118

0

COL351: Quiz-2
Maximum marks: 5+5 = 10

Name: VIRAJ AGASHE
Entry number: 2 0 2 0 C S 1 0 5 6 7

Question 1 Let $X, Y, Z \in \{0,1\}^n$ be three n-length strings. Describe an $O(n^2)$ time algorithm to compute largest k such that there exists a k-length string that is substring of X, Y, and Z.

~~Sol. We wish to find LCS of X, Y, Z.~~

$x_1\ x_2\ \dots\ x_n$
$y_1\ y_2\ \dots\ y_n$
$z_1\ z_2\ \dots\ z_n$

Sol. Algorithm:

Pass the string Y over the string X to find the set of all common substrings.

For each configuration, this takes $O(n-k)$ time, & n configurations possible.

$x_1\ x_2\ \dots\ x_n$

$y_1\ y_2\ \dots\ \underbrace{y_k\ \dots\ y_n}$

So overall it takes $O(n^2)$ time.

Further, for each of the substrings we get, we can treat it as a "pattern" & search for it in the string Z.

$O(n^3)$.

0

By KMP algorithm, we know we can do this in $O(n+k)$ time, where k is the size of the pattern.

$\therefore$ we get an algorithm which is

$O(n^2 + p\cdot(n+k))$

where p is the no. of common strings.

let dp[i] → answer of X[i, i+1 ⋯ n], Y[i, ⋯ n], Z[i, ⋯ n]

initialize dp[n] = (X[n] == Y[n] == Z[n])

for i in range(n-1, 1, -1): # backward interation

X1, Y1, Z1 = first_1(X, i), first_1(Y, i), first(Z, i) # O(n)

dp[i] = max(dp[i], 1 + dp[max(X1, Y1, Z1) + 1])

X0, Y0, Z0 = first_0(X, i), first_0(Y, i), first_(Z, i) # O(n)

dp[i] = max(dp[i], 1 + dp[max(X0, Y0, Z0) + 1])

→ T.C = O(n·n)

→ Can be improved to O(n) by precomputation

→ ans = dp[0]

Question 2 Let $G = (V, E)$ be a weighted digraph with no cycle of negative weight, and let $S \subseteq V$ be a set of size k. A path P is said to be an S-path if the internal vertices of P lie in S.

Describe an $O(kn^2)$ time algorithm to compute a binary matrix B such that $B[i, j] = 1$ if and only if there exists an S-path of ~~non~~-negative weight from vertex i to vertex j in G.

Ans. Note that if there $\exists$ a S path of -ve weight, ~~it must also be true~~ shortest S-path must also be of -ve weight.

i ... j

We modify Bellman-Ford as follows:

B[n][n] ← Initialize with 0

For each v ∈ V:

D = (d_1, d_2 ... d_n)

D[k] = ∞ ∀ k, D[v] = 0

For i in (1.... k):

~~if (D[v]~~ ~~for~~ ~~v~~ ~~in~~ ~~V~~ for all (v,w) ∈ S:

if (D[v] > D[w] + wt(v,w) and (v,w) ∈ S):

D[v] = D[w] + wt(v,w)

$\Big\}\ O(nk)$

After k iterations we will know if ~~there is~~ shortest path exists using only vertices in S

Now we can ~~con~~ fill up the matrix B.

for all k s.t. (D[k] < 0]:

B[v][k] = 1

$\Big\}\ O(n)$

$O(n^2k)$

Finally, return B

The inner loop runs in O(nk) times, and for each vertex we run it, so we get total time (once)

$= O(n^2k)$

$B[i,j]$ = Initialize with $c(i,j)$, if no edge, ∞

```
for i in V:          → # O(n)
    for j in V:      → # O(n)
        for k in S:  → # O(k)
            B[i,j] = min(B[i,j], B[i,k] + B[k,j])
                                        → O(1)

for i in V:
    for j in V:
        if B[i,j] < 0:
            B[i,j] = 1
        else:
            B[i,j] = 0
```

$\rightarrow O(n^2 k) + O(n^2)$

$= O(n^2 k)$