Eksamen H2020 ADS101
Algoritmer og dastrukturer for spill

10/12/20

Tillatte hjelpemidler: Filer og programmer på egen pc, lærebok og notater, kursets Canvas-rom med lenker.

1

(teori, 30%)
Gitt tallene 6, 8, 11, 4, 3, 9, 7. I hvert av punktene nedenfor skal du sette inn tallene i denne rekkefølgen.

a) Tegn opp et binært søketre etter at tallene er satt inn.

b) Tegn opp treet etter at tallet 6 er slettet.

c) Tegn opp array representasjonen til treet fra punkt b.

d) Tegn og vis hvordan du bygger en min-heap med de samme tallene.

e) Tegn heapen etter en pop operasjon.

f) Gitt en hashtabell med n=7 og hashfunksjonene
```
int hash(int key) { return (key % n); }
int dobbelthash(int key) {return (1 + key%(n-1)); }
```

Tegn opp hashtabellen etter at de samme tallene er satt inn.

2

(programmering, 15%)
Følgende C++ kode er gitt:

```
class BinaryNode {
public:
    BinaryNode(double data=0.0);
    void insert(double data);
    void stigende() const;
    void avtakende() const;
private:
    int m_data;
    BinaryNode* m_left;
    BinaryNode* m_right;
};

BinaryNode::BinaryNode(int data)
   : m_data{data}, m_left{nullptr}, m_right{nullptr} { }

void BinaryNode::insert(int data) {
    if (data < m_data) {
        if (m_left)
            m_left->insert(data);
        else
            m_left = new BinaryNode(data);
    } else if (data > m_data) {
        if (m_right)
            m_right->insert(data);
        else
            m_right = new BinaryNode(data);
    }
}
```

a) Skriv koden til funksjonen stigende() slik at den skriver ut nodenes verdier i stigende rekkefølge når den kalles av roten i et tre.

b) Skriv koden til funksjonen avtakende() slik at den skriver ut nodenes verdier i avtakende rekkefølge når den kalles av roten i et tre.

c) Skriv et testprogram hvor du setter inn verdiene 14, 33, 12, 4, 7, 19, 29, 11, 21, 17 i opplistet rekkefølge og deretter skriver ut stigende og avtakende.

3

(teori og programmering, 15%) Linjene nedenfor viser sorteringen av noen tall.

```
14, 33, 12, 4, 7, 19, 29, 11, 21, 17,
14, 33, 4, 12, 7, 19, 11, 29, 17, 21,
4, 12, 14, 33, 7, 11, 19, 29, 17, 21,
4, 7, 11, 12, 14, 19, 29, 33, 17, 21,
4, 7, 11, 12, 14, 17, 19, 21, 29, 33,
```

a) Forklar hvordan tallene er sortert, og hvilken algoritme som er benyttet.

b) Skriv kode hvor du bruker std::priority_queue til å sortere de samme tallene (14, 33, 12, 4, 7, 19, 29, 11, 21, 17) i avtakende rekkefølge.

4

(programmering, 25%)
I denne oppgaven skal du bruke kildekode fra listing nedenfor (Node, Edge og Graph kode).

a) Implementer en konstruktør Graph::Graph() som legger inn nodene til grafen på figur 1.

b) Implementer Node* Graph::find_node(char navn) slik at den returnerer en peker til noden med gitt navn hvis den fins i grafen, og en nullpeker ellers.

c) Implementer void Graph::make_edges(char x, char y) slik at den setter inn kanter mellom xy og yx med vekt 1, gitt at noder med navn x og y fins i grafen.

d) Implementer void make_all_edges() slik at funksjonen setter inn alle kantene på grafen på figur 1.

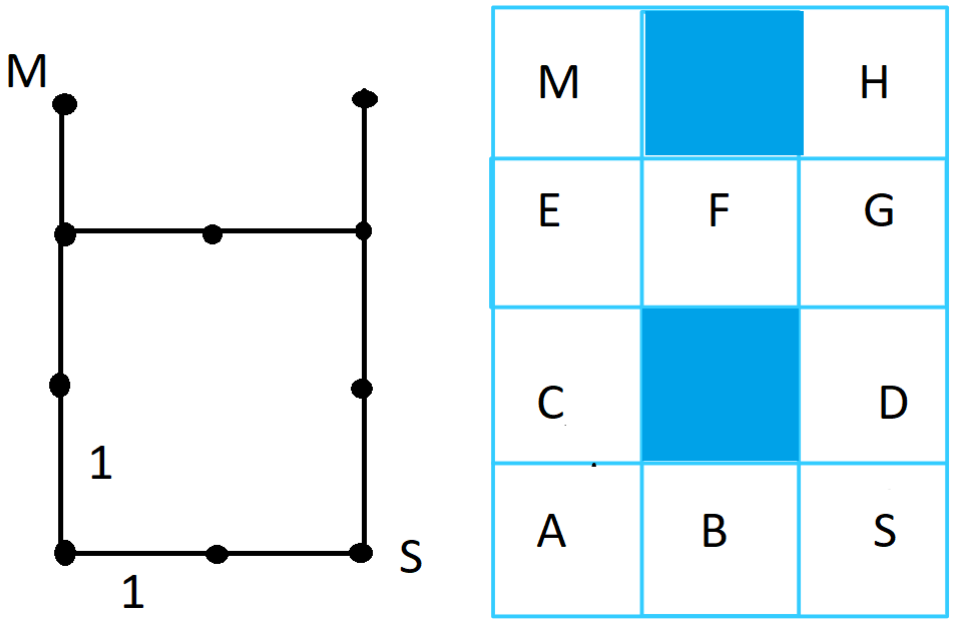
e) Implementer void Graph::print() const slik at den skriver ut en oversikt over alle noder og tilhørende kanter i grafen. Skriv deretter main() hvor grafen på figur 1 opprettes og skrives ut.

Listing 1: Node, Edge og Graph kode

```
struct Node;
struct Edge {
    Node* m_fra;
    Node* m_til;
    double m_vekt;
    Edge();
    Edge(Node* fra, Node* til);
};
Edge::Edge() : m_fra{nullptr}, m_til{nullptr}, m_vekt{0.0} { }
Edge::Edge(Node* fra, Node* til) : m_fra{fra}, m_til{til}, m_vekt{1.0} { }

struct Node {
    char m_navn;
    std::vector<Edge*> m_kanter;
    void push_edge(Edge* kant) { m_kanter.push_back(kant); }
};

struct Graph {
    std::vector<Node*> m_noder;
    Graph();
    Node* find_node(char navn);
    void make_edges(char x, char y);
    void make_all_edges();
    void print() const;
};
```



Figur 1: Graf

5

(teori, 15%)
Figur 1 viser samme graf tegnet på to forskjellige måter, med navngitte noder og hvor alle kantene har vekt = lengde = 1. La S være en startnode og M en sluttnode. Bruk A* algoritmen og bestem korteste vei fra startnode til sluttnode. Skriv opp prioritetskøa med vekt/lengde underveis.

Slutt på oppgaven.