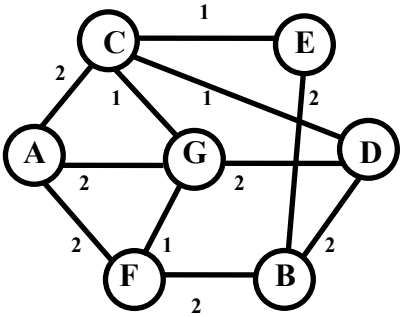
Obligatorisk oppgave nr.5

IDATG2102 – Algoritmiske metoder, høsten 2024

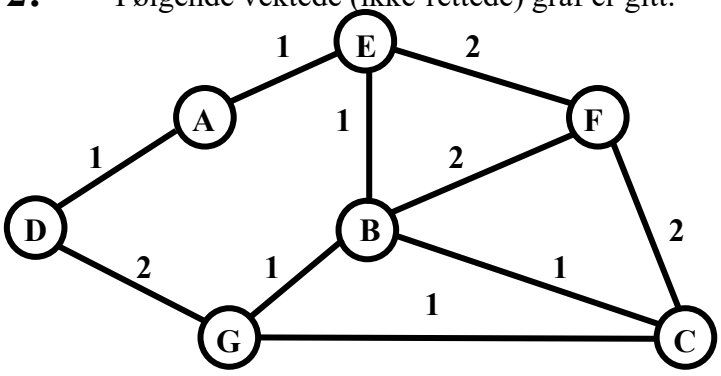
Frist: 29.oktober 2024 kl.11:00 (*må* overholdes) *via Blackboard*

Oppgave 1: Følgende vektede (ikke-rettede) graf er gitt:



Vi bruker nabomatrise, og starter i node A. **Tegn opp minimums spenntreet (MST)** for denne grafen, etter at koden i EKS_31_MST.cpp er utført/kjørt.
Skriv også opp innholdet i/på fringen etterhvert som konstruksjonen av MST pågår.
NB: *Husk at ved lik vekt så vil noden sist oppdatert (nyinnlagt eller endret) havne først på fringen ift. andre med den samme vekten.*

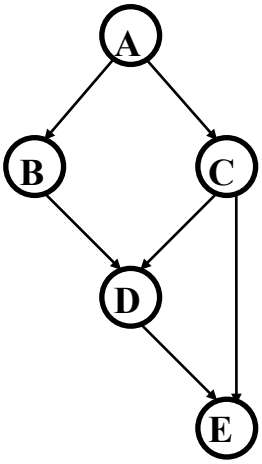
Oppgave 2: Følgende vektede (ikke-rettede) graf er gitt:



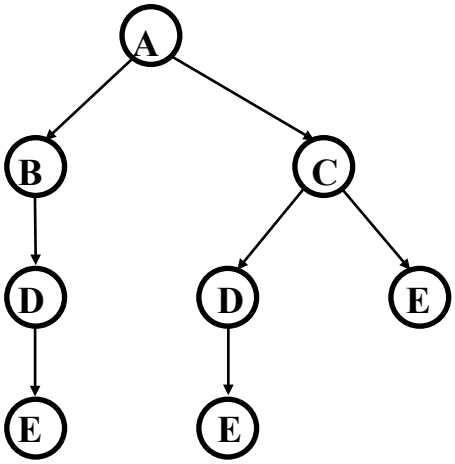
Vi bruker nabomatrise. Koden i EKS_32_SP.cpp utføres/kjøres på denne grafen.
Hvilke kanter er involvert i korteste-sti spenntreet fra noden C til *alle* de andre nodene? Skriv også opp innholdet i/på fringen etterhvert som koden utføres.
NB: *Husk at ved lik vekt så vil noden sist oppdatert (nyinnlagt eller endret) havne først på fringen ift. andre med den samme vekten.*

Oppgave 3:

Gitt en rettet graf uten løkker/sykler. *Alle mulige veier gjennom grafen, fra en bestemt node, kan beskrives som et tre der nodene er kopier av nodene i grafen.*
I eksemplet under er det vist hvordan en graf kan «brettes» ut til et slikt tre:



Opprinnelig graf med nodene A, B, C, D, E.



Treet som representerer alle veier fra noden A.

Vi skal ta utgangspunkt i følgende grafrepresentasjonen av en node og dens naboer:

```
struct Node  {
     char  ID;            // Nodens id/key/data
     Node* naboer[10];    // Nodens naboer (ligger i indeks nr. 0 - 9)
     int   antNaboer;     // Antall naboer (max. 10 stk)
     bool  besokt;        // Om besøkt noden hittil eller ei
};
```

Skriv *en* rekursiv funksjon: `void brettUt(Node* node);`
som omformer grafen med utgangspunkt i node `node` til et tre som forklart ovenfor.

Du kan forutsette at *alle* nodene i grafen kan nåes fra noden `node` (dvs. at grafen er sammenhengende). Du kan også anta at `besokt` er lik `false` (=USETT) i alle nodene når funksjonen starter opp. I tillegg kan du anta at følgende funksjon er tilgjengelig/ferdiglaget:

```
                    //  Kopierer rekursivt noden "node" og alle dens etterfølgere:
Node* kopier(Node* node)  {
  Node* kopi = new Node(node->ID, node->antNaboer);
  for (int i = 0;  i < node->antNaboer;  i++)
      kopi->naboer[i] = kopier(node->naboer[i]);
  return kopi;
}
```

Denne returnerer altså en peker/referanse til en kopi av den delen av en graf som består av noden `node` og alle nodene som kan nåes fra denne noden (og som altså allerede er brettet ut).

Hele obligen (både tegninger, tekst og kode) leveres som _en_ PDF innen fristen via emnets rom i Blackboard.

FrodeH