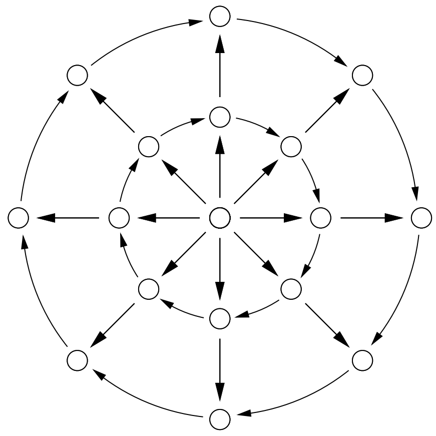
Opgave 1 (20%)

En *hjulgraf* er en orienteret graf af følgende type:



Mere præcist består en hjulgraf af en centerknude c, hvorfra der udgår k orienterede "eger", hver bestående af s kanter. Alle egers i'te knude (for $i = 2, 3, \ldots, s+1$) er desuden forbundet til en orienteret kreds. Orienteringen af alle kanter er som indikeret af figuren ovenfor. På figuren er $k = 8$ og $s = 2$.

Spørgsmål a: Angiv antallet m af kanter i en hjulgraf som funktion af antallet n af knuder. □

Vi lader nu kanterne i hjulgrafen være vægtede med positive heltal. Vi ønsker at finde længderne af de korteste veje fra centerknuden c til alle andre knuder i hjulgrafen.

Spørgsmål b: Angiv udførelsestiden (udtrykt ved n), hvis vi bruger Dijkstras algoritme. □

Spørgsmål c: Beskriv (i ord) en $O(n)$ algoritme, som finder de korteste veje fra centerknuden c til alle andre knuder i en hjulgraf. Du skal både gøre rede for algoritmens kompleksitet, og for at den finder det ønskede.

[Vink: Tag en kreds ad gangen, og start undersøgelsen af hver kreds ved en knude, som i en passende forstand er minimal.]

□

Opgave 2 (25%)

I denne opgave betragtes den sædvanlige ADT *dictionary* (jf. Goodrich og Tamassia, side 248), som ønskes udvidet med en operation

COUNT(k): Angiv antallet af elementer i D, hvis nøgle er større end k.

Spørgsmål a: Vis hvorledes en sådan udvidet dictionary kan implementeres, så operationerne FINDELEMENT, INSERTITEM, REMOVE og COUNT alle får logaritmiske udførelsestider.

[Vink: Udvid et rød-sort træ med passende ekstra information i knuderne.] □

En *inversion* i et array S er som bekendt et par af elementer $S[i]$, $S[k]$, hvor $i < k$ og $S[i] > S[k]$.

Spørgsmål b: Beregn antallet af inversioner, hvis array'ets indhold i rækkefølge er

$$[5, 2, 7, 1, 9, 4, 6]$$

□

Spørgsmål c: Beskriv en $O(n \log n)$ algoritme, der under brug af den udvidede dictionary fra spørgsmål **a** finder antallet af inversioner i et array S. Du skal både gøre rede for algoritmens kompleksitet, og for at den finder det ønskede. □

Opgave 3 (30%)

Denne opgave handler om at give penge tilbage med et mindst muligt antal mønter. De tilgængelige møntstørrelser kaldes tilsammen et *møntsæt*, og beskrives ved en liste $(m_1, m_2, \ldots, m_K)$ af heltal i stigende orden, hvor K er antallet af forskellige møntstørrelser. Vi antager, at $m_1 = 1$ (for at være sikker på at kunne give alle beløb tilbage), og at alle tallene er forskellige. Eksempler på møntsæt er $(1, 5, 10, 50)$ og $(1, 7, 9, 13)$.

For et givet møntsæt kan en samling mønter beskrives ved en vektor $(v_1, v_2, \ldots, v_K)$ af ikke-negative heltal, hvor v_i angiver, hvor mange mønter vi har af størrelse m_i. Det samlede antal mønter er naturligvis $\sum_{i=1}^{K} v_i$. Hvis

$$\sum_{i=1}^{K} v_i m_i = B,$$

d.v.s. hvis mønterne tilsammen repræsenterer beløbet B, kalder vi vektoren en B-vektor. En B-vektor er *optimal*, hvis ingen anden B-vektor bruger færre mønter. For eksempel er $(0, 3, 2, 0)$ og $(0, 1, 3, 0)$ begge 35-vektorer for det første møntsæt ovenfor, men kun den sidste vektor er optimal.

Givet et møntsæt og et beløb B, ønsker vi at finde en optimal B-vektor.

En oplagt algoritme af grådig type er beskrevet ved nedenstående pseudo-kode, hvor **div** og **mod** betegner henholdsvis heltalsdivision og rest ved heltalsdivision:

```
OptimalSum(B)
  for i = K down to 1 do
    v_i = B div m_i;
    B = B mod m_i;
```

Man kan vise (men det kræves ikke her), at for visse møntsæt er denne algoritme korrekt (f.eks. hvis møntsættet opfylder, at m_i går op i m_{i+1} for $i = 1, 2, \ldots, K-1$). Dette gælder dog ikke for alle møntsæt:

Spørgsmål a: Vis med et eksempel, at denne algoritme ikke er korrekt for alle møntsæt. D.v.s. find et møntsæt og et beløb, hvor algoritmen svarer forkert. □

For et givet møntsæt $(m_1, m_2, \ldots, m_K)$ lader vi $a[k, b]$ betegne det mindste *antal* mønter i nogen b-vektor som kun anvender møntstørrelserne $m_1, m_2, \ldots, m_k$. Her er k og b heltal, hvor $1 \le k \le K$ og $0 \le b$.

Spørgsmål b: Gør rede for, at $a[k,b]$ kan beskrives ved følgende rekursionsformel:

$$a[k,b] = \begin{cases} b\,, & k = 1 \\ \min\limits_{0 \le i \le b \,\mathbf{div}\, m_k} \{\, i + a[k-1, b - i \cdot m_k] \,\}\,, & k > 1 \end{cases}$$

$\square$

Spørgsmål c: Beskriv en algoritme baseret på dynamisk programmering, som for et givet møntsæt $(m_1, m_2, \ldots, m_K)$ og beløb B finder antallet af mønter i en optimal B-vektor i tid $O(KB^2)$. Der kræves et argument for, at algoritmen har den angivne kompleksitet. $\square$

Spørgsmål d: Gør rede for, hvordan algoritmen kan udvides til ud over det mindste antal mønter også at finde en optimal B-vektor. $\square$

Opgave 4 (25%)

Heltalskvadratroden af et tal $a \geq 1$ er som bekendt det positive heltal x, for hvilket

$$x^2 \leq a < (x+1)^2.$$

Herons formel er en klassisk metode til at beregne kvadratrødder iterativt. I denne opgave ser vi på heltalsversionen af Herons formel (i det følgende betegner vi således heltalsdivision med ":").

Algoritme: Heron
Input: a, positivt heltal
Output: x, hvor $x^2 \leq a < (x+1)^2$
Metode: $x \leftarrow a$
$\{\ a < (x+1)^2 \ \wedge\ x \geq 1\ \}$
while $a < x * x$ **do**
  $x \leftarrow (x + a : x) : 2$

Spørgsmål a: Angiv hvilke bevisbyrder, der skal eftervises i et gyldighedsbevis for algoritmen. □

Spørgsmål b: Eftervis bevisbyrderne fra spørgsmål **a**.

[Vink: Observér at $\frac{1}{2}(x + \frac{a}{x}) - 1 \leq (x + a : x) : 2 \leq \frac{1}{2}(x + \frac{a}{x})$.] □

Spørgsmål c: Argumentér for, at algoritmen er korrekt. □