Norsk informatikkolympiade 2016–2017 — 1. runde

Sponset av

UNIVERSITETET I BERGEN
Institutt for informatikk

FFI Forsvarets forskningsinstitutt

STARTUP LAB

Kantega

Forskningsrådet

Uke 46, 2016

Tid: 90 minutter
Tillatte hjelpemidler: Kun skrivesaker. Det er *ikke* tillatt med kalkulator eller trykte eller håndskrevne hjelpemidler.
Instruksjoner: Oppgavesettet består av 16 oppgaver, med fire svaralternativer på hver oppgave. Det er kun ett riktig svar på hver oppgave. Du får fire poeng for hvert riktige svar, null poeng for feil svar, og ett poeng for hver oppgave du ikke svarer på (det vil si at det ikke lønner seg å gjette dersom du ikke vet hva svaret er). Du kan godt krysse av på alternativene i oppgaveteksten underveis, men du *må* føre inn svarene på svararket helt bakerst.
Oppgavene som handler om programmering starter med beskrivelser av temaet de handler om, slik at de som ikke har vært borti programmering før også kan prøve seg på disse oppgavene. De som kan programmere trenger ikke å lese disse beskrivelsene; all koden oppfører seg som i vanlige programmeringsspråk.

FASIT

1. Du skal tegne opp et rutenett på 8 ganger 8 ruter hvor hver rute er 1 cm $\times$ 1 cm. Hvor mange cm med streker trenger du å tegne?

   A. 128

   B. 144

   C. 196

   D. 256

   Løsningskommentar: Du må tegne 9 horisontale streker med lengde 8, og 9 horisontale streker med lengde 8. $9 \times 8 \times 2 = 144$

2. Arne har en vanlig kortstokk med 52 kort. Han ber Nils om å tenke på et valgfritt kort. Deretter deler Arne kortstokken i to bunker (han kan bygge opp bunkene nøyaktig slik han vil) og spør Nils om hvilken av bunkene kortet ligger i. Arne fortsetter å dele kortstokken i to og spørre helt til han er sikker på hvilket kort Nils hadde valgt. Dersom Arne deler opp kortstokken optimalt hver gang, hvor mange ganger må han spørre før han vet helt sikkert hvilket kort Nils valgte?

   A. 6

   B. 7

   C. 13

   D. 26

   Løsningskommentar: Først deler Arne kortstokken i to like store bunker (26 kort hver). I neste runde deler Arne den slik at hver av bunkene inneholder 13 av de 26 kortene som det fortsatt er mulig at Nils har valgt. Ved å fortsette det slik så kan han halvere (rundet av oppover) antall mulige kandidater for hver gjetting. Etter 6 gjettinger så er det bare én kandidat igjen.

3. I det binære tallsystemet bruker man kun to forskjellige sifre: 0 og 1. Sifferet lengst til høyre er 1'er plassen, til venstre for dette har man 2'er-plassen, deretter 4'er-plassen, 8'er-plassen, 16-plassen, og så videre. Hver sifferplass er altså dobbelt så mye verdt som sifferplassen ett hakk til høyre. Verdien av et tall i det binære tallsystemet finner man ved å summere verdiene for posisjonene som har sifferverdi 1. For eksempel så tilsvarer det binære tallet 11001 verdien $16 + 8 + 1$, altså 25. Hvordan skriver man tallet 199 i binært?

   A. 11000111

   B. 11010011

   C. 11100001

   D. 11100111

   Løsningskommentar: $199 = 128 + 64 + 4 + 2 + 1$

4. Når man skal behandle bilder må man som regel laste inn fargeverdiene for hvert eneste *piksel* (punkt) på bildet som 3 *bytes*. De 3 bytesene for et piksel er tall mellom 0 og 255 som forteller

hvor mye det er av henholdsvis blått, grønt og rødt i det pikselet. Hvor mange bytes med minne vil man trenge for å holde på et bilde som er 2000 piksler bredt og 1500 piksler høyt på denne måten?

A. 10500

B. 3000000

C. 9000000

D. 765000000

Løsningskommentar: Man har 2000×1500 piksler og hvert piksel bruker 3 bytes.

5. I *boolsk logikk* jobber vi kun med verdiene **true** og **false**. Vi skal se på tre operatorer som kan brukes til å lage boolske *uttrykk* (formler):

   - NOT-operatoren gir ut motsatt sannhetsverdi av det man mater inn i den: NOT x blir **true** dersom x er **false**, og **false** dersom x er **true**.
   - AND-operatoren brukes med to verdier: x AND y, og gir ut **true** bare hvis både x og y er **true**.
   - OR-operatoren brukes slik: x OR y, og gir ut **true** så lenge minst én av x og y (eller begge to) er true. Altså gir den ut **false** bare hvis både x og y er **false**.

   Disse operatorene kan kombineres til større uttrykk. Parenteser angir rekkefølgen uttrykket skal *evalueres* (regnes ut) i, slik som vi kjenner fra matematikken.

   Hvilket av følgende uttrykk er **true** kun når a er **true** og b er **false**?

   A. NOT (a AND b)

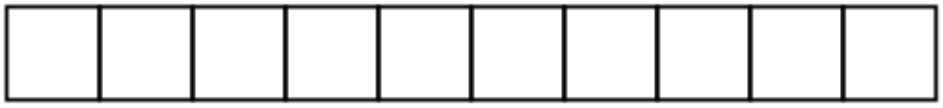
   B. b AND (NOT A)

   C. (a OR b) AND (NOT b)

   D. (NOT (a AND b)) AND (a OR (NOT b))

   Løsningskommentar: B er ikke sann når a er **true** og b er **false**. Både A og D er sanne i tilfellet når både a og b er **false**.

6. David øsnker å fargelegge hver av rutene under enten røde eller blå. For å ha litt variasjon så øsnker han at det aldri skal være 4 eller flere ruter etter hverandre i samme farge. Hvor mange forskjellige fargeleggінger er det som tilfredsstiller dette kravet?



A. 377

B. 504

C. 512

D. 548

Løsningskommentar: La $F(n)$ betegne antall måter å fargelegge n ruter, slik at kravet over er tilfredsstillt. Alle fargeleggingar av 3 eller færre ruter er gylig, så $F(n) = 2^n$ når $n \leq 3$. Alle fargeleggingar av n ruter når $n > 3$ kan lages ved å ta en fargelegging av $n-1$, $n-2$ eller $n-3$ ruter, og så legge til hhv. 1, 2 eller 3 ruter i motsatt farge av den siste i den forrige fargeleggingen. Altså er $F(n) = F(n-1) + F(n-2) + F(n-3)$ når $n > 3$. Ved lage en tabell over $F(n)$ fra $n = 1$ til 10 får man at $F(n) = 548$.

7. I de fleste programmeringsspråk brukes operatoren `=` på en annen måte enn i matematikken. Den instruerer nemlig datamaskinen om å regne ut verdien av uttrykket på høyre side av likhetstegnet og legge den inn i variabelen på venstre side. (Verdien *kopieres* alltid, den *flyttes* ikke.) En variabel holder på en verdi helt frem til du legger noe annet inn i den; da forsvinner den gamle verdien. Linjer som står etter hverandre utføres etter tur. For eksempel vil

```
x = 9
x = x - 3
```

gjøre at variabelen x ender opp med å inneholde verdien 6. Hva blir verdien av x etter at følgende kode er utført?

```
x = 4
y = (x * 3) + 9
x = y - x
```

A. 12

B. 13

C. 17

D. 21

8. En *funksjon* er en navngitt samling med programinstruksjoner, som kan *kalles* (startes) med *inndata* og *returnere* (gi tilbake) *utdata*. `return`-kommandoen brukes til å avslutte funksjonen og returnerer resultatet av uttrykket som står på samme linje. Hva returnerer funksjonen under dersom den kalles `f(5)`? (altså slik at `A` har verdien 5 når funksjonen starter)

```
f(A) {
    X = A * 3
    Y = X - 4
    X = Y
    return A + X + Y
}
```

A. 27

B. 29

C. 31

D. 33

9. Kommandoen `if` sjekker om uttrykket som står i de tilhørende parentesene er **true** (sann) eller **false**(usann); hvis det er **true**, gjøres det som står i krøllparentesene etter `if`; hvis det er **false**, gjøres det som står i krøllparentesene etter den tilhørende `else`'n.

   En *while-løkke* utfører koden inni krøllparentesene sine gjentatte ganger. Før hver utførelse sjekker løkken betingelsen inni parentesene rett etter `while`. Dersom betingelsen er usann, avsluttes løkken, og man fortsetter med koden etter avslutningskrøllparentesen.

   Funksjonen under finner ut en egenskap om det positive heltallet x. I nøyaktig hvilke tilfeller vil funksjonen returnere **true**.

```
f(x) {
    s = 1
    while (s < x) {
        s = s + 2
    }
    if (s > x) {
        return true
    } else {
        return false
    }
}
```

A. når x er et partall

B. når x er et oddetall

C. når x er en toerpotens ($x = 2^n$ for et heltall $n \geq 0$)

D. når x er et primtall

Løsningskommentar: s går gjennom alle positive oddetall helt til man kommer til et tall som ikke er mindre enn x. Dersom tallet er større enn x vil det si at man hoppet over x, og dermed er x et partall.

10. Hva returnerer funksjonen under dersom den kalles med f(10)

```
f(x) {
    a = 0
    b = 1
    c = 1
    i = 0
    while (i < x) {
        c = a + b
```

```
        a = b
        b = c
        i = i + 1
    }
    return a
}
```

A. 34

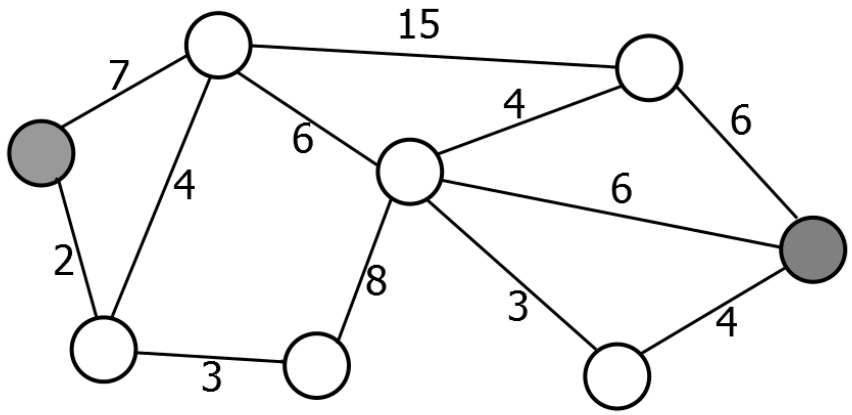
B. 55

C. 64

D. 72

Løsningskommentar: Funksjonen beregner fibonacci-tallene.

11. I datasammenheng er en *graf* en samling av *noder* (punkter; her tegnet som sirkler) og *kanter* (streker) mellom nodene. Vi vil i denne oppgaven se på grafer der hver kant har en *lengde*, som her er markert med tallene ved siden av strekene (lengden av kanten har ingen sammenheng med den "fysiske" lengen av streken den er tegnet med). *Avstanden* mellom to noder er den totale lengden av den korteste stien (sekvensen av kanter) mellom dem.



Hva er avstanden mellom de to fargelagte nodene i tegningen over?

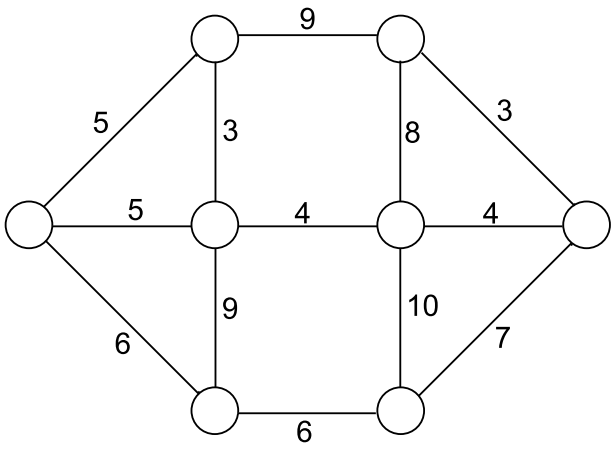
A. 18

B. 19

C. 20

D. 21

12. *Diameteren* til en graf er definert som avstanden mellom de to nodene i grafen som er lengst unna hverandre. (Merk at avstanden mellom to noder forstatt er lengden av den korteste stien mellom dem). Hva er diameteren til grafen under?



A. 14

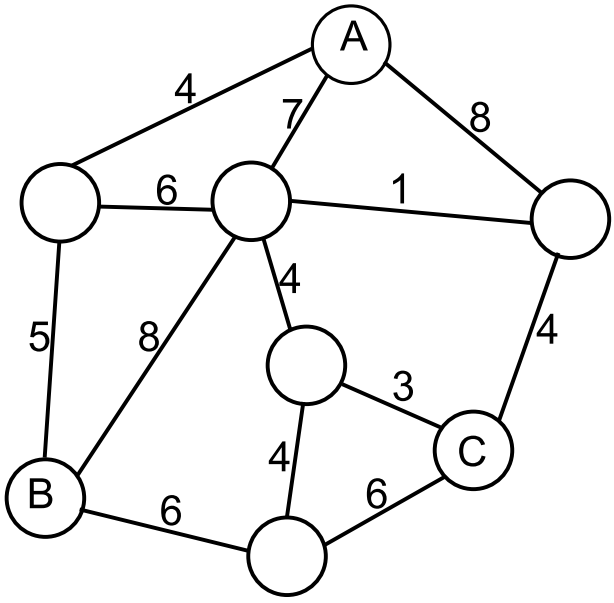
B. 15

C. 16

D. 17

Løsningskommentar: Nodene med størst avstand er den øverst til venstre og den nederst til høyre. Avstanden mellom disse er $5 + 6 + 6 = 17$.

13. Grafen under viser et kart over byer og tettsteder. Nodene markert med A, B og C er byer, og de andre nodene er tettsteder. Strekene viser veier mellom byene, og tallene er lengden på veiene i kilometer. Man ønsker å utvide noen av veiene til å bli motorveier, slik at det vil bli mulig å reise mellom alle de tre byene ved å bare bruke motorveier. Hva er det minste antall kilometer med vei som må bygges om til motorveier for å få til dette?

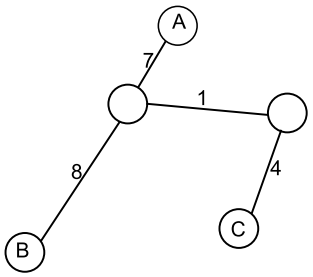


A. 12

B. 20

C. 27

D. 31



Løsningskommentar:

14. Funksjoner i programmeringsspråk kan også inneholde kall til seg selv. Dette kalles *rekursive* funksjoner. Når en rekursiv funksjon kaller seg selv, starter en ny utgave av den samme funksjonen, og den gamle utgaven venter til den nye er blitt ferdig.

    Den rekursive funksjonen under kan brukes til å beregne eksponensialer x^n hvor n er et heltall.

```
eksp(x, n)
    if (n == 0) {
        return 1
    }
    halvN = n / 2
    halvEksp = eksp(x, halvN)
    if (n % 2 == 0) {
        return halvEksp * halvEksp
    } else {
        return halvEksp * halvEksp * x
    }
}
```

    Merk:

    - `==` sjekker om to verdier er like
    - `%` er modulo-operatoren: `a % b` gir resten etter å ha delt `a` på `b`; f.eks. vil `26 % 7` bli 5.
    - `/` er heltallsdivisjon, og utfører avrunding **nedover** til nermeste heltall. F.eks. er `8 / 3 == 2`.

    Totalt hvor mange multiplikasjonsoperasjoner vil bli utført dersom man kaller funksjonen `eksp` med $n = 87$

    A. 7

    B. 9

    C. 12

    D. 14

Løsningskommentar:

Kall	Kaller	Multiplikasjoner
eksp(x, 87)	eksp(x, 43)	2
eksp(x, 43)	eksp(x, 21)	2
eksp(x, 21)	eksp(x, 10)	2
eksp(x, 10)	eksp(x, 5)	1
eksp(x, 5)	eksp(x, 2)	2
eksp(x, 2)	eksp(x, 1)	1
eksp(x, 1)	eksp(x, 0)	2
eksp(x, 0)	-	-

Generelt kan man se at hvis n har a siffer når man skriver det i binært, og b av disse sifferene er 1, så vil man bruke $a + b$ multiplikasjoner. ($87 = 1010111_b$).
Denne måten å halvere eksponenten for hvert kall er en effektiv teknikk som man ofte bruker for å beregne eksponenter av matriser.

15. Et *array* er en samling med et bestemt antall elementer. `A[0]` er det første elementet, `A[1]` er det andre osv.

    Du har kommet over en gammel sorteringsfunksjon 'sorter'. Denne funksjonen tar inn en array med nøyaktig 6 elementer, og sorterer elementene i stigende rekkefølge ved å gjentatte ganger bytte om på to elementer dersom de står feil i forhold til hverandre. Dessverre så har noen strøket over de 4 siste tallene i funksjonen, slik at funksjonen ikke lenger fungerer.

```
byttRiktig(int[] A, int p1, int p2) {
    int v1 = A[p1]
    int v2 = A[p2]
    if (v1 > v2) {
        //tallene står feil. Bytt om.
        A[p1] = v2
        A[p2] = v1
    }
}

sorter(int[] A) {
    byttRiktig(A, 2, 3)
    byttRiktig(A, 3, 5)
    byttRiktig(A, 1, 4)
    byttRiktig(A, 1, 2)
    byttRiktig(A, 1, 3)
    byttRiktig(A, 0, 4)
    byttRiktig(A, 0, 2)
    byttRiktig(A, 2, 5)
    byttRiktig(A, 0, 3)
    byttRiktig(A, 2, 4)
    byttRiktig(A, 0, 1)
    byttRiktig(A, 3, 4)
    byttRiktig(A, ■, ■)
    byttRiktig(A, ■, ■)
}
```

Hva er summen av de fire tallene som er strøket over?

A. 8

B. 10

C. 12

D. 14

Løsningskommentar: Denne måten å sortere tall på kalles et *sorteringsnettverk*. I denne oppgaven så kan man prøve ut en del forskjellige rekker for å se hvordan de kallene man har til byttRiktig oppfører seg. Hvis man f.eks. prøver å sortere tallrekken $[6, 5, 4, 3, 2, 1]$ så ser man at man ender opp med $[1, 2, 4, 3, 6, 5]$. For at de to siste kallene til `byttRiktig` skal sortere rekken så må de være `byttRiktig(A, 2, 3)` og `byttRiktig(A, 4, 5)` i en eller annen rekkefølge. Merk at dette ikke beviser at sorterningsnettverket vil sortere alle rekker - men det holder til å bevise at dersom det er mulig å fullføre sorteringsnettverket med bare to kall så må dette være kallene.

16. Funksjonene `f` og `g` er definert som følger

```
f(x) {
    s = 0
    a = 1
    while (a < x) {
        s = s + 1
        a = a * 2
    }
    return g(x, s)
}
g(a, b) {
    if (a < b) {
        return a + g(a,b-1)
    } else if (b < a) {
        return b + g(a-1,b)
    } else if (a == 0) {
        return 0
    } else {
        return g(a-1, b+1) + 1
    }
}
```

Hvis `t = f(2016)`, hva er siste siffer i `t`?

A. 0

B. 1

C. 2

D. 6

Løsningskommentar: Løkken i `f` setter `a` til minste toerpotens som er større enn eller lik x, og s slik at $2^s = a$. Dette får vi ved $a = 2^{11} = 2048$. Deretter kalles `g(2016, 11)`. `g` er en veldig komplisert måte å bergene $a \times b$ på, så den returnerer 22176.

Svarark

Oppgave	**A**	**B**	**C**	**D**	Poeng (til bruk for læreren)
1		X			
2	X				
3	X				
4			X		
5			X		
6				X	
7			X		
8	X				
9	X				
10		X			
11	X				
12				X	
13		X			
14			X		
15				X	
16				X	
Poengsum					

Til læreren: Husk at korrekt svar gir 4 poeng, feil svar gir 0 poeng, og fraværende svar gir 1 poeng.