**Data Structures**

**COMP6048001 - LA09 - LEC**

**Soal GSLC Session 6**

1. **Jalskan tentang Infix, Prefix, Postfix Notation dan berikan contoh!**
2. **Jelaskan mengapa diperlukan prefix/postfix notation!**
3. **Jelaskan mengenai cara Conversion Infix to Postfix Algorithm!**
4. **Jelaskan mengenai cara Conversion Infix to Prefix Algorithm!**
5. **Tuliskan secara detail tahapan Conversion Infix to Postfix => 5 + 7 \* (4 – 7) / 8**

**Lalu tuliskan tahapan evaluation terhadap Postfix notation yang diperoleh!**

1. **Tuliskan secara detail tahapan Conversion Infix to Prefix => 5 + 7 \* (4 – 7) / 8**

**Lalu tuliskan tahapan evaluation terhadap Prefix notation yang diperoleh!**

1. **Jelaskan dan berikan contoh Stack Concept pada array dan linked list!**
2. **Jelaskan dan berikan contoh Queue Concept pada array dan linked list!**

**Diagram

Description automatically generated**

1. **Berdasarkan gambar tree diatas sebutkan :**
   1. **Visit order dengan menggunakan BFS**
   2. **Visit order dengan menggunakan DFS**
2. Infix operator is written between operands (ex. 2\*10, 6+2-2, 6\*5). Prefix operator is written before operands (ex. \*2 10, +6-2 2, \*6 5)(dikerjakan dari notasi paling belakang) . Postfix operator is written after operands (ex. 4 10\*, 6 2+2-, 6 5\*)(dikerjakan dari notasi paling depan).
3. Prefix dan postfix mempermudah kerja pemograman komputer. Selain itu, prefix dan postfix juga tidak memerlukan bracket untuk menunjukan posisi prioritas sebuah operator.
4. Cara mengubah infix jadi postfix adalah dengan memindahkan notasi operator menjadi sesudah angka yang akan dihitung. Misalnya dari 6+2-2 menjadi 6 2+2-. Jadi, yang pertama dilakukan adalah 6 ditambah dengan 2 lalu dilanjutkan dengan dikurang 2.
5. Cara mengubah infix jadi prefix adalah dengan memindahkan notasi operator sebelum angka yang akan dihitung. Jika operatornya sama, maka notasinya hanya diperlukan sekali. Misalnya, 6+2+2 menjadi +6 2 2. Bila notasinya berbeda, seperti 6+2-2 maka dapat dituliskan sebagai +6-2 2.
6. Infix: 5 + 7 \* (4 – 7) / 8

Postfix: 5 7+4 7- \* 8/

Evaluation:

**5 7**+4 7- \*8/ Add 5 and 7

**12** 4 7- \*8/ the result is 12

12 **4 7-** \*8/ Minus 4 by 7

12 **-3** \*8/ the result is -3

**12 -3 \***8/ Multiply 12 by -3

**-36** 8/ the result is -36

**-36 8/** Divide -36 by 8

**-4.5** the result is -4.5

1. Infix: 5 + 7 \* (4 – 7) / 8

Prefix: / \* + 5 7 – 4 7 8

Evaluation:

/ \* **+ 5 7** – 4 7 8 Add 5 and 7

/ \* **12** – 4 7 8 the result is 12

/ \* 12 **– 4 7** 8 Minus 4 by 7

/ \* 12 **-3** 8 the result is -3

/ \* **12 -3** 8 Multiply 12 by -3

/ **-36** 8 the result is -36

**/ -36 8** Divide -36 by 8

**-4.5** the result is -4.5

1. Stack is a collection data structure with two important or main operands, push and pop. These data was Stored in **Last in First Out (LIFO)** way. Stack Operations are push(a) to add item a to the **top** of the list, pop() to remove an item from the **top** of the stack, and top() to reveal the **top** item.

Example:

[] [] [] [] [] [] n = 0 <- array 0-5

0 1 2 3 4 5

[14] [] [] [] [] [] push(14); n = 1

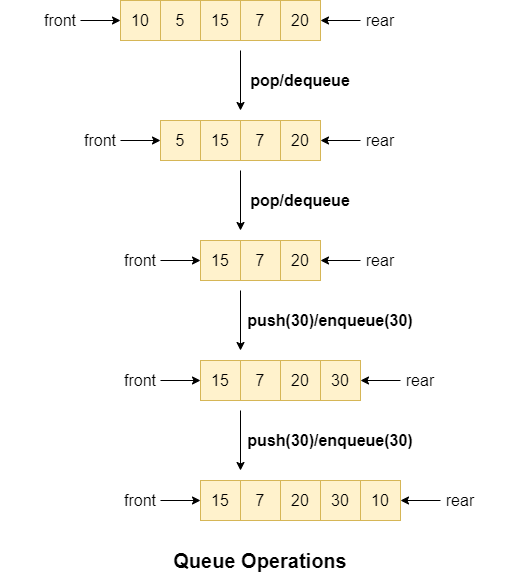
[14] [17] [] [] [] [] push(17); n = 2

[14] [] [] [] [] [] pop(); n = 1

[14] [26] [] [] [] [] top() = 26; n = 2

1. Queue is also a collection data structer with two important operands: push n pop. The data is stored in **First In First Out (FIFO)** way. Queue Operations are push(x) to add item x to the BACK of the wueue, pop() to remove item from the FRONT, and front() to reveal the most front item. Front = peek.

Example:



1. A. BFS

Push(a)

Pop(a); push(b); push(c)

Pop(b); push(d); push(e); push(f); push(g)

Pop(c)

Pop(d)

Pop(e)

Pop(f)

Pop(g)

Visit order: A, B, C, D, E, F, G

B. DFS

Push(a)

Pop(a); push(b); push(c)

Pop(c)

Pop(b); push(d); push(e); push(f); push(g)

Pop(g)

Pop(f)

Pop(e)

Pop(d)

Visit order: A, C, B, G, F, E, D

Kesimpulan:

DFS memiliki visit order yang berbeda karena DFS akkan membuang atau mengeluarkan (pop) item dari yang paling belakang. Berbeda dengan BFS yang akan mengeluarkan item dari yang paling depan. Kalau dari diagram pohon yang ada, jadi dapat dilihat perbedaannya bahwa di level yang sama, dalam DFS maka ranting paling kanan yang akan dikeluarkan terlebih dahulu dan dikerluarkan menuju ke kiri. Sedangkan menurut diagram pohon, dalam BFS dikeluarkan dari ranting yang paling kiri menuju ranting yang paling kanan. Kedua hal ini dilihat dengan kondisi terdapat ranting yang sejajar kedudukannya seperti B dan C atau d, e, f, dan g.