just need to insert the new element depend on the K[x]. Find the correct position and insert just O(logn). For extract-lr(remove) it is also take O(logn) to find and remove the corresponding node. Due to we create AVL tree depend on the value of K[x], thus it also take log(n) to find the node with the largest K[x].

Question 3:

1. As shown in the class we know that the lower bounds for sorting is O(nlogn), Thus for this question, set SE we should firstly sort both setA and setB, then check corresponding element one by one to see whether the return value is Yes or No. Thus the set SE is Omega-(nlogn).
2. For this one, we should modify the size of A and size of B, firstly we should create a new set that include all the elements of A and B. As what we did in the i) we should firstly sort the new set of c and compare the new set C with A to check if A and C has the exactly n-3 elements in common. Thus SE<O(n)S3E. And for the sort the lower bond is nlogn thus the lower bound is nlogn for the S3E.

Question 4:

We use the failure function to tell how long for a string that its suffix as also as the prefix. As show in the class slide for the Algorithm KMP

At we do the each iteration, we will check the Failure function. Thus for the n P = A N N A N O B A N N A N N A

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***j*** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| ***P*** | ***A*** | ***N*** | ***N*** | ***A*** | ***N*** | ***O*** | ***B*** | ***A*** | ***N*** | ***N*** | ***A*** | ***N*** | ***N*** | ***A*** |
| F | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 3 | 4 |

For the F(6) and F(12), the length that the number of prefix of P[0,j] compare with the suffix of p[1,j]. Thus we can see for the F[6] = 0 because there is mismatch at j[6] = B and for ther F[12] = 3 it is match for A N N.