

Subject Name (Gujarati)

1333203 -- Winter 2024

Semester 1 Study Material

Detailed Solutions and Explanations

પ્રશ્ન 1(અ) [3 ગુણ]

રેખીય ડેટા સ્ટ્રક્ચર્સના નામ લખો.

જવાબ

રેખીય ડેટા સ્ટ્રક્ચર્સ

- એરે (Array)
- સ્ટેક (Stack)
- ક્વુ (Queue)
- લિંકડ લિસ્ટ (Linked List)

મેમરી ટ્રીક

"બધા વિદ્યાર્થીઓ લાઈનમાં ઊભા રહે છે"

પ્રશ્ન 1(બ) [4 ગુણ]

ટાઇમ અને સ્પેસ કોમ્પ્લેક્શનીટી વ્યાખ્યાયીત કરો.

જવાબ

કોમ્પ્લેક્શની પ્રકાર

વ્યાખ્યા

નોટેશન

ટાઇમ કોમ્પ્લેક્શની

માપે છે કે ઇનપુટ સાઇઝ વધતાં
એક્ઝિક્યુશન ટાઇમ કેવી રીતે વધે છે

$O(n)$, $O(1)$, $O(\log n)$

સ્પેસ કોમ્પ્લેક્શની

માપે છે કે ઇનપુટ સાઇઝ વધતાં મેમરી
વપરાશ કેવી રીતે વધે છે

$O(n)$, $O(1)$, $O(\log n)$

ડાયાગ્રામ:

1	INPUT SIZE	ALGORITHM	
2	(n)		
3	+-----+	+-----+	
4			
5		v v	
6		+-----+	
7		TIME	SPACE
8		O(n)	O(n)
9		+-----+	+-----+
0			

મેમરી ટ્રીક

"ટાઇમ સ્ટેપ્સ, સ્પેસ સ્ટોર્સ"

પ્રશ્ન 1(ક) [7 ગુણ]

ઉદાહરણ સાથે કલાસ અને ઓફજેક્ટનો કોન્સોટ સમજાવો.

જવાબ

ડાયાગ્રામ:

Student

-int rollNo

-string name

+setData()

+displayData()

કોન્સોટ	વ્યાખ્યા	ઉદાહરણ
કલાસ	ઓફજેક્ટ બનાવવા માટેનો બ્લૂપ્રિન્ટ અથવા ટેમ્પલેટ	Student કલાસ જેમાં properties (rollNo, name) અને methods (setData, displayData) છે student1 (rollNo=101, name="રાજ")
ઓફજેક્ટ	કલાસનું ચોક્કસ ડેટા ધરાવતું ઇન્સ્ટન્સ	

કોડ ઉદાહરણઃ

```

1 class Student:
2     def __init__(self):
3         self.rollNo = 0
4         self.name = ""
5
6     def setData(self, r, n):
7         self.rollNo = r
8         self.name = n
9
10    def displayData(self):
11        print(self.rollNo, self.name)
12
13#
14student1 = Student()
15student1.setData(101, " ")

```

મેમરી ટ્રીક

"કલાસ બનાવે, ઓફજેક્ટ વાપરે"

પ્રશ્ન 1(ક) OR [7 ગુણ]

વિદ્યાર્થીઓના રેકોર્ડ્સ ને સંચાલિત કરવા માટેનો એક કલાસ બનાવો જેમા વિદ્યાર્થીને ઉમેરવા તેમજ બાદ કરવા માટેની મેથડ હોય.

જવાબ

ડાયાગ્રામ:

StudentManager

- Student[] students
- int count

- +addStudent()
- +removeStudent()
- +displayAll()

sol:

```
1 class StudentManager:  
2     def __init__(self):  
3         self.students = []  
4  
5     def addStudent(self, roll, name):  
6         student = Student()  
7         student.setData(roll, name)  
8         self.students.append(student)  
9  
10    def removeStudent(self, roll):  
11        for i in range(len(self.students)):  
12            if self.students[i].rollNo == roll:  
13                self.students.pop(i)  
14                break  
15  
16    def displayAll(self):  
17        for student in self.students:  
18            student.displayData()
```

મેમરી ટ્રીક

"ઓપરેશન વિધાન, કાઢો ઘટાડો"

પ્રશ્ન 2(અ) [3 ગુણ]

કલાસમાં કન્સ્ટ્રક્ટરનું મહત્વ સમજાવો.

જવાબ

કન્સ્ટ્રક્ટરનું મહત્વ

- ઓપરેશન ડેટા મેમ્બરને પ્રારંભિક મૂલ્ય આપે છે
- ઓપરેશન બનતી વખતે આપોઆપ કોલ થાય છે
- અલગ અલગ પ્રકારના હોઈ શકે (ડિફોલ્ટ, પેરામીટરાઈઝડ, કોપી)

મેમરી ટ્રીક

"શરૂઆત હેઠળ સારી"

પ્રશ્ન 2(બ) [4 ગુણ]

સ્ટેક પર વિવિધ ઓપરેશન સમજાવો.

જવાબ

ઓપરેશન	વર્ણન	ઉદાહરણ
પુશ (Push)	ટોપ પર એલિમેન્ટ ઉમેરે છે	push(5)
પોપ (Pop)	ટોપ પરથી એલિમેન્ટ દૂર કરે છે	x = pop()
પીક/ટોપ (Peek/Top)	ટોપ એલિમેન્ટને દૂર કર્યા વગર જુઓ છે	x = peek()
isEmpty	ચકાસે છે કે સ્ટેક ખાલી છે કે નહીં	if(isEmpty())

ડાયાગ્રામ:

1	PUSH	POP
2		~
3	v	
4	+---+	+---+
5	5	8
6	+---+	+---+
7	7 PEEK/TOP ----->	7
8	+---+	+---+
9	8	2
0	+---+	+---+

મેમરી ટ્રીક

"નાખો કાઢો જુઓ"

પ્રશ્ન 2(ક) [7 ગુણ]

પોસ્ટફિક્સ એક્સપ્રેશન ABC+*D/ નું મૂલ્યાંકન અલગોરિધમનું વર્ણન કરો.

જવાબ

ડાયગ્રામ:

```

1 Input: A B C + * D /
2
3 +---+---+---+---+---+---+
4 | A | B | C | + | * | D | / |   |
5 +---+---+---+---+---+---+
6     Read left to right

```

સ્ટેપ	સિમ્બોલ	એક્શન	સ્ટેક
1	A	સ્ટેક પર પુશ કરો	A
2	B	સ્ટેક પર પુશ કરો	A,B
3	C	સ્ટેક પર પુશ કરો	A,B,C
4	+	B,C પોપ કરો; B+C પુશ કરો	A,B+C
5	*	A,B+C પોપ કરો; A*(B+C) પુશ કરો	A*(B+C)
6	D	સ્ટેક પર પુશ કરો	A*(B+C),D
7	/	A(B+C),D પોપ કરો; A(B+C)/D પુશ કરો	A*(B+C)/D

મેમરી ટ્રીક

“વાંચો, પુશ કરો, પોપ કરો, ગણતરી કરો”

પ્રશ્ન 2(અ) OR [3 ગુણ]

સ્ટેક અને ક્યુ વચ્ચેનો તફાવત લખો.

જવાબ

કીચર	સ્ટેક	ક્યુ
સિદ્ધાંત	LIFO (છેલ્લું આવે પહેલું જાય)	FIFO (પહેલું આવે પહેલું જાય)
આપરેશન	પુશ/પોપ	એનક્યુ/ડિક્યુ
એક્સેસ પોઇન્ટ્સ	એક છેડો (ટોપ)	બે છેડા (ફાન્ટ, રીઅર)

મેમરી ટ્રીક

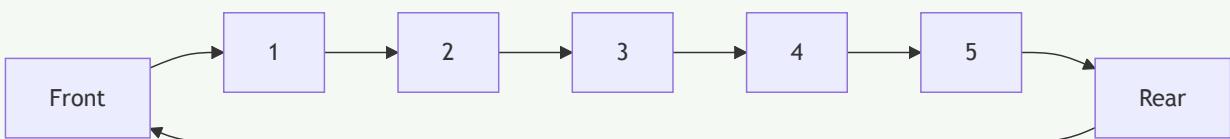
“સ્ટેક છેલ્લું પહેલું, ક્યુ પહેલું પહેલું”

પ્રશ્ન 2(બ) OR [4 ગુણ]

સર્કુલર ક્યુ નો કોન્સેપ્ટ સમજાવો.

જવાબ

ડાયગ્રામ:



કીચર	વર્ણન
સ્ટ્રક્ચર	છેડાઓ જોડાયેલ હોય તેવો લીનિયર ડેટા સ્ટ્રક્ચર
ફાયદો	ખાલી જગ્યાનો ફરીથી ઉપયોગ કરીને મેમરી કાર્યક્ષમ રીતે વાપરે છે

મેમરી ટ્રીક

"સર્કુલર ફન્ટને રીઅર સાથે જોડે"

પ્રશ્ન 2(ક) OR [7 ગુણ]

સિંગલી લિંકડ લિસ્ટમાં આપેલ નોડ પછી અને પહેલાં નવા નોડ દાખલ કરવાની પ્રક્રિયાનું વર્ણન કરો.

જવાબ

ડાયાગ્રામ:

```

1 Insert After Node X:
2 Before: A \rightarrow X \rightarrow B
3 After: A \rightarrow X \rightarrow N \rightarrow B
4
5 Insert Before Node X:
6 Before: A \rightarrow X \rightarrow B
7 After: A \rightarrow N \rightarrow X \rightarrow B
    
```

ઇન્સર્શન

નોડ X પછી

નોડ X પહેલાં

સ્ટેપ્સ

1. નવો નોડ N બનાવો2. N નો next X ના next પર સેટ કરો3. X નો next N પર સેટ કરો
1. નવો નોડ N બનાવો2. X પર પોઇન્ટ કરતો નોડ A શોધો3. N નો next X પર સેટ કરો4. A નો next N પર સેટ કરો

મેમરી ટ્રીક

"પછી: લિંક બદલો, પહેલાં: અગાઉનો શોધો"

પ્રશ્ન 3(અ) [3 ગુણ]

લિંકડ લિસ્ટ માં એક છેડાથી બીજા છેડા સુધી પસાર થવાની પ્રક્રિયા સમજાવો.

જવાબ

ડાયાગ્રામ:

```

1 start \rightarrow [10] \rightarrow [20] \rightarrow [30] \rightarrow NULL
2           ^   ^   ^
3           |   |   |
4      Visit  Visit  Visit
    
```

સ્ટેપ	એક્શન
1	હેડ નોડથી શરૂ કરો
2	વર્તમાન નોડનો ડેટા એક્સેસ કરો
3	પોઇન્ટરને આગામના નોડ પર ખસેડો
4	NULL મળે ત્યાં સુધી દોહરાવો

મેમરી ટ્રીક

"શરૂ કરો, જુઓ, આગામ વધો, દોહરાવો"

પ્રશ્ન 3(બ) [4 ગુણ]

ઇનફિક્સથી પોસ્ટફિક્સમાં એક્સપ્રેસનનું રૂપાંતર સમજાવો.

જવાબ

ડાયગ્રામ:

```

1 Infix: A + B * C
2 Postfix: A B C * +

```

સ્ટેપ	એક્ષન	સ્ટેક	આઉટપુટ
1	ડાબેથી જમણે સ્કેન કરો		
2	જો ઓપરેન્ડ હોય, તો આઉટપુટમાં ઉમેરો	A	
3	જો ઓપરેટર હોય, તો ઉચ્ચ પ્રાધાન્યતા હોય તો પુશ કરો	+	A
4	ઓછી પ્રાધાન્યતાવાળા ઓપરેટર પોપ કરો	+	A B
5	વર્તમાન ઓપરેટર પુશ કરો	*	A B
6	એક્સપ્રેશન પૂરું થાય ત્યાં સુધી ચાલુ રાખો	*	A B C
7	બાકીના ઓપરેટર પોપ કરો		A B C * +

મેમરી ટ્રીક

“ઓપરેટર પુશ-પોપ, ઓપરેન્ડ સીધા આઉટપુટમાં”

પ્રશ્ન 3(ક) [7 ગુણ]

સિંગલી લિંકડ લિસ્ટની શરૂઆતનો અને અંતનો નોડ ડિલીટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

ડાયગ્રામ:

```

1 Before: Head \rightarrow [10] \rightarrow [20] \rightarrow [30] \rightarrow NULL
2 After: Head \rightarrow [20] \rightarrow NULL

```

કોડ:

```

1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
5
6 class LinkedList:
7     def __init__(self):
8         self.head = None
9
10    def deleteFirst(self):
11        if self.head is None:
12            return
13        self.head = self.head.next
14
15    def deleteLast(self):
16        if self.head is None:
17            return
18
19        #
20        if self.head.next is None:
21            self.head = None
22            return
23
24        temp = self.head
25        while temp.next.next:
26            temp = temp.next
27

```

28 | temp.next = None

મેમરી ટ્રીક

“પહેલો: હેડ શિફ્ટ કરો, છેલ્લો: પાછલો શોધો”

પ્રશ્ન 3(અ) OR [3 ગુણ]

લિંકડ લિસ્ટમાં કોઇ એલિમેન્ટ શોધવાની પ્રક્રિયા સમજાવો.

જવાબ

ડાયાગ્રામ:

```

1 Head → [10] → [20] → [30] → NULL
2           ^   ^   ^
3       Check Check Check
  
```

સ્ટેપ	વર્ણન
1	હેડ નોડથી શરૂ કરો
2	વર્તમાન નોડના ડેટાને કી સાથે સરખાવો
3	જો મેચ મળે, તો true રીટર્ન કરો
4	નહીંતર, આગળના નોડ પર જાઓ અને રિપીટ કરો

મેમરી ટ્રીક

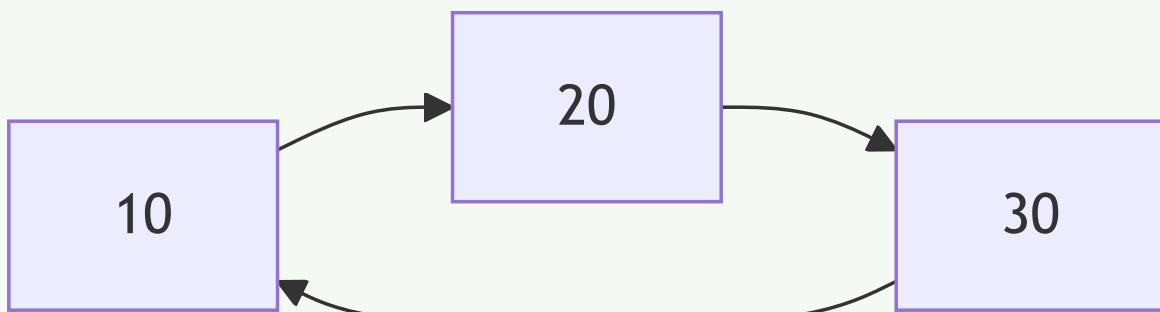
“શરૂ કરો, ચેક કરો, આગળ વધો, દોહરાવો”

પ્રશ્ન 3(બ) OR [4 ગુણ]

સર્કૃતુલર લિંકડ લિસ્ટ નો કોન્સોટ સમજાવો.

જવાબ

ડાયાગ્રામ:



ફીચર	વર્ણન
સ્ટ્રક્ચર	છેલ્લો નોડ પહેલા નોડને પોઇન્ટ કરે છે
ફાયદો	NULL પોઇન્ટર્સ નથી, સર્કૃતુલર ઓપરેશન માટે કાર્યક્ષમ
ટ્રાવર્સલ	અનંત લૂપ ટાળવા માટે વધારાની શરત જરૂરી

મેમરી ટ્રીક

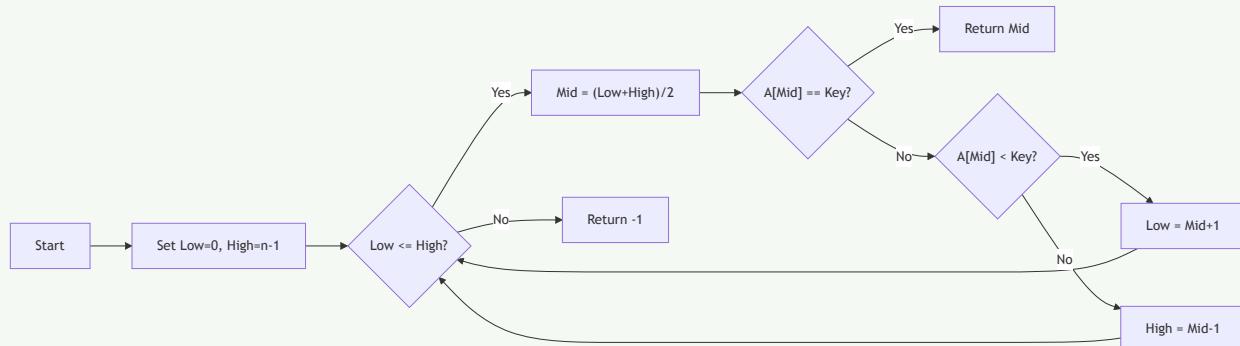
“છેલ્લો પહેલાને જોડો”

પ્રશ્ન 3(ક) OR [7 ગુણ]

લિસ્ટમાંથી બાઇનરી સર્ચનો ઉપયોગ કરીને કોઇ એક એલિમેન્ટ શોધવાનું અલગોરિધમ સમજાવો.

જવાબ

ડાયગ્રામ:



કોડ:

```

1 def binarySearch(arr, key):
2     low = 0
3     high = len(arr) - 1
4
5     while low <= high:
6         mid = (low + high) // 2
7
8         if arr[mid] == key:
9             return mid
10        elif arr[mid] < key:
11            low = mid + 1
12        else:
13            high = mid - 1
14
15    return -1
  
```

મેમરી ટ્રીક

“મદય, તુલના, અડધું કાઢો”

પ્રશ્ન 4(અ) [3 ગુણ]

લિંકડ લિસ્ટના ઉપયોગ લખો.

જવાબ

લિંકડ લિસ્ટના ઉપયોગ

1. સ્ટેક અને ક્યુનો અમલીકરણ
2. ડાયનામિક મેમરી એલોકેશન
3. ઇમેજ વ્યૂઅર (આગલી/પાછલી ઇમેજ)

મેમરી ટ્રીક

“ડેટા ડાયનામિક સ્ટોર કરો”

પ્રશ્ન 4(બ) [4 ગુણ]

સિંગલી અને ડબલી લિંકડ લિસ્ટ વચ્ચેનો તફાવત લખો.

જવાબ

ફીચર	સિંગલી લિંકડ લિસ્ટ	ડબલી લિંકડ લિસ્ટ
નોડ સ્ટ્રક્ચર	એક પોઇન્ટર (next)	બે પોઇન્ટર (next, prev)
ટ્રાવર્સલ	માત્ર ફોરવર્ડ	બંને દિશામાં
મેમરી	ઓછી મેમરી	વધુ મેમરી
ઓપરેશન	સરળ, ઓછો કોડ	જटિલ, વધુ ફ્લેક્સિબલ

ડાયાગ્રામ:

1 Singly: [Data|Next] → [Data|Next] → [Data|Next]
 2 Doubly: [Prev|Data|Next] → [Prev|Data|Next] → [Prev|Data|Next]

મેમરી ટ્રીક

“એક દિશા, બે દિશા”

પ્રશ્ન 4(ક) [7 ગુણ]

સિલેક્શન સોર્ટ અલગોરિધમનો ઉપયોગ કરીને આંકડાઓને ચઢતા ક્રમમાં ગોઠવવાનો પ્રોગ્રામ લખો.

જવાબ

ડાયાગ્રામ:

1 Initial: [5, 3, 8, 1, 2]
 2 Pass 1: [1, 3, 8, 5, 2] (Swap 5,1)
 3 Pass 2: [1, 2, 8, 5, 3] (Swap 3,2)
 4 Pass 3: [1, 2, 3, 5, 8] (Swap 8,3)
 5 Pass 4: [1, 2, 3, 5, 8] (No swap)

કોડ:

```
1 def selectionSort(arr):
2     n = len(arr)
3
4     for i in range(n):
5         min_idx = i
6
7         for j in range(i+1, n):
8             if arr[j] < arr[min_idx]:
9                 min_idx = j
10
11         #
12         arr[i], arr[min_idx] = arr[min_idx], arr[i]
13
14     return arr
15
16 #
17 arr = [5, 3, 8, 1, 2]
18 sorted_arr = selectionSort(arr)
19 print(sorted_arr) # : [1, 2, 3, 5, 8]
```

મેમરી ટ્રીક

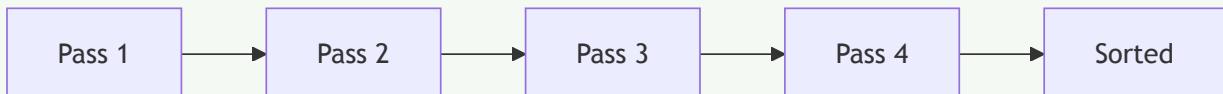
“મિનિમમ શોધો, પોર્ઝિશન બદલો”

પ્રશ્ન 4(અ) OR [3 ગુણ]

બબલ સોર્ટ અલગોરિધમ સમજાવો.

જવાબ

ડાયગ્રામ:



મુખ્ય પોઇન્ટ્સ

આસપાસના એલિમેન્ટની તુલના કરો
જો ખોટા કમમાં હોય તો સ્વેપ કરો
દરેક પાસમાં મોટા એલિમેન્ટ છેવટે પહોંચો

મેમરી ટ્રીક

“મોટા બબલ ઉપર જાય”

પ્રશ્ન 4(બ) OR [4 ગુણ]

લિનિયર અને બાઇનરી સર્ચ વર્ષેનો તફાવત લખો.

જવાબ

ફીચર	લિનિયર સર્ચ	બાઇનરી સર્ચ
કાર્ય સિદ્ધાંત	કમિક ચકાસણી	વિભાજન અને જીત
ટાઇમ કોમ્પ્લેક્સિટી	$O(n)$	$O(\log n)$
ડેટા અરેન્જમેન્ટ	અનસોર્ટેડ અથવા સોર્ટેડ	સોર્ટેડ હોવું જરૂરી
શેના માટે સારં	નાના ડેટાસેટ	મોટા ડેટાસેટ

મેમરી ટ્રીક

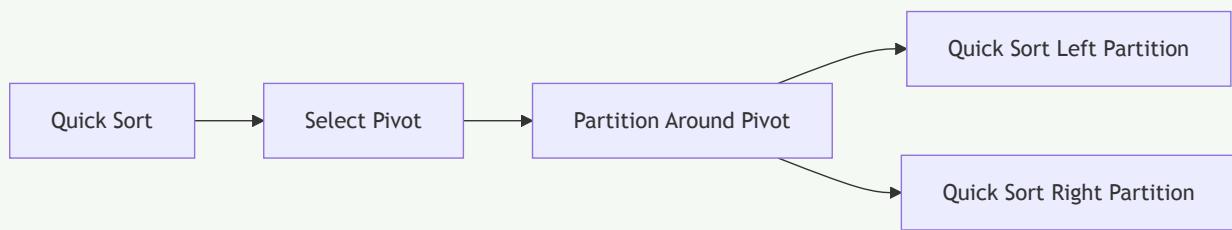
“લિનિયર બધાને જુઓ, બાઇનરી અડધું કાપે”

પ્રશ્ન 4(ક) OR [7 ગુણ]

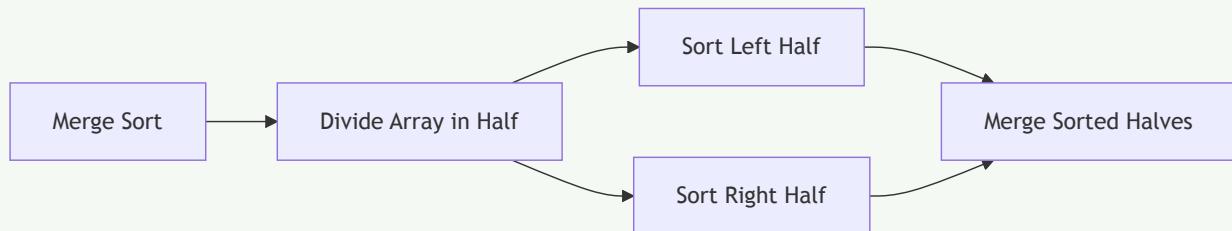
કવીક સોર્ટ અને મર્જ સોર્ટ સમજાવો.

જવાબ

કવીક સોર્ટ:



મર્જ સોટ:



અલગોરિધમ	સિદ્ધાંત	સરેરાશ ટાઇમ	સ્પેસ કોમ્પ્લેક્સિટી
ક્વીક સોટ	પીવોટની આસપાસ પાર્ટિશનિંગ	$O(n \log n)$	$O(\log n)$
મર્જ સોટ	વિભાજન, જુત, જોડાણ	$O(n \log n)$	$O(n)$

મેમરી ટ્રીક

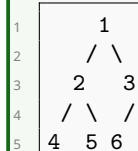
“ક્વીક વિભાજે, મર્જ જોડે”

પ્રશ્ન 5(અ) [3 ગુણ]

પૂર્ણ બાઇનરી ટ્રી ની વ્યાખ્યા આપો.

જવાબ

ડાયાગ્રામ:



પ્રોપર્ટી	વર્ણન
બધા લેવલ ભરેલા	છેલ્લા લેવલ સિવાય
છેલ્લુ લેવલ ડાબેથી ભરેલું	નોડ ડાબેથી જમણે એડ થાય

મેમરી ટ્રીક

“ડાબેથી જમણે, લેવલ દર લેવલ ભરો”

પ્રશ્ન 5(બ) [4 ગુણ]

બાઇનરી ટ્રી મા ઇનઓર્ડર ટ્રાવર્સલ સમજાવો.

જવાબ

ડાયાગ્રામ:



```

2      / \
3      B   C
4      / \
5      D   E
6
7 Inorder: D \rightarrow B \rightarrow E \rightarrow A \rightarrow C

```

સ્ટેપ	એક્શન
1	ડાબા સબટ્રી પર ટ્રાવર્સ કરો
2	રૂટ નોડની મુલાકાત લો
3	જમણા સબટ્રી પર ટ્રાવર્સ કરો

કોડ:

```

1 def inorderTraversal(root):
2     if root:
3         inorderTraversal(root.left)
4         print(root.data, end=" \rightarrow ")
5         inorderTraversal(root.right)

```

મેમરી ટ્રીક

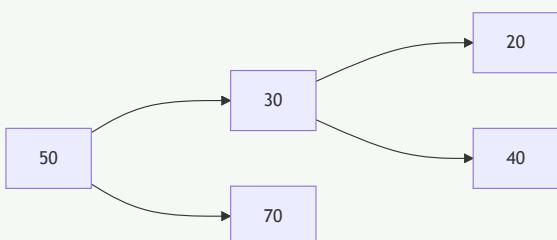
“ડાબુ, રૂટ, જમણુ”

પ્રશ્ન 5(ક) [7 ગુણ]

બાઇનરી સર્ચ ટ્રી મા નોડ દાખલ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

ડાયાગ્રામ:



કોડ:

```

1 class Node:
2     def __init__(self, key):
3         self.key = key
4         self.left = None
5         self.right = None
6
7     def insert(self, key):
8         if self.key is None:
9             return Node(key)
10
11         if key < self.key:
12             self.left = insert(self.left, key)
13         else:
14             self.right = insert(self.right, key)

```

```
6     return root
```

મેમરી ટ્રીક

"તુલના કરો, મૂવ કરો, દાખલ કરો"

પ્રશ્ન 5(અ) OR [3 ગુણ]

બાઇનરી સર્ચ ટ્રીની મૂળભૂત ખાસિયતો જણાવો.

જવાબ

બાઇનરી સર્ચ ટ્રીની ખાસિયતો

1. ડાબા ચાઈલ્ડ નોડ < પેરેન્ટ નોડ
2. જમણા ચાઈલ્ડ નોડ > પેરેન્ટ નોડ
3. ડુલિકેટ વેલ્યુ માન્ય નથી

મેમરી ટ્રીક

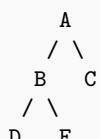
"ડાબે ઓછું, જમણો વધુ"

પ્રશ્ન 5(બ) OR [4 ગુણ]

બાઇનરી ટ્રી મા પોસ્ટ ઓર્ડર ટ્રાવર્સલ સમજાવો.

જવાબ

ડાયાગ્રામ:



Postorder: D → E → B → C → A

સ્ટેપ	એક્શન
1	ડાબા સબટ્રી પર ટ્રાવર્સ કરો
2	જમણા સબટ્રી પર ટ્રાવર્સ કરો
3	રૂટ નોડની મુલાકાત લો

કોડ:

```

1 def postorderTraversal(root):
2     if root:
3         postorderTraversal(root.left)
4         postorderTraversal(root.right)
5         print(root.data, end=" \rightarrow ")

```

મેમરી ટ્રીક

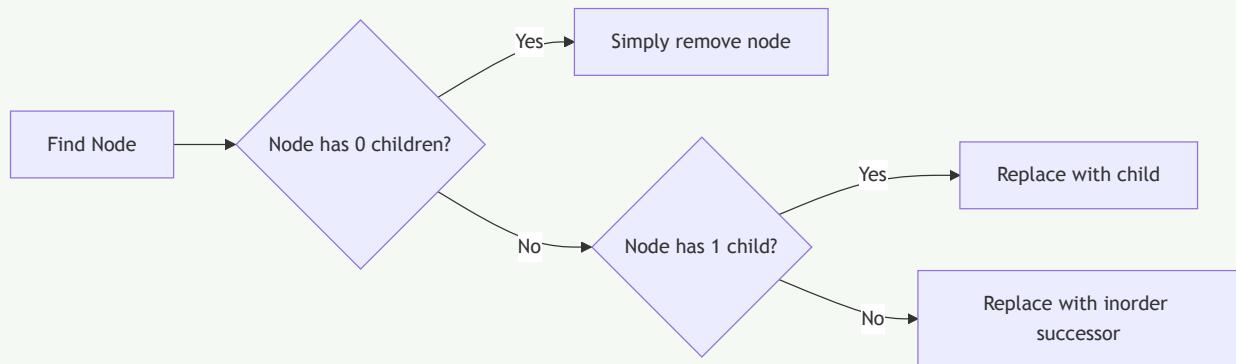
"ડાબું, જમણું, રૂટ"

પ્રશ્ન 5(ક) OR [7 ગુણ]

બાઇનરી સર્ટ ટ્રી માંથી નોડ ડિલિટ કરવા માટેનો પ્રોગ્રામ લખો.

જવાબ

ડાયગ્રામ:



કોડ:

```
1 def minValueNode(node):
2     current = node
3     while current.left is not None:
4         current = current.left
5     return current
6
7 def deleteNode(root, key):
8     if root is None:
9         return root
10
11     if key < root.key:
12         root.left = deleteNode(root.left, key)
13     elif key > root.key:
14         root.right = deleteNode(root.right, key)
15     else:
16         #
17         if root.left is None:
18             return root.right
19         elif root.right is None:
20             return root.left
21
22         #
23         successor = minValueNode(root.right)
24         root.key = successor.key
25         root.right = deleteNode(root.right, successor.key)
26
27 return root
```

મેમરી ટ્રીક

“શોધો, બદલો, ફરી જોડો”