

Denne øvingen er en to-ukers øving (prosjekt) og inneholder én stor oppgave.  
Les oppgaveteksten nøye, og ta deg god tid.

## 1 Skumleskogen

Som del av din ferd til Dragvoll idrettssenter<sup>®</sup> for å avlegge eksamen i ITGK er du nødt til å traversere Skumleskogen. Skumleskogen er, som alle skumle skoger, formet som en labyrint (figur 1). Å komme inn i skogen er ikke noe problem, men siden utgangen av skogen er låst, er det er ikke mulig å forlate uten å finne de forskjellige nøklene gjemt i de aller skumleste hjørnene av skogen. Som om ikke det var nok så huser skogen noen skikkelig skumle skapninger, nærmere bestemt søte menneskespisende kaniner. Hvis du skulle være så uheldig å møte på en kanin så ville jo det bety en garantert død, men heldigvis gir menneskespisende kaniner fra seg en ekkel stank som kan kjennes noen meter unna.

Hvordan kan du løse dette problemet og komme deg til eksamen før kl 08:50?<sup>1</sup>

### Oppgavebeskrivelse

I denne oppgaven skal du skrive et program som løser et labyrintspill representert som et tre. Målet med spillet er å komme seg fra inngangsnoden til utgangsnoden, men på veien til utgangsnoden befinner det seg noder som du ikke kan traversere uten å “låse opp” først (låsnoder). For å låse opp låsnoder må du først samle nøkler ved besøke nøkkelnoder i treet og “plukke opp nøkkelen”. Noen ganger så betyr dette at du må snu og utforske en annen retning i jakt på en nøkkel. I tillegg til låsnoder finnes det såkalte superlåsnoder som krever to nøkler for å bli låst opp. Til slutt inneholder treet kanin-noder, som er dødelige (dvs. at du taper spillet hvis du besøker de), og stank-noder, som har den funksjon at de alltid befinner seg ved siden av kanin-noder.

Figurene 1, 2, og 3 illustrerer tre forskjellige representasjoner av den samme instansen (dvs. brettet) av spillet. Den siste figuren, figur 3 er en mye enklere representasjon enn det grafiske bildet i figur 1.

I stedet for å skrive kode som traverserer dette treet, skal du bruke et rammeverk, dvs. en samling med ferdige funksjoner, for å løse denne oppgaven. Rammeverket er implementert i fila `skumleskogen.py` som er lagt ved øvingen, og består av en rekke funksjoner som utfører forskjellige handlinger eller gir informasjon om noden man er i. Rammeverket holder styr på hvor man er i treet og sjekker alltid hvorvidt en handling du prøver å utføre er lovlig. Rammeverket er beskrevet i detalj i tabell 2. Husk å inkludér rammeverket ved å skrive “`from skumleskogen import *`” i starten av svarfilen din.

Den interne representasjonen som rammeverket jobber med er det binære treet i figur 3 som består av noder av forskjellige typer<sup>2</sup>. De forskjellige nodene og deres funksjon er beskrevet i tabell 1.

<sup>1</sup>Retorisk spørsmål.

<sup>2</sup>Alternativt kan du se på det som at nodene inneholder forskjellig informasjon/data.

## Løsningshint

Før du begynner å skrive programkode er det anbefalt å først løse problemet med penn og papir. Etterpå kan du transformere penn-og-papir-algoritmen din til programkode.

Det finnes flere måter å løse dette problemet på, men en rett-fram måte er å holde styr på hvilke barn (høyre/venstre) man har besøkt for hver node man møter på, og gå tilbake og prøve en annen retning hvis man har besøkt begge barna uten å komme i mål (dvs. at noden er en blindvei eller at utgangene er stengt). Vi kaller denne samlingen av besøkt-informasjon for hver node for hukommelsen til algoritmen.

(Node)hukommelse kan du implementere ved hjelp av en Python-dictionary som mapper nodenummer med en annen dictionary (eller liste) med en boolsk verdi for hvert barn.

### Eksempel 1: Representasjon av hukommelse

---

- Definer denne en eller annen plass, f.eks på toppen av fila:

```
hukommelse = {}
```

- For hver ny node som ikke er i hukommelsen:

```
n = nummer()
if n not in hukommelse:
    hukommelse[n] = { "venstre": False, "hoyre": False }
```

- Oppdater nøklene venstre og høyre etter hvert som du besøker de retningene:

```
hukommelse[n]["venstre"] = True # f.eks
```

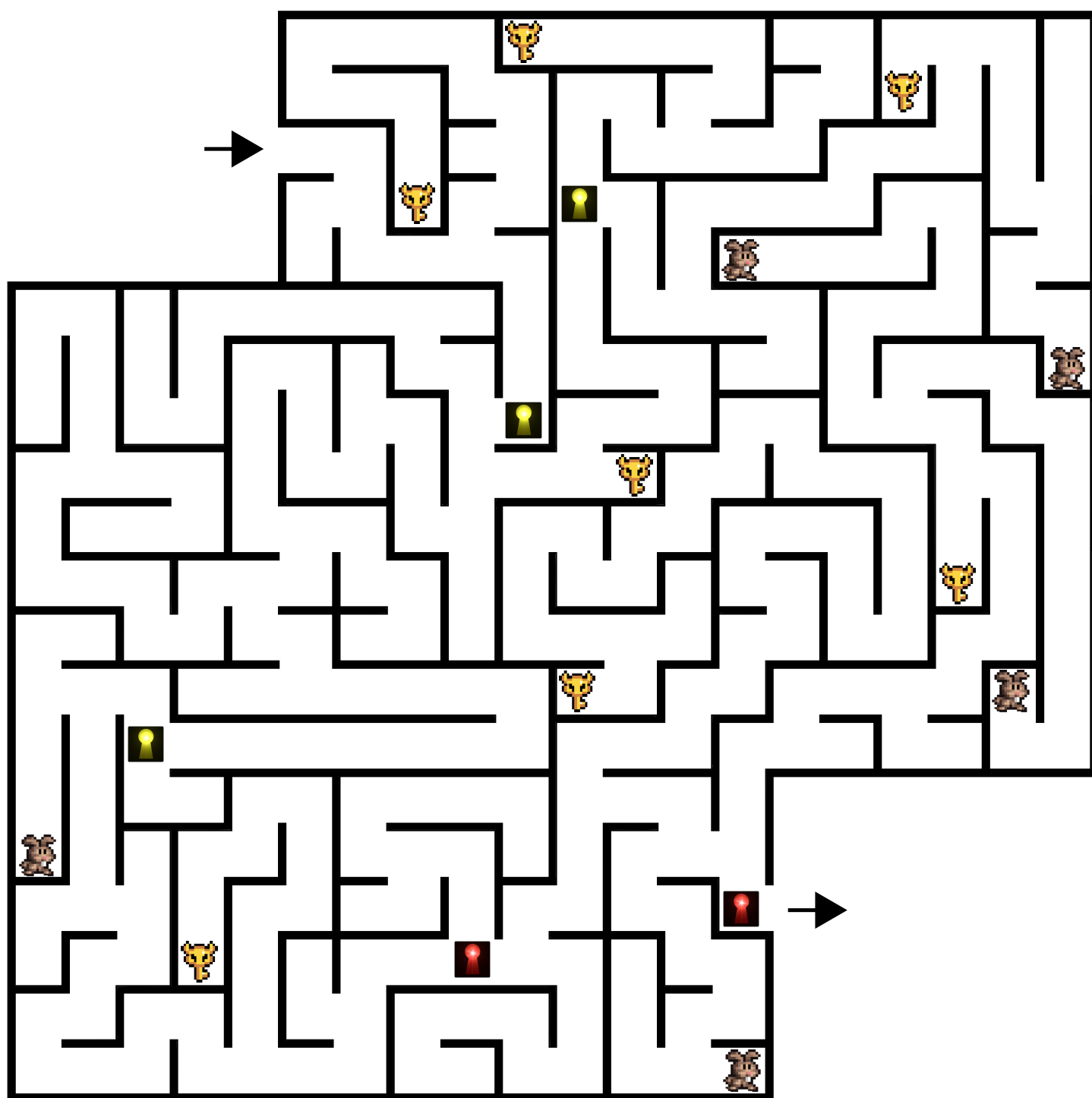
- For å nullstille hukommelsen:

```
hukommelse = {}
```

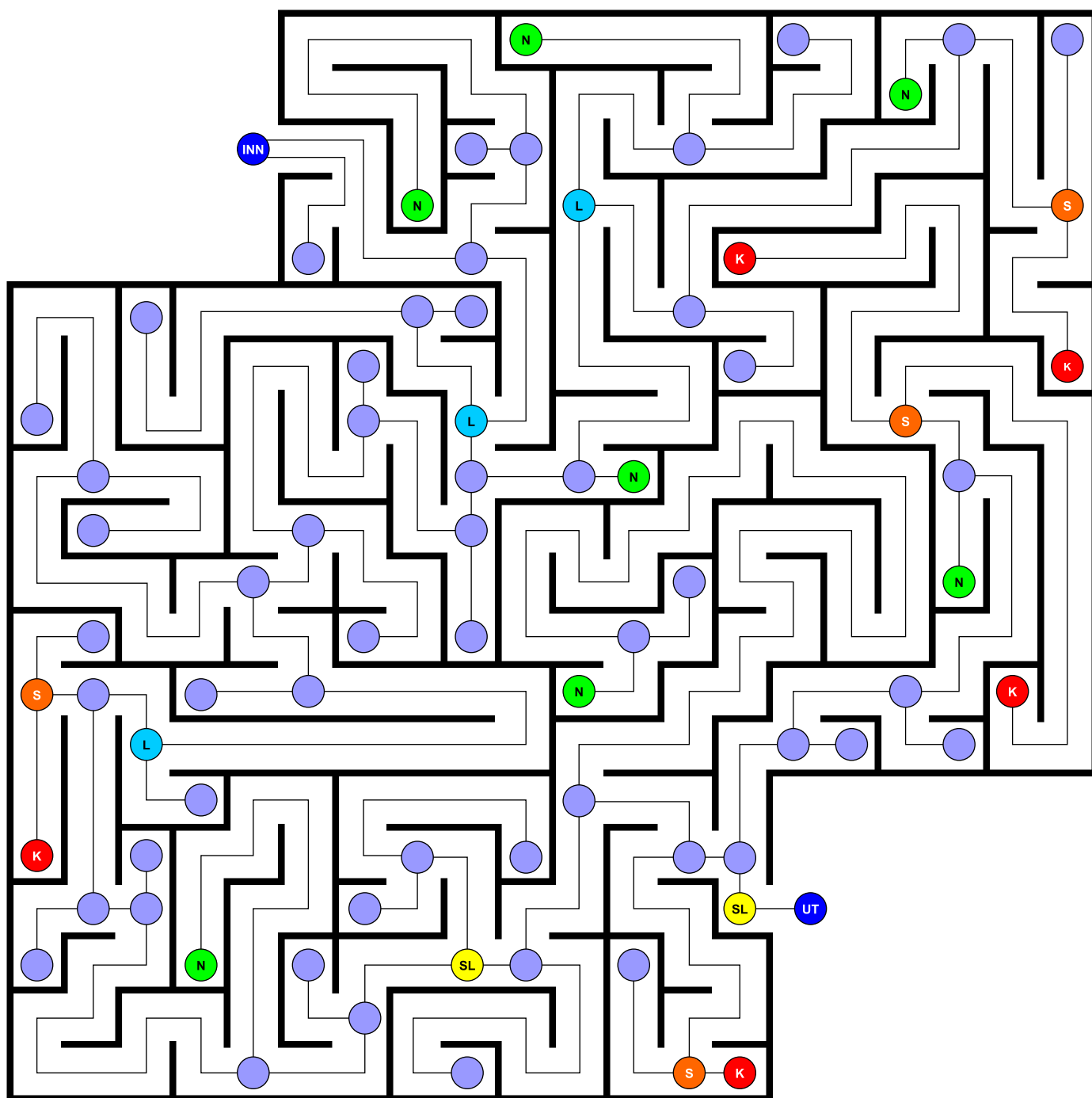
---

**NB:** Hvis du plukker opp en nøkkel er det enklest å glemme all informasjon om nodene (nullstille hukommelsen) siden denne representasjonen kan ha markert stien til en låsnod som en blindvei (fordi du ikke hadde en nøkkel når du fant låsnoden). Nå som du har en nøkkel, er du nødt til å besøke låsnoden igjen. Alternativt kan du oppdatere hukommelsen på en slik måte at stien til låsnoden blir satt som ubesøkt, eller så kan du bruke en helt annen representasjon/algoritme enn den som er anbefalt her.

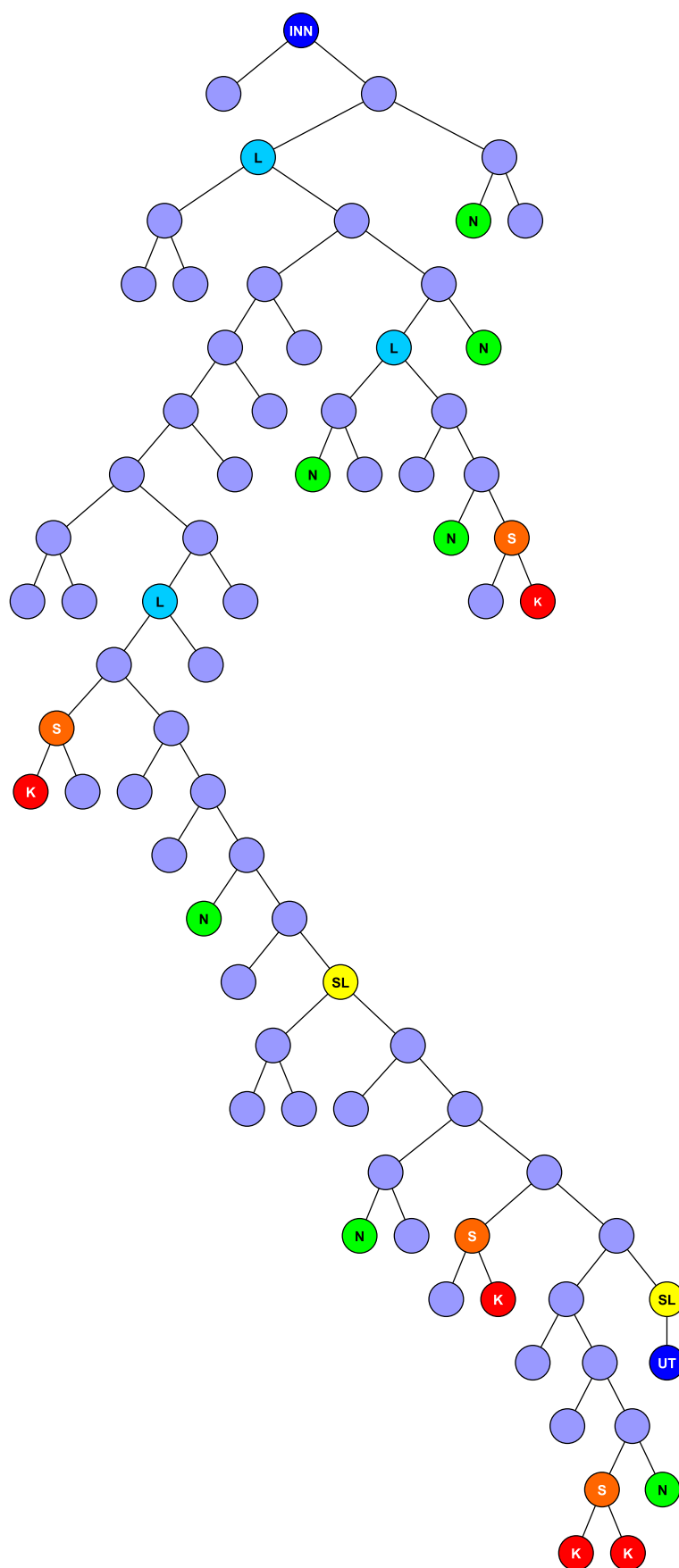
- Skriv et program som importerer `skumleskogen.py` og løser dette problemet (kommer seg ut av skumleskogen) ved hjelp av grensesnittet/rammeverket beskrevet i tabell 2. `skumleskogen.py` inneholder dens samme skogen som er vist i figur 1.
- `skumleskogen2.py` inneholder et alternativt tre som ikke er vist grafisk i denne oppgaven. Importer `skumleskogen2.py` isteden for `skumleskogen.py` og kjør algoritmen din på det nye treet som en test på at algoritmen din fungerer generelt for alle skumle skoger (dvs. at den ikke hardkoder løsningen for treet i figur 1).















Figur 1: Skumleskogen



Figur 2: Hvis vi representer alle blindveier og vendepunkt (kryss) i labyrinten som en node og kobler alle nodene sammen, vil vi ha en representasjon av labyrinten som kan brukes til å løse problemet programatisk.



Figur 3: Ved å omrokere litt på disse nodene ser man at de faktisk former et binært tre. Dette er på grunn av at labyrinthen er hva som kalles en klassisk labyrinth.

Bilde	Node	Nodetype	Beskrivelse
		Vanlig	Dette er helt vanlige noder. Du kan gå til høyre, venstre, eller tilbake fra en slik node.
		Nøkkel	Denne noden inneholder én nøkkel, men er ellers som en vanlig node. Noden blir til en vanlig node om du plukker opp nøkkelen.
		Lås	Denne noden er en låsnode. Du må låse opp låsnoden med én nøkkel for å kunne gå videre (men du kan gå tilbake).
		Superlås	Denne noden er en superlåsnode. Disse fungerer som vanlige låsnoder, men du må bruke to nøkler for å låse den opp.
		Kanin	Denne noden inneholder en kanin. Hvis du havner på denne taper du spillet, og algoritmen din er ganske sikkert feil.
		Stank	Denne noden stinker, noe som betyr at en eller flere noder ved siden av inneholder en kanin.
		Inngang	Denne noden representerer inngangen til skogen.
		Utgang	Denne noden representerer utgangen av skogen. Hvis du havner på denne noden så kan du <b>gå ut</b> og vinne spillet.

Tabell 1: Beskrivelse av nodetypene

Handling	Type	Beskrivelse
<b>nummer()</b>	Alle	Returnerer det (unike) nummeret til noden. Dette kan du bruke for å holde styr på om du har besøkt en node før.
<b>er_vanlig()</b>	Alle	Returnerer <b>True</b> om noden er en vanlig node; <b>False</b> ellers.
<b>er_nokkel()</b>	Alle	Returnerer <b>True</b> om noden er en nøkkelnod; <b>False</b> ellers.
<b>er_laas()</b>	Alle	Returnerer <b>True</b> om noden er en låsnod eller superlåsnod; <b>False</b> ellers.
<b>er_superlaas()</b>	Alle	Returnerer <b>True</b> om noden er en eller superlåsnod; <b>False</b> ellers.
<b>er_inngang()</b>	Alle	Returnerer <b>True</b> om noden er inngangsnoden; <b>False</b> ellers.
<b>er_utgang()</b>	Alle	Returnerer <b>True</b> om noden er utgangsnoden; <b>False</b> ellers.
<b>er_stank()</b>	Alle	Returnerer <b>True</b> om noden er en stanknod; <b>False</b> ellers.
<b>gaa_hoyre()</b>	Alle	Gå til den venstre noden. Hvis det ikke er en (åpen) vei til venstre så returnerer denne funksjonen <b>False</b> .
<b>gaa_venstre()</b>	Alle	Gå til den venstre noden. Hvis det ikke er en (åpen) vei til venstre så returnerer denne funksjonen <b>False</b> .
<b>gaa_tilbake()</b>	Alle	Gå tilbake til foreldrenoden. Hvis du er på inngangsnoden så returnerer denne funksjonen <b>False</b> .
<b>plukk_opp()</b>	Nøkkel	Plukk opp en nøkkel. Nøkkelnoden blir til en vanlig node etter at du kaller denne funksjonen.
<b>laas_opp()</b>	Lås, Superlås	Lås opp en låsnod eller superlåsnod. Vanlige låsnoder krever én nøkkel mens superlåsnoder krever to nøkler for å bli låst opp. Ved suksess blir noden til en vanlig node. Ellers returnerer funksjonen <b>False</b> .
<b>gaa_ut()</b>	Utgang	Flykt fra skumleskogen. Hvis du utfører denne handlingen så vinner du spillet.

Tabell 2: Rammeverk for denne oppgaven