Algdat øving 4 - kø, stakk, liste, trær

Henrik Halvorsen Kvamme

September 20, 2023

Contents

1 Introduksjon

Kravet for godkjenning er å gjøre to oppgaver. Jeg har gjort alle sammen. Jeg håper å få i hvert fall to oppgaver godkjent. All koden ligger i main.cpp og hver oppgave har sin egen funksjon med navn ‘oppgave[n]‘.

2 Oppgave 1 - Jopheus problem

I denne oppgaven brukes en sirkulær liste for å løse Jopheus problem 4.3-5 som er på side 91 i boka.

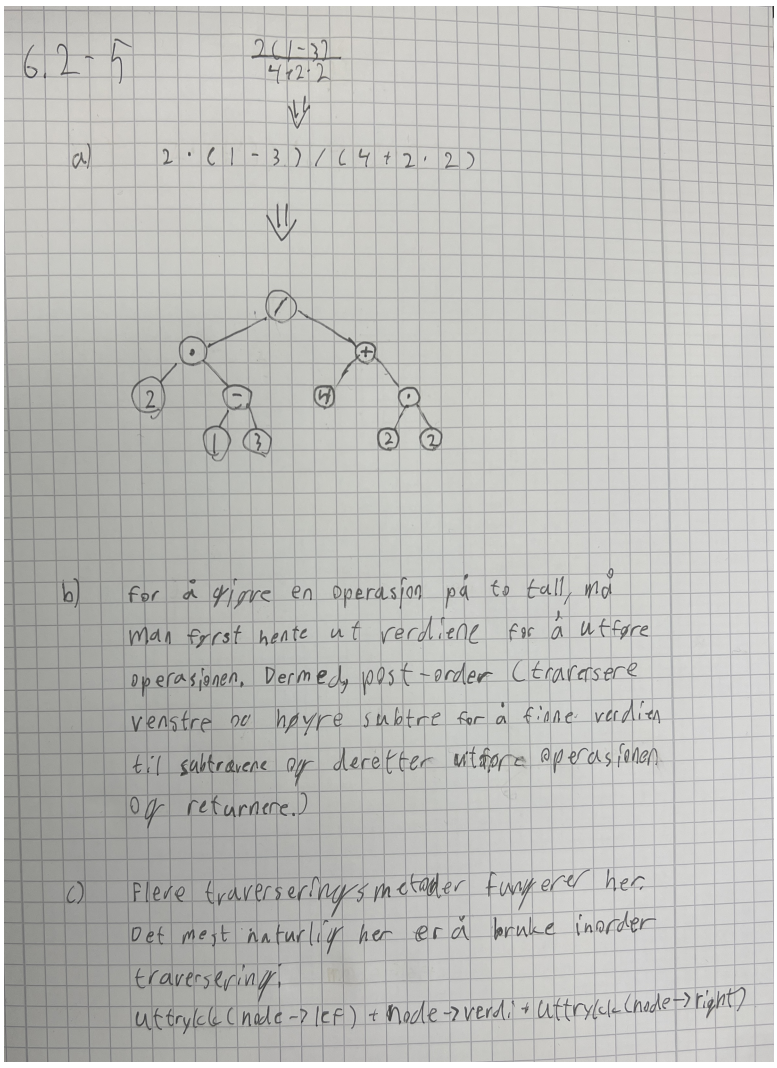
Om antall personer er n, og intervallet på antall som blir drept er m, vil tidskomplsiteten være $\theta(n) + \theta((n-1) \cdot m) = \theta(n \cdot m)$. Dette er fordi det tar n operasjoner for å initialisere den sirkulære listen. Den må også slette n-1 noder, og for hver node den sletter må den traversere mellom m noder, dermed (n-1)*m.

3 Oppgave 2 - matche parenteser, klammer og krøllparenteser

Programmet i denne oppgaven skal kunne lese en vanlig kildekodefil og sjekke bruken av parenteser, klammer og krøllparenteser. Her sjekkes det om det er noen åpne eller om noen paranteser er lukket med feil type. Jeg har utvidet programmet litt slik at den ignorerer paranteser i strenger og i kommentarer. Den leser c++ filer.

4 Oppgave 3 - uttrykkstre

I denne oppgaven skal du lage et program som kan beregne et uttrykk som er lagret i et uttrykkstre. Desuten skal den kunne skrive ut uttrykket på vanlig aritemtisk form. Deloppgavene løste jeg på papir:



5 Kildekode - main.cpp

Lettere å lese fra main.cpp filen som er vedlagt.

```
#include <fstream>
#include <iostream>
#include <map>
#include <set>
#include <stack>
#include <unistd.h>

using namespace std;

struct ListNode {
    int index;
    ListNode *next;

    ListNode(int index) : index(index),
        next(nullptr) {}
};

struct TreeNode {
    char value;
    TreeNode *left;
    TreeNode *right;

    TreeNode(char value) : value(value), left(nullptr),
right(nullptr) {}
};

int oppgave1(int n, int m) {
    // n = antall soldater, m = lengde paa hopp

    // Creating circluar list, O(n)
    ListNode *head = new ListNode(1);
    ListNode *current = head;

    for (int i = 2; i <= n; i++) {
        current->next = new ListNode(i);
        current = current->next;
    }
```

```
    current->next = head;

    // Removing nodes from list until one left: O(m*(n-1)) =
    O(m*n), because n-1 nodes is removed and m nodes must be
    traveresed for each node to be deleted.

    int t = 0;
    ListNode *prev = current;
    current = head;
    while (current->next != current) {
        t++;

        if (t == m) {
            t = 0;
            prev->next = current->next;
            current = prev;
        }

        prev = current;
        current = current->next;
    }

    // Found the last soilder standing:
    return current->index;
}

bool oppgave2(const string &fileName) {
    map<char, char> closedToOpen = {
        {'}', '{'},
        {']', '['},
        {')', '('},
    };

    set<char> openBrackets = {'{', '[', '(', '"'};

    stack<char> lastOpenBrackets;

    ifstream file(fileName);
    if (!file.is_open()) {
        cerr << "Error: Could not open file " << fileName << endl;
```

```
    return false;
}

bool inComment = false;
bool inString = false;

char c;
while (file.get(c)) {
    if (inComment) {
        if (c == '\n')
            inComment = false;

        continue;
    }

    if (inString) {
        if (c == '"')
            inString = false;

        continue;
    }

    if (c == '/') {
        file.get(c);
        if (c == '/')
            inComment = true;

        continue;
    }

    if (c == '"') {
        inString = true;
        continue;
    }

    if (!lastOpenBrackets.empty() && lastOpenBrackets.top() == '"'
        if (c == '"')
            lastOpenBrackets.pop();
        continue;
    }
```

```
        if (openBrackets.count(c)) {
            lastOpenBrackets.push(c);
        } else if (closedToOpen.count(c)) {
            if (lastOpenBrackets.empty() || lastOpenBrackets.top() !=
                return false;

            lastOpenBrackets.pop();
        }
    }

    file.close();
    return lastOpenBrackets.empty();
}

double verdiTilNode(TreeNode *node) {
    if (node->value == '+')
        return verdiTilNode(node->left) + verdiTilNode(node->right);
    else if (node->value == '-')
        return verdiTilNode(node->left) - verdiTilNode(node->right);
    else if (node->value == '*')
        return verdiTilNode(node->left) * verdiTilNode(node->right);
    else if (node->value == '/')
        return verdiTilNode(node->left) / verdiTilNode(node->right);
    else
        return node->value - '0';
}

string uttrykkFraNode(TreeNode *node) {
    // inorder
    if (node == nullptr)
        return "";

    return "(" + uttrykkFraNode(node->left) + node->value + uttrykkFra

    // preorder
    string s = "";
    if (node->value == '+') {
        s += uttrykkFraNode(node->left) + "+" + uttrykkFraNode(node-
    } else if (node->value == '-') {
        s += uttrykkFraNode(node->left) + "-" + uttrykkFraNode(node-
    } else if (node->value == '*') {
```

```
        s += "(" + uttrykkFraNode(node->left) + ") * (" + uttrykkFraNo
    } else if (node->value == '/') {
        s += "(" + uttrykkFraNode(node->left) + ") / (" + uttrykkFraNo
    } else {
        s += node->value;
    }

    return s;
}

void oppgave3() {
    TreeNode *root = new TreeNode('/');

    TreeNode *teller = new TreeNode('*');

    teller->left = new TreeNode('2');

    teller->right = new TreeNode('-');
    teller->right->left = new TreeNode('1');
    teller->right->right = new TreeNode('3');

    TreeNode *nevner = new TreeNode('+');
    nevner->left = new TreeNode('4');

    nevner->right = new TreeNode('*');
    nevner->right->left = new TreeNode('2');
    nevner->right->right = new TreeNode('2');

    root->left = teller;
    root->right = nevner;

    cout << "Uttrykket til treet er: " << uttrykkFraNode(root) << endl
    cout << "Verdien til treet er: " << verdiTilNode(root) << endl;
};

int main() {

    // Oppgave 1
    cout << "Oppgave 1:\n";
    cout << "Soilder number " << oppgave1(40, 3) << " is the last one
    cout << "Soilder number " << oppgave1(40, 3) << " is the last one
```

```
    // Oppgave 2, har også litt for kommentarer
    cout << "\nOppgave 2:\n";
    cout << oppgave2("oving4/oppgave2test1.cpp") << endl;
    cout << oppgave2("oving4/oppgave2test2.cpp") << endl;

    cout << "\nOppgave3:\n";
    oppgave3();

    return 0;
}
```