Struktur Data dan Algoritma

Kasus Penggunaan Stack: Komputasi Ekspresi Aritmatika

Suryana Setiawan

Definisi Problem

- Diberikan suatu ekspresi aritmatika dengan operator “+”, “-” , “*”, “/”, “^”, dan tanda kurung “(“ dan “)”,
- Contoh: “1 – 2 – 4^5 * 3 * 6 / 7^2^2”
- bagaimana algoritma untuk mengkomputasinya?

Terminologi

- Ekspresi infiks: setiap operasi aritmatika dituliskan dalam format "operand1 operator operand2"
  - Contoh "1 + 2" adalah menjumlahkan 1 dan 2.
- Ekspresi posfiks: setiap operasi aritmatika dituliskan dalam format "operand1 operand2 operator"
  - Misalnya ekspesi infiks di atas dituliskan sebagai ekspresi posfiks "1 2 +"

Bentuk Kompleks

- Bentuk kompleks ekspresi adalah jika operand adalah ekspresi aritmatika juga (secara rekursif)
  - Contoh:
    Ekspresi infiks kompleks “1 * 2 + 3”
    adalah “X + 3” dengan X adalah “1 * 2”
  - ekspresi posfiks kompleks “1 2 * 3 +”
    adalah “X 3 +” dengan X adalah “1 2 *”

Ekspresi Infiks & Masalah Interpretasi

- Perhatikan:
  - "A – B – C"
  - "A – B * C"
  - "A / B * C"
  - "A ^ B ^ C"
  - "A – B * C ^ D"
- Bagaimana urutannya?
- Perlu didefinisikan kaidah pengurutan operator.

Kaidah Urutan

- Kaidah urutan operasi “kalkulator toko” dari kiri ke kanan apapun operatornya.
- Dalam ekspresi aritmatika yang lebih umum,
  - terdapat hirarki dari operator : yang lebih tinggi dikerjakan terlebih dahulu
  - penggunaan tanda kurung untuk “mengatur” urutan jika dikehendaki operator yang lebih rendah untuk dikerjakan terlebih dahulu.

Hirarki Operator

- Paling tinggi tanda pangkat (“^”), dalam deretan berturutan tanda-tanda tsb, dikerjakan dari kanan ke kiri, misalnya “2^3^3”
- Berikutnya tanda kali (“*”) dan bagi (“/”) dalam deretan berturutan tanda-tanda tsb, dikerjakan dari kiri ke kanan, misalnya “2/3*4”
- Terendah tambah (”+”) dan kurang (“–”), dalam deretan berturutan tanda-tanda tsb, dikerjakan dari kiri ke kanan, misalnya “2-3+4”

Left / Right Associative Operator

- Tanda pangkat disebut right associative operator, karena bila sama harus menunggu terkanan dulu untuk dikomputasi sebelum ia dikomputasi
- Tanda “*”, “/”, “+”, “–” adalah left associative karena tidak perlu menunggu ke kanan.

Ekspresi Posfiks

- Ekspresi yang kurang readable untuk mata manusia dibandingkan ekspresi infiks
- Terdapat kepastian urutan operasi sehingga tidak diperlukan kaidah-kaidah seperti pada ekspresi infiks termasuk tanda kurung.
- Contoh
  - infiks “a + b / c”, posfiks “a b c / +”
  - Infiks “(a + b) / c”, posfiks “a b + c /”
  - Infiks “(a + b) / (c + e * f)”, posfiks “a b + c e f * + /”

Masalah komputasi Ekspresi

- Dengan adanya kaidah hirarki maka komputasi naif ekspresi infiks tidak bisa dilakukan secara O(n).
- Komputasi secara O(n) dapat dilakukan
  1. Pada infiks secara rekursif atau
  2. Pada postfix dengan postfix machine, yaitu algoritma dengan bantuan stack.

  No 1. tidak menjadi pokok bahasan kita, no 2. sebagai kasus penggunaan stack.

Algoritma Postfix Machine

1) Baca input T
2) WHILE input T bukan operator, push(T), baca input T.
3) Dengan input T adalah operator
   - Op1 = Pop()
   - op2 = pop()
   - Eksekusi operasi "op2 T op1" dan hasilnya Z, push(Z).
4) IF masih ada input T, Ulangi langkah (1), ELSE (jika input habis) hasil=pop()

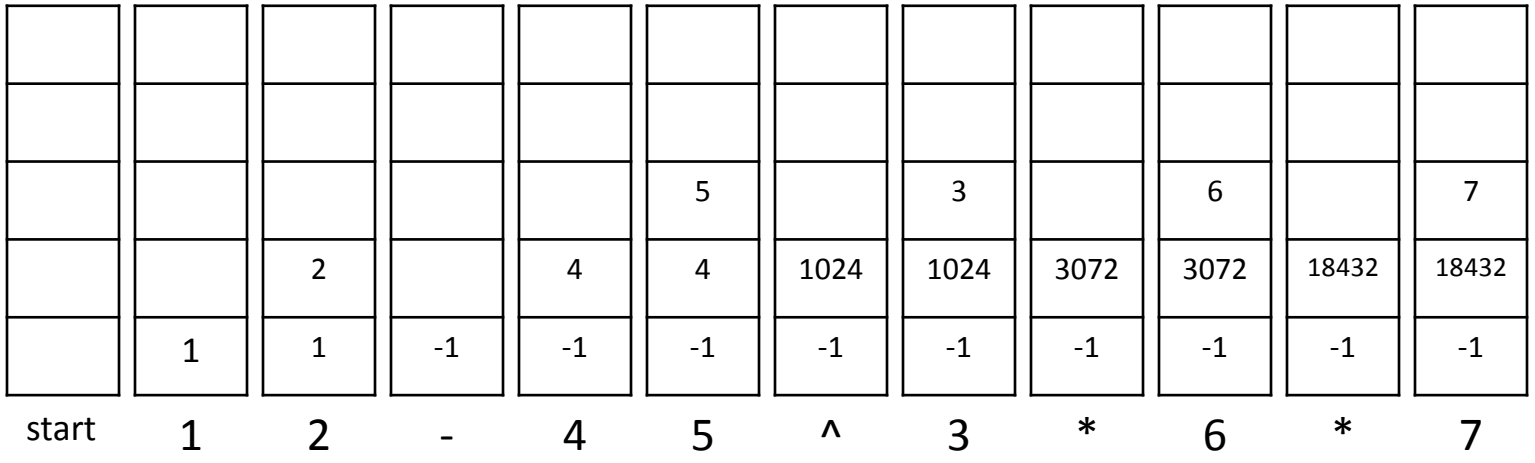
Contoh

- Ekspresi Infiks:

  “1 – 2 – 4^5 * 3 * 6 / 7^2^2”

- Ekspresi postfiks:

  “1 2 – 4 5 ^ 3 * 6 * 7 2 2 ^ ^ / -”



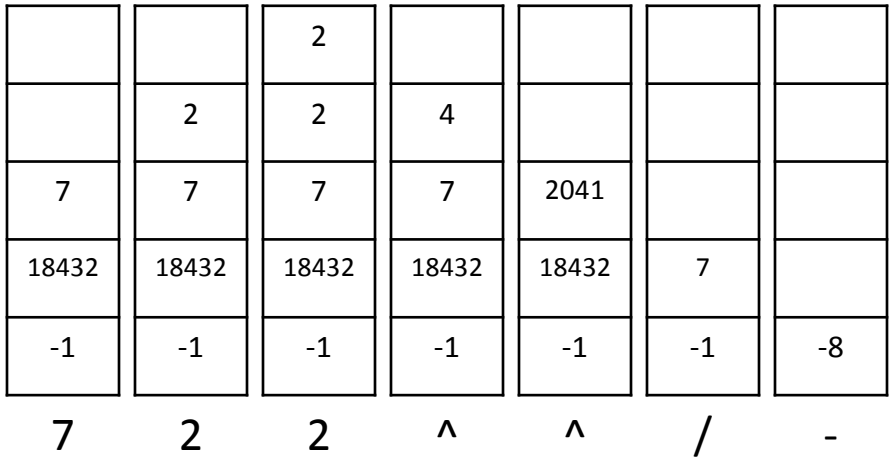
Contoh

- Ekspresi Infiks:

  “1 – 2 – 4^5 * 3 * 6 / 7^2^2”

- Ekspresi postfiks:

  “1 2 – 4 5 ^ 3 * 6 * 7 2 2 ^ ^ / -”



Hasil = -8

Kesimpulan

- Postfix machine adalah algoritma yang bekerja secara O(n)
- Masukannya berupa ekspresi postfiks
- Keluarannya adalah hasil komputasi
- Pertanyaan:
  - Bagaimana caranya mengkonversi infiks menjadi postfiks?
  - Adakah algoritma yang O(n)?

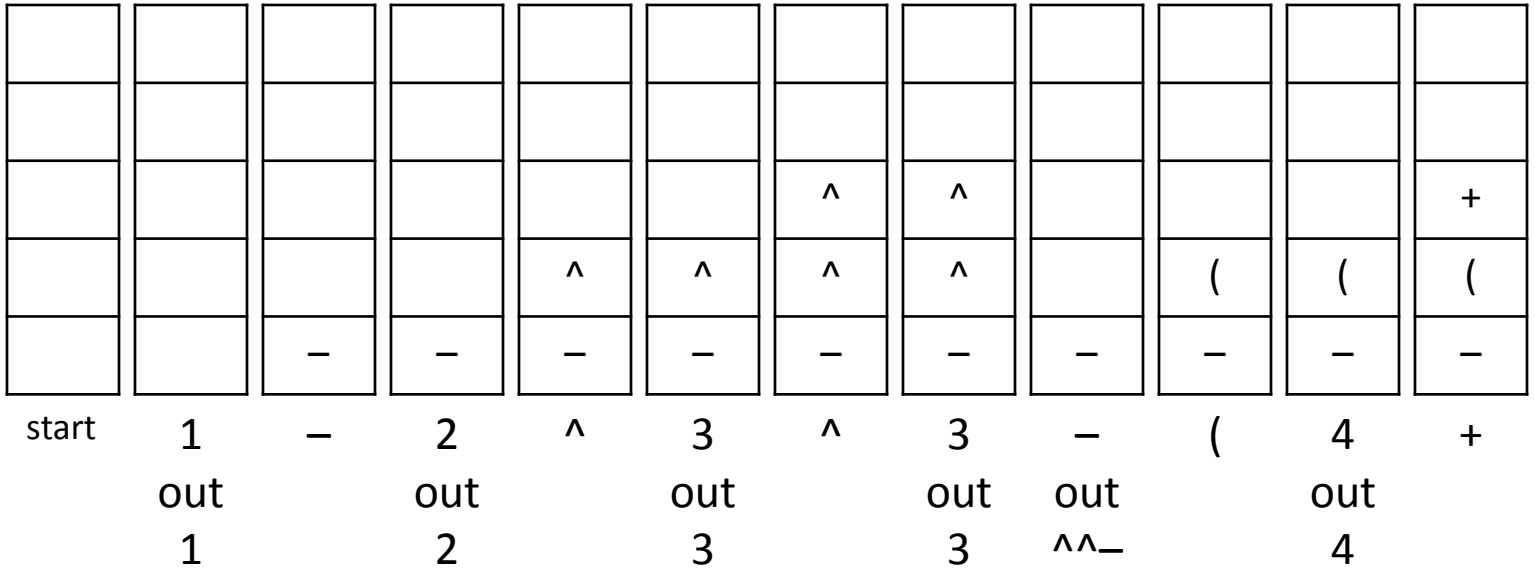
Algoritma konversi Infiks ke Postfiks

1) Baca input X
   - X operand, outputkan X
   - X kurung tutup,
     lakukan berulang
     Z=pop() dan outputkan Z jika Z operator
     hingga Z adalah kurung buka
   - X operator, lakukan hingga Top_of_stack adalah operator yang < X jika X left-associative atau Top_of_stack adalah operator yang <= X jika X right-associative , secara berulang:
     Z=pop() dan outputkan Z
   - push (X)
2) IF masih ada input, ulangi langkah (1)
   ELSE //input habis
   secara berulang, Z=pop() dan outputkan Z

Contoh

- Ekspresi Infiks:

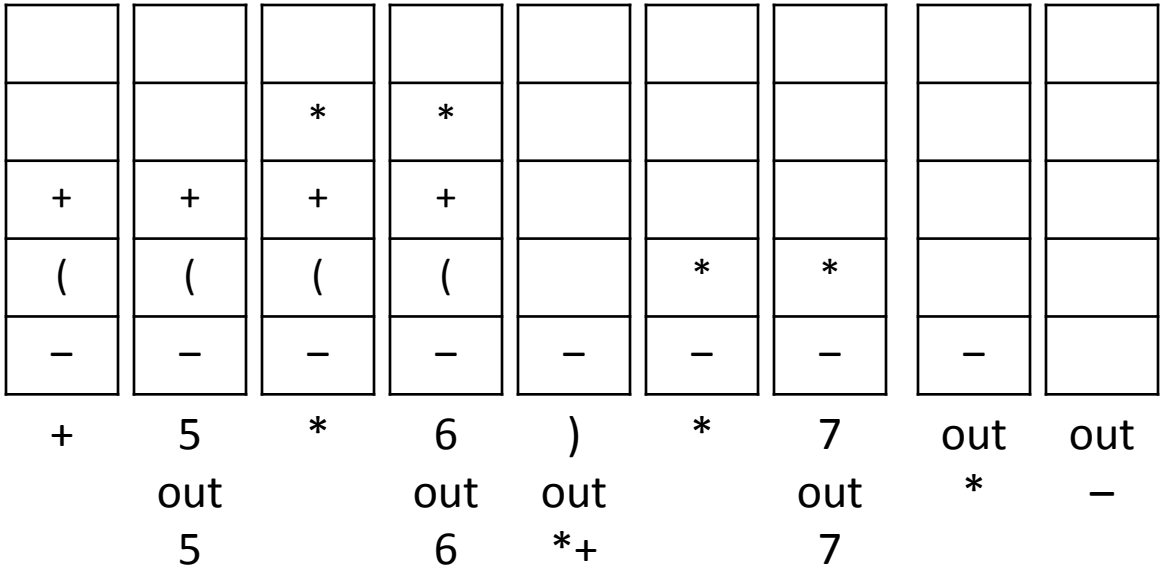
  “1 – 2 ^ 3 ^ 3 – (4 + 5 * 6) * 7”



Contoh

- Ekspresi Infiks:

  “1 – 2 ^ 3 ^ 3 – (4 + 5 * 6) * 7”



Ekpresi postfiks output: “1 2 3 3 ^ ^ – 4 5 6 * + 7 * –”

Kesimpulan

- Komputasi infiks dengan dua tahap
  - Konversi infiks menjadi posfiks → O(n)
  - Komputasi posfiks → O(n)

  adalah komputasi O(n).
- karena bantuan stack pada masing-masing tahapan.