BT3051

End-Sem Exam

Ishan. Chokshi
BE19B018
24-11-2021
Ichokshi

Q.1] a) True.

As the computers store the value of ~~decimal~~ numbers in binary format, there are some numbers which cannot be represented precisely.

e.g:-

In python, 0.1 + 0.2 will not give 0.3 but a value like 0.30004

b) True.

Reason : $\Theta(g(n))$ is a tighter bound than $O(g(n))$

~~If $\Theta(g(n)$~~

$\therefore \Theta(g(n)): \quad c'g(n) \leq f(n) \leq c''g(n) \quad$ for $n \geq n_0$

$\implies f(n) \leq c''g(n)$

$\therefore$ [~~$\Omega(n)$~~ $O(g(n))$

c) ~~False~~ False

Reason: The difference in iterations is too less to 'guarantee' that the value will be closer to its actual value.

d) False

Expected final state: [None, None, None, None, None, 5, 6, 7, 8, 9]

Reason:

Step 1: Enqueue 0-9:

∴ Q = [0 1 2 3 4 5 6 7 8 9]

Step 2: Dequeing:

Q = [None None None None None 5 6 7 8 9]

Assumption: Q is not a circular queue.

e) False.

Selection sort is not adaptive, i.e, it does not speed up the sorting process if the array is already sorted.

Bubble sort can be made adaptive by few modifications in the its algorithm and thus will speed up the sorting process.

f) False.

Adjacency matrix uses a space of $O(n^2)$ although it can retrieve an edge, as well as delete or insert an edge in constant time. We can instead use an adjacency list which also has constant time for inserting or removing an edge.

g) True.

Even after multiplying the weights by a constant, all the weights will be scaled by the same factor. So shortest path will remain via same route as previous one

h) False.

In BFS we can use queue or recursion and not stack.

i) True.

Reason :- In dynamic programming, most of the sub problems are dependent. Hence, most of them will be calculated multiple times if recursion is used. This is highly inefficient. Instead, the values of the smaller sub problems should be stored in dynamic programming methods so those values are not calculated again.

j) True.

Reason: __add__ is a special method in python for addition or concatenation two variables but of the same datatype (depends on the datatype).

Syntax:

x.__add__(y)

It takes 'y' as a parameter, and performs the operation on 'x'.

If x and y are strings, the output will be: xy

Q.2) a) All options are correct because the value of the bit generated is random, ~~but between~~ i.e., either 0 or 1.

Since all options have either 0 or 1, all of them are correct

b)ii) $\sim N\lg(N)$ where $\lg N = \log_2 N$

Reason:

```
while (i<N):
{
    i*=2
}
```

$\leftarrow$ ~~$\log_2$~~ $\lg N$

```
for j in range (i):
    Sum+=i
```

$\leftarrow$ $N \times \lg N$ $= N\lg N$

Option (iv)

Q.2) c) i) 10110 1, 01001 0

iii) 10010 1, 01101 0

Reason: Parents : 10 1 : 0 1 0 ⟶ 10 1 10 1 } Children

01 0 : 1 01 01001 0 }

↑ ↑ Crossing over event

10 1 : 0 1 0 ⟶ 10010 1

01 0 : 1 01 01101 0

Q.3] a) `^t[^t]*$`

b) `^[^AEIOU]* [AEIOUaeiou] [^aeiou]* [aeiou] [^aeiou]* [aeiou] [^aeiou]* $`

c) `^[ATC]* TG* [ATC]*$`

d) `^([A-Za-z])([a-z])([a-z])([a-z])[a-z]\4\3\2\1$`

e) `^([ATGC][ATGC][ATGC])\1*$`

f) `^[^a]*(a[^a]*a[^a]*)*$`

g) `^[aeiouAEIOU][a-z]*[aeiou]$`

h) `^[A-Za-z]{20,}`

Q.4](a) t.* t$

steepest

strategist

throughput

transcendent

b) \<(.)(.).* \2\1 a\>

addenda

etiquette

c) a.* e * t

strategist

transcendent

d) ([din])@(.* \1){2}

addenda

transcendent

e) \< ([tea]) c.*)\1.* \1 o\>

anaconda

automata

eigenvalue

etiquette

Q.17]

	O	C	C	U	A	U	G	C
O	0	2	4	6	8	10	12	14
C	2	0	2	4	6	8	10	12
U	4	2	1	2	4	6	8	10
A	6	4	3	2	2	4	6	8
U	8	6	5	3	3	2	4	6
G	10	8	7	5	4	4	2	4
G	12	10	9	7	6	5	4	3
C	14	12	10	9	8	7	6	4

Final Edit distance

Q.16]

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Start from A:

A	B	C	D	E	F
0	3	1	14	∞	∞

CEdge Relaxations = 3)

Now include C:

A	B	C	D	E	F
0	2	1	12	∞	3

CEdge relaxations = 3)

Now ~~include~~ include B:

A	B	C	D	E	F
0	2	1	12	∞	3

CEdge relaxations = 0)

Now include F

A	B	C	D	E	F
0	2	1	12	15	3

CEdge Relaxations = 1)

Now ~~include~~ include D:

A	B	C	D	E	F
0	2	1	12	15	3

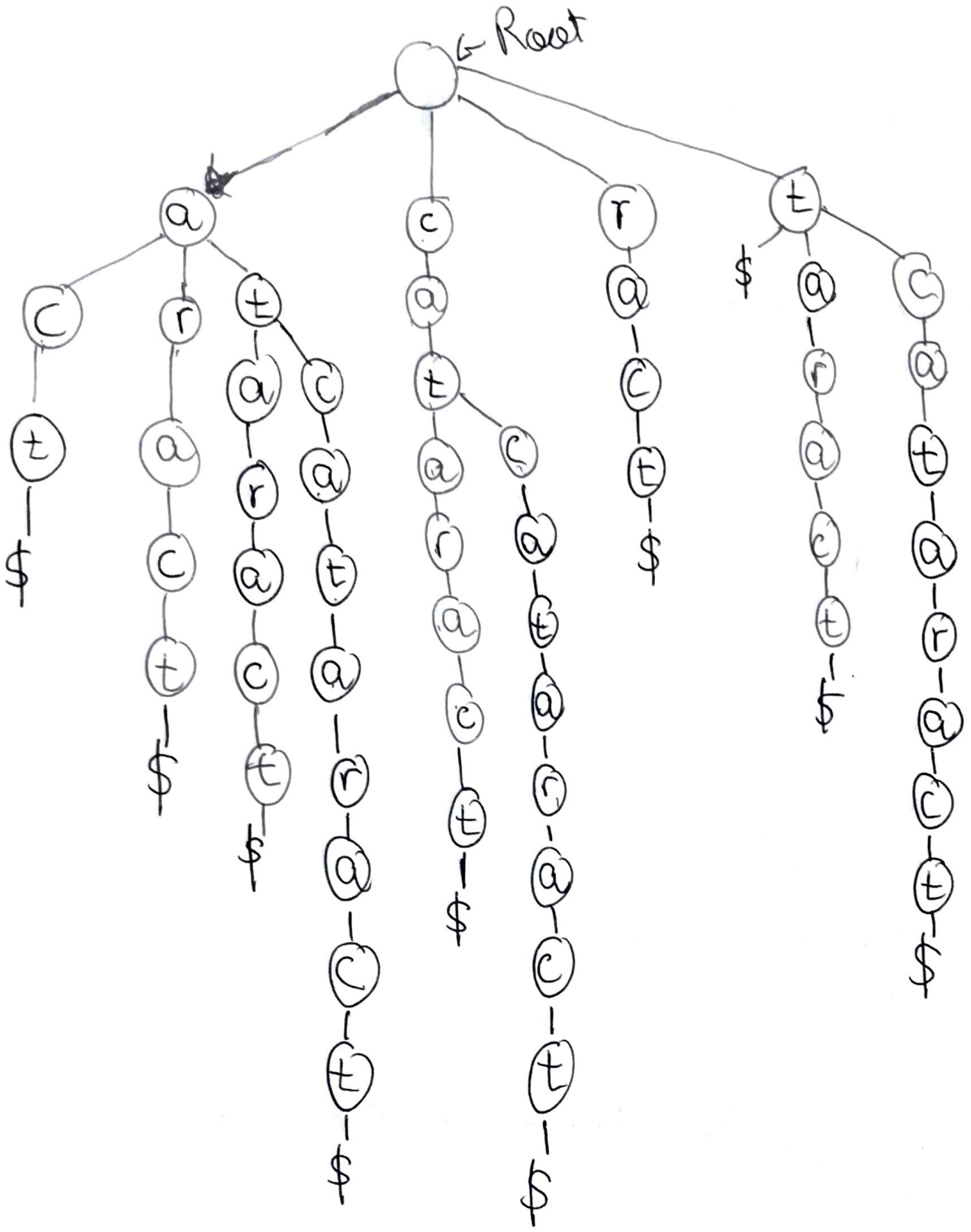
← Final distance values

Total edge relaxations = 37

Q.14) cat#cataract

t
ct
act
ract
aract
taract
ataract
cataract
tcataract
atcataract
catcataract

Suffix trie drawn on next page



Pattern : an

First, it will ~~search amongst the sta~~ go from root node to node 'a'.
Then it checks the child nodes of node 'a'.
Since no child node of 'a' has value 'n', the search will terminate and display that no pattern is found.

Q.15) We can use a modified edit distance method to find shortest evolutionary path between a pair of sequences S_a and S_b.

Let S_a be the source ~~string~~ RNA sequence

Let S_b be the target RNA sequence.

$$E(i,j) = \min \begin{cases} E(i-1, j) + 2 \\ E(i, j-1) + 2 \\ E(i-1, j-1) + 1 \end{cases}$$

But now we will apply an extra condition before calculating the distance.

The condition is:

$E(i-1, j)_a \in S$

$E(i, j-1) \in S$

$E(i-1, j-1) \in S$

~~Q.12]~~

~~def import math~~

Q.13]

```
import math
from numpy import random

def func(x):
    return math.sin(x)* math.cos(x)

def area (func, a, b):
    area = 0
    points = []
    for i in range(10000):
        x = random.uniform(a, b)
        y = random.uniform(0, 0.5)
        points.append((x,y))

    interior = 0
    for i in range (len(points)):
        if (points[i][1] - func(points[i][0]) <=0):
            interior = interior + 1

    area = (interior/10000) * (b - a) * 0.5
    return area
#Actual area of curve through Riemann integration = 0.5
#To estimate error:
area_curve = area (func, 0, math.pi/2)
```

```
error = 0.5 - area_curve
print (error)
print (area_curve)
```

Q.7) $$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}}$$

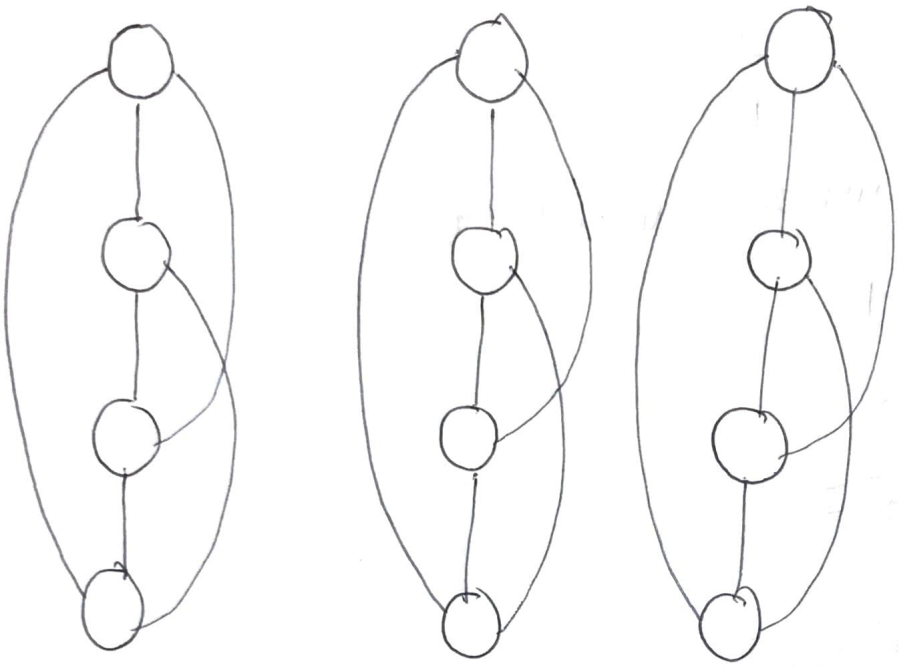
The above equation can be written as:

$$\varphi = 1 + \frac{1}{\varphi}$$

Python program:

```
g = 1
for i0 in range (100000):
    g = 1 + 1/g
print(g)
```

Q. 8)



Q. 7) Genome Assembly Problem is a problem where we have short sequences (say n sequences), then we have design an algorithm such that the 'n' sequences are aligned in such a way that it forms one single large genome sequence, in shortest time. In other words, we have to find a superstring 'S' which contains all the short sequences ' p_i ' as substrings. We can use hamiltonian method to solve this problem. All the substrings represent vertices of the graph. We can create a directed edge between overlapping vertices and then find the path that covers all the vertices at least once. The order in which the vertices are passed can be noted and combine them to form the genome. 18

Q. 9] A drawback of hill climbing algorithms is it can get stuck at a local minima or a local maxima. At these points, the ~~function~~ cost function can move in any direction which is a problem.
Method to overcome this: We can use simulated annealing.

We can use simulated annealing to overcome this issue. In this method, we don't reject moves that will increase the cost. We occasionally accept such moves with low probability. If this selected move then improves our cost ~~Q~~, ~~the~~ then we will ~~or~~ permanently accept this move.

Q.5]

```
k = int(input("Enter value of k"))
seq = input("Enter the sequence:")
kmers = []
for i in range(0, len(seq)-k+1):
    kmers.append(seq[i:i+k])

kmers_sorted = sorted(kmers)
for io in kmers_sorted:
    print(i)
```

Q.10] Currently, all intervals lie between 0 and 1

∴ Let the value be s.

∴ $0 < s < 1$

∴ Here, $b = 12$, $a = 1$

Multiplying by $(b-a)$ on both sides

$$0 < s(11) < 11$$

Adding a on both sides

$$\therefore 1 < 11s + 1 < 12$$

$$\therefore 1 < \lceil 11s \rceil + 1 < 12$$

⌈ ⌉ :- a like math.ceil

Now we use $\lceil 11s+1 \rceil$ for transformation

where 's' are the original probabilities

$s = (0.8147, 0.9058, 0.1270, 0.9134, 0.6324, 0.0975,$
$0.2785, 0.5469, 0.9575, 0.9649)$

∴ Applying the transformation $\lceil 11s \rceil + 1$,

New numbers: 10, 11, 3, 12, 8, 3, 5, 8, 12, 12

Q.12] Given: 1111000010 → let it be chromosome A

∴ x: 11110 y: 00010

∴ y = 2

x = 30

∴ Fitness function: $x^2 + y^3 - 5xy^2$

a) ∴ Fitness of A: $30^2 + 2^3 - 5 \times 30 \times 2^2$

= 308

b) New chromosome: 0111000011

∴ x = 01110

x = 14

y = 00011

y = 3

Fitness of B = $14^2 + 3^3 - 5 \times 14 \times 3^2$

= -407

Q.11) First generate a random ~~number~~ number within a sphere of radius 1:

∴ The x, y, z coordinates can be calculated as:

$$x = r\cos\theta\sin\phi \;,\; y = r\sin\theta\cos\phi \;,\; z = \cos\phi$$
$$\theta \in [0, 2\pi],\ \phi \in [0, \pi]$$

$$x = \text{sqrt}(\text{random}(0,1)) * \cos(\text{random}(0, 2\pi)) * \sin(\text{random}(0, \pi))$$

$$y = \text{sqrt}(\text{random}(0,1)) * \sin(\text{random}(0, 2\pi)) * \sin(\text{random}(0, \pi))$$

$$z = \text{sqrt}(\text{random}(0,1)) * \cos(\text{random}(0, \pi))$$

Now, we can scale them to the ellipsoid:

$$x\_ellipse = x * a$$
$$y\_ellipse = y * b$$
$$z\_ellipse = z * c$$