ADA exam

190483865

January 3, 2025

Contents

1 Opgave 1

```
// Jeg går ud fra at index start ved længden af ordet minus 1.
// Eksempel kald:
// std::string word = "banana";
// countLettersInWord(word, 'a', word.length() - 1);

int countLettersInWord(std::string word, char letter, int index) {
  if (index < 0) {
    return 0;
  }
  if (word[index] == letter) {
    return 1 + countLettersInWord(word, letter, index - 1);
  } else {
    return countLettersInWord(word, letter, index - 1);
  }
}
```

2 Opgave 2

For loop 1:

Yderste for loop har: $O(\log N)$ pga. $i+ = i$.

Mellemste har $O(\log^2 N)$ da $\log^2 N \sim 2 \cdot \ln^2 N$

Inderste har: $O(\sqrt{N})$, da k kører til kvadratroden af N.

For loop 2:

Tidskompleksitet $O(N \cdot \sqrt{N})$

Værdier fra test:
N:300,x:5346,y:5197
N:301,x:5346,y:5223
N:302,x:5346,y:5249
N:303,x:5346,y:5275
N:304,x:5346,y:5301
N:305,x:5346,y:5327
N:306,x:5346,y:5353
N:307,x:5346,y:5380
N:308,x:5346,y:5406
N:309,x:5346,y:5432
N:310,x:5346,y:5459

Det ses at ved værdier over $N >= 306$ er 2. for loop langsommere end det første. Dvs. $N < 306$ er 2. for loop hurtigere end det første.

Den samlede tidskompleksitet er

$$\begin{aligned} N >= 306: &\quad O(N \cdot \sqrt{N}) \\ N < 306: &\quad O(\log^3 N \cdot \sqrt{N}) \end{aligned}$$

3 Opgave 3

Nyt kode er har "// Ny" efter sig.

```
int main(){
  // Samme kode oppe over
  string minTask = "";// Ny
  string finalMaxSlack = "";// Ny
  int minVarighedAktuelEvent = numeric_limits<int>::max();// Ny (#include <limits>)
  int maxSlack = 0;// Ny

  while (true) {
    while (indeks < tabel.size() && tabel[indeks].getEvent() == aktuelEvent) {
      if (maxVarighedAktuelEvent < tabel[indeks].getDuration()) {
        maxVarighedAktuelEvent = tabel[indeks].getDuration();
        maxTask = tabel[indeks].getTask();
      }
      if (minVarighedAktuelEvent > tabel[indeks].getDuration()) { // Ny
        minVarighedAktuelEvent = tabel[indeks].getDuration();// Ny
        minTask = tabel[indeks].getTask();// Ny
      }// Ny

      indeks++;
    }
    int slackAktuelEvent = maxVarighedAktuelEvent - minVarighedAktuelEvent;// Ny
    if (slackAktuelEvent > maxSlack) {// Ny
      maxSlack = slackAktuelEvent;// Ny
      finalMaxSlack = minTask;// Ny
    }// Ny

    laengdeKritiskVej += maxVarighedAktuelEvent;
    kritiskVej += maxTask;
    maxVarighedAktuelEvent = 0;
    minVarighedAktuelEvent = numeric_limits<int>::max();// Ny
    maxTask = "";
    minTask = ""; // Ny

    if (indeks == tabel.size())
      break;
    aktuelEvent = tabel[indeks].getEvent();
  }
  cout << "Max Slack: " << finalMaxSlack << " : " << maxSlack << endl;

  // Samme kode efter dette
}
```

4 Opgave 4

```
void BinarySearchTree::printRoute(int value) {
  std::string route = findRoute(root, value);
  if (route != "") {
    cout << route << endl;
  } else {
    cout << "No route to " << value << " found." << endl;
```

```
  }
}

string BinarySearchTree::findRoute(BinaryNode *root, int value) {
  if (!root)
    return "";
  if (root->element == value) {
    return to_string(value);
  }
  std::string left = findRoute(root->left, value);
  std::string right = findRoute(root->right, value);
  if (left.find(to_string(value)) != std::string::npos) {
    return to_string(root->element) + " " + left;
  } else if (right.find(to_string(value)) != std::string::npos) {
    return to_string(root->element) + " " + right;
  }
  return findRoute(root->left, value) + findRoute(root->right, value);
}
```

5 Opgave 5

Brug af Dijkstras Algorithm:

v	Known	d_v	p_v
F	true	0	0
A	true	15	B
B	true	14	D
C	true	16	A
D	true	5	E
E	true	4	F
G	true	34	J
H	true	21	C
I	true	19	A
J	true	24	I

Jeg bruger Kruskal's. Rækkefølge følger tabellen.

Vægt	Node-par
1	(E,D)
1	(B,A)
1	(A,C)
2	(D,I)
4	(F,E)
4	(A,I)
5	(C,H)
5	(I,J)
10	(J,G)

Samlet vægt er: 33

6 Opgave 6

Y0=3
Y1=4
Y2=7
Y3=12
Y4=6
Y5=2
Y6=0

De samme tal fortsætter herefter...

Y7=0
Y8=2
Y9=6
Y10=12

Sådan vil tabellen se ud:

Index	Item
0	Y6
1	
2	Y5
3	X
4	Y1
5	
6	Y4
7	Y2
8	
9	
10	
11	
12	Y3

7 Opgave 7

Dette skulle representere en min-heap det vil sige alle tal under skal være større. 2 bryder dette:

27 på index 11 er mindre end den prioritet oppeover på 28.

18 på index 15 er mindre end den prioritet oppeover på 26.

Derfor er dette ikke en prioritetskø.