UNIVERSITETET
I OSLO

IN2010: Forelesning 11

Kombinatorisk søking

Beregnbarhet og kompleksitet

UNIVERSITETET
I OSLO

KOMBINATORISK SØKING

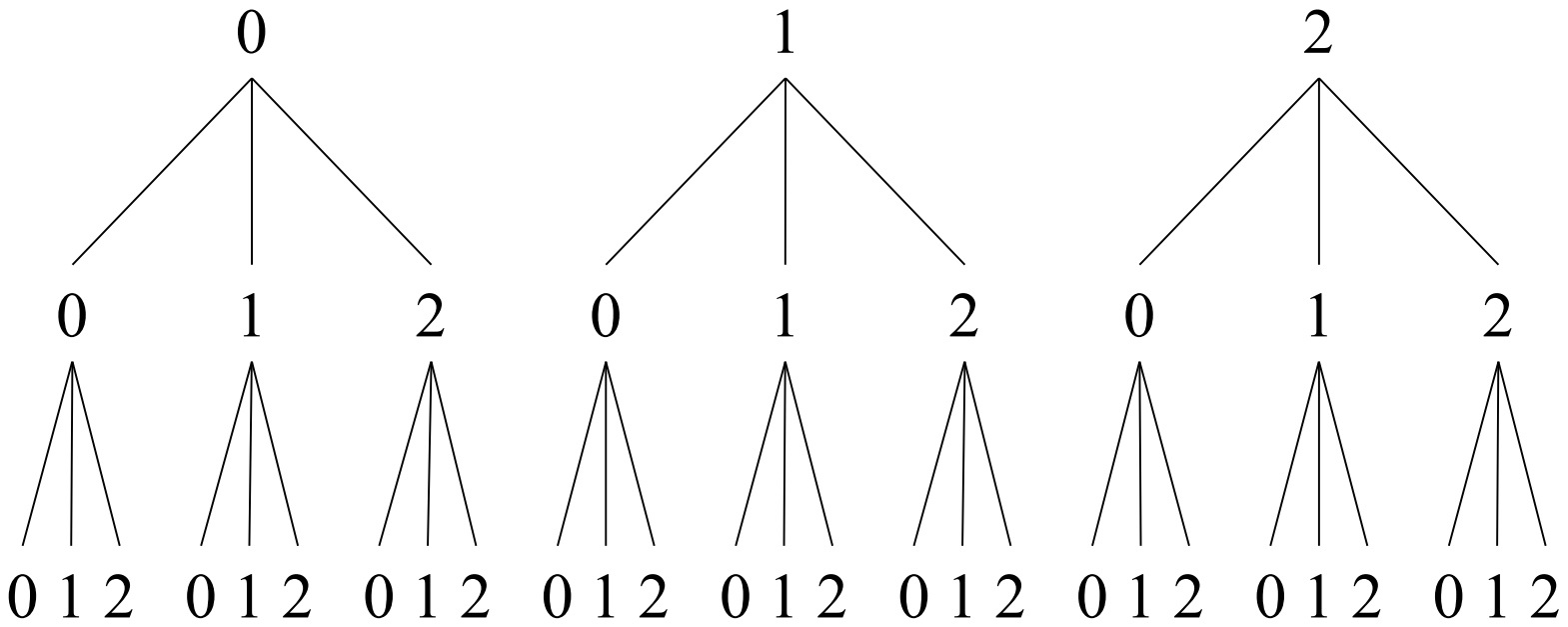
Oversikt

- Generering av permutasjoner
  - Lett: Sekvens-generering
  - Vanskelig: Alle tallene må være forskjellige
- Eksempel: Finne korteste reiserute
- Eksempel: Dronning-oppgaven

Les notatet *Om å lete gjennom «alle muligheter»* av Stein Krogdahl og Arne Maus.

Sekvens-generering

Alle mulige sekvenser av lengde tre av tallene 0, 1 og 2:



Sekvens-generering: mulig implementasjon

Oppgave:
Generaliser dette til å generere alle mulige sekvenser av lengde n av tallene fra 0 til n-1.

```
int[] sekv = new int[3];


// Startkall: gen(0);

void gen(int pos) {
    for (int siffer = 0; siffer < 3; siffer++) {
        sekv[pos] = siffer;
        if (pos < 2) {
            gen(pos + 1);
        } else {
            < Skriv ut innholdet i sekv. >
        }
    }
}
```

Generalisering

UNIVERSITETET I OSLO

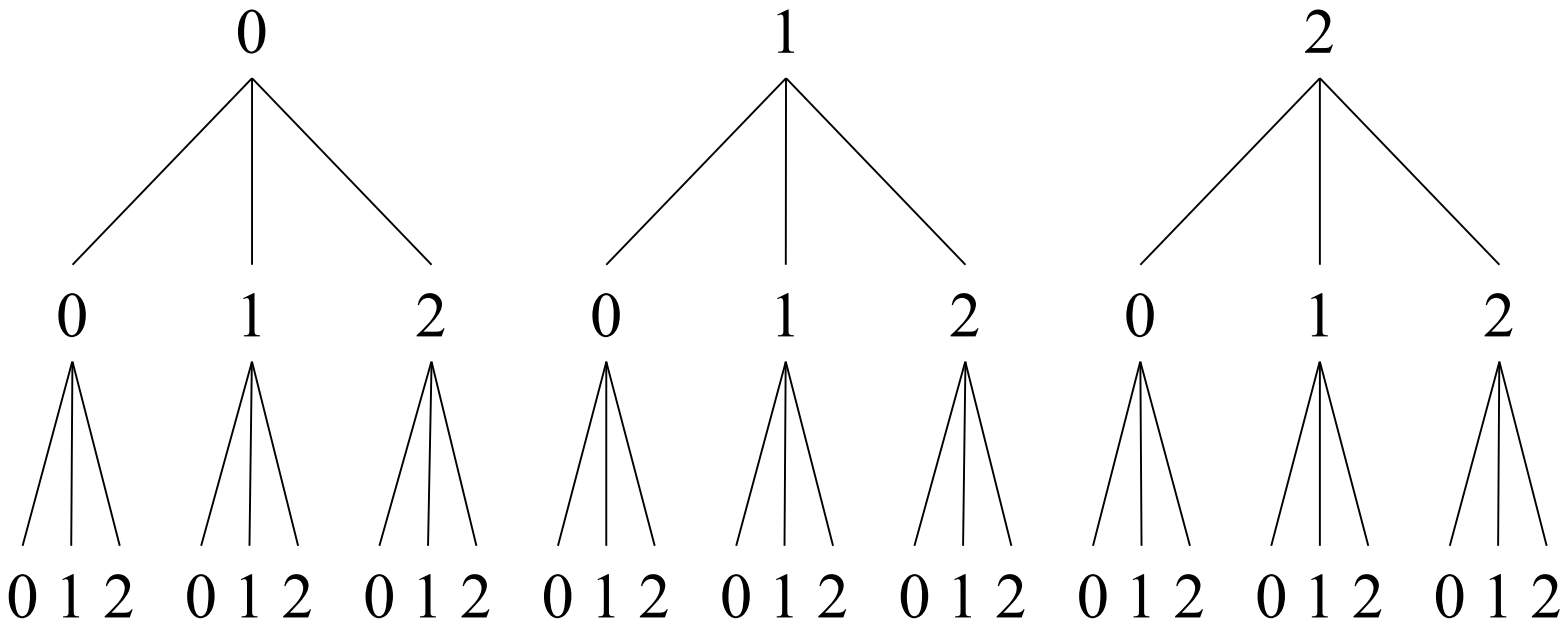
Alle mulige sekvenser av lengde n av tallene fra 0 til n-1:

```
class Gen {
    int[] sekv; int n;

    Gen(int i) { n = i; sekv = new int[n]; }

    void gen(int pos) {
        for (int siffer = 0; siffer < n; siffer++) {
            sekv[pos] = siffer;
            if (pos < n - 1) {
                gen(pos + 1);
            } else {
                < Skriv ut innholdet i sekv. >
            }
        }
    }
}
```

Oppgave:
Hva må til for å endre dette slik at det bare genereres sekvenser hvor alle sifrene er ulike?

Permutasjoner

Alle mulige kombinasjoner av tallene 1-3 slik at hvert tall bare er brukt en gang:



Permutasjoner

Alle mulige permutasjoner av tallene fra 0 til n-1:

```
class Gen {
    int[] sekv; int n; boolean[] brukt;
    Gen(int i) {
        n = i; sekv = new int[n]; brukt = new boolean[n];
        for (int j=0; j<n; j++) { brukt[j] = false; }
    }

    void gen(int pos) {
        for (int siffer = 0; siffer < n; siffer++) {
            if (!brukt[siffer]) {
                brukt[siffer] = true;
                sekv[pos] = siffer;
                if (pos < n - 1) { gen(pos + 1); }
                else { < Lever sekv til videre bruk. > }
                brukt[siffer] = false;
            }
        }
    }
}
```

Kombinatorisk søking

Har: Et endelig antall elementer.
Skal: Finne en rekkefølge, gjøre et utplukk, lage en oppdeling, …

Eksempler:

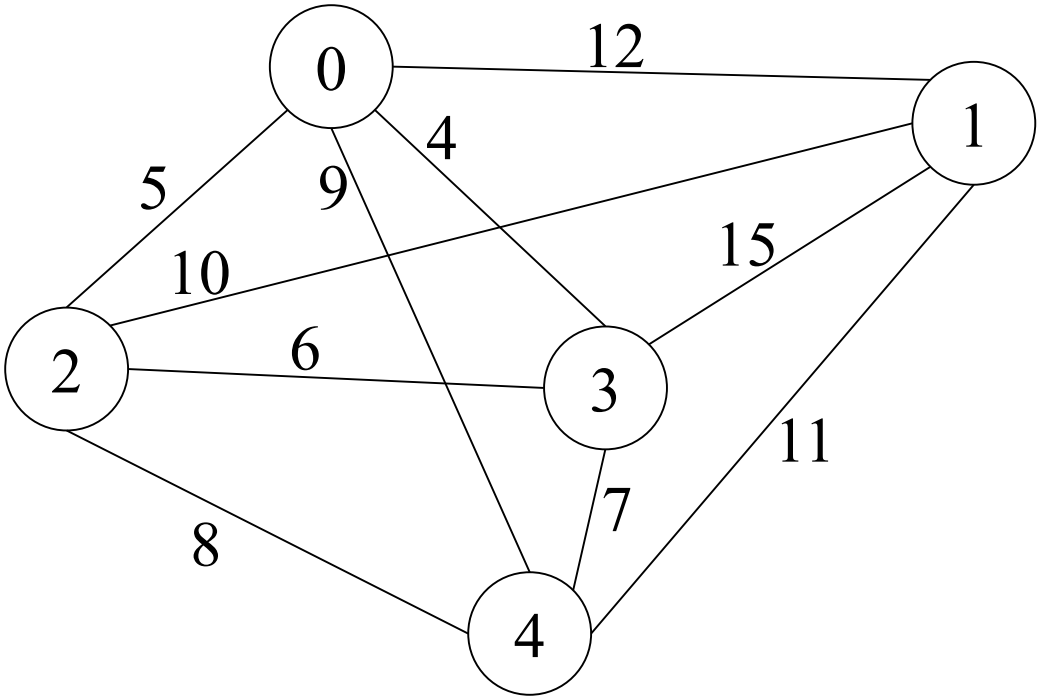
- Plassere 8 dronninger på et sjakkbrett slik at ingen av dronningene kan slå hverandre.
- Finne korteste rundtur blant 'n' byer hvor hver by bare besøkes en gang (avstanden mellom ethvert par av byer er kjent).

Vi har da problemer hvor vi må:

1. Lage alle interessante kombinasjoner.
2. Teste om en kombinasjon er en løsning på det aktuelle problemet.

NB: Ofte slår vi disse sammen, slik at vi bare lager kombinasjoner som er potensielle løsninger.

Eksempel: Korteste reiserute



Korteste vei: 36.0

Korteste reiserute (by nr): 0 2 1 4 3 0

Korteste reiserute

UNIVERSITETET I OSLO

- nytt navn kortesteVei, variabelnavn ”by” i stedet for ”siffer”
- startkall: kortesteVei(1); sekv[0] = 0 er startbyen

```
void kortesteVei(int pos) {
    for (int by = 1; by < n; by++) {
        if (!brukt[by]&&
            (lengde + avst[sekv[pos-1]][by] < besteLengde)  ){
            brukt[by] = true;
            lengde += avst[sekv[pos-1]][by];
            sekv[pos] = by;
            if (pos < n - 1) {
                kortesteVei(pos + 1);
            } else if (lengde + avst[by][0] < besteLengde) {
                besteLengde = lengde + avst[by][0];
                < Lever sekv til videre bruk >
            }
            brukt[by] = false;
            lengde -= avst[sekv[pos-1]][by];
        }
    }
}
```

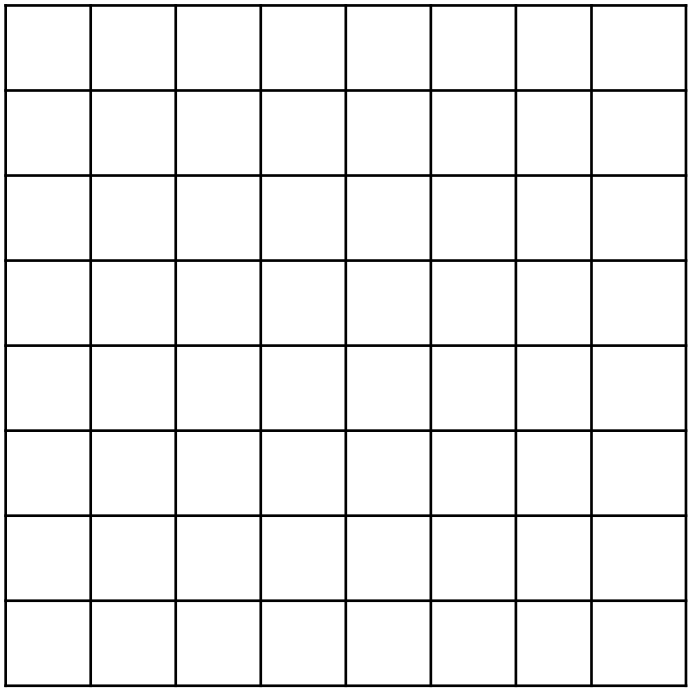
float lengde: lengden på nåværende rute

float[][] avst: avstanden mellom ethvert par av byer

float besteLengde: korteste hittil

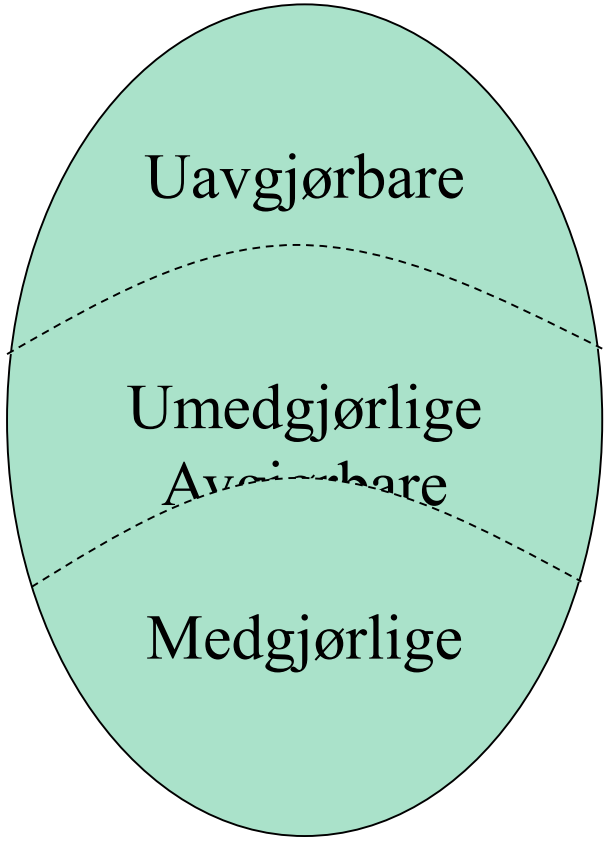
Dronningproblemet

- Hvordan plassere n dronninger på et nxn-sjakkbrett slik at ingen av dronningene kan slå hverandre?

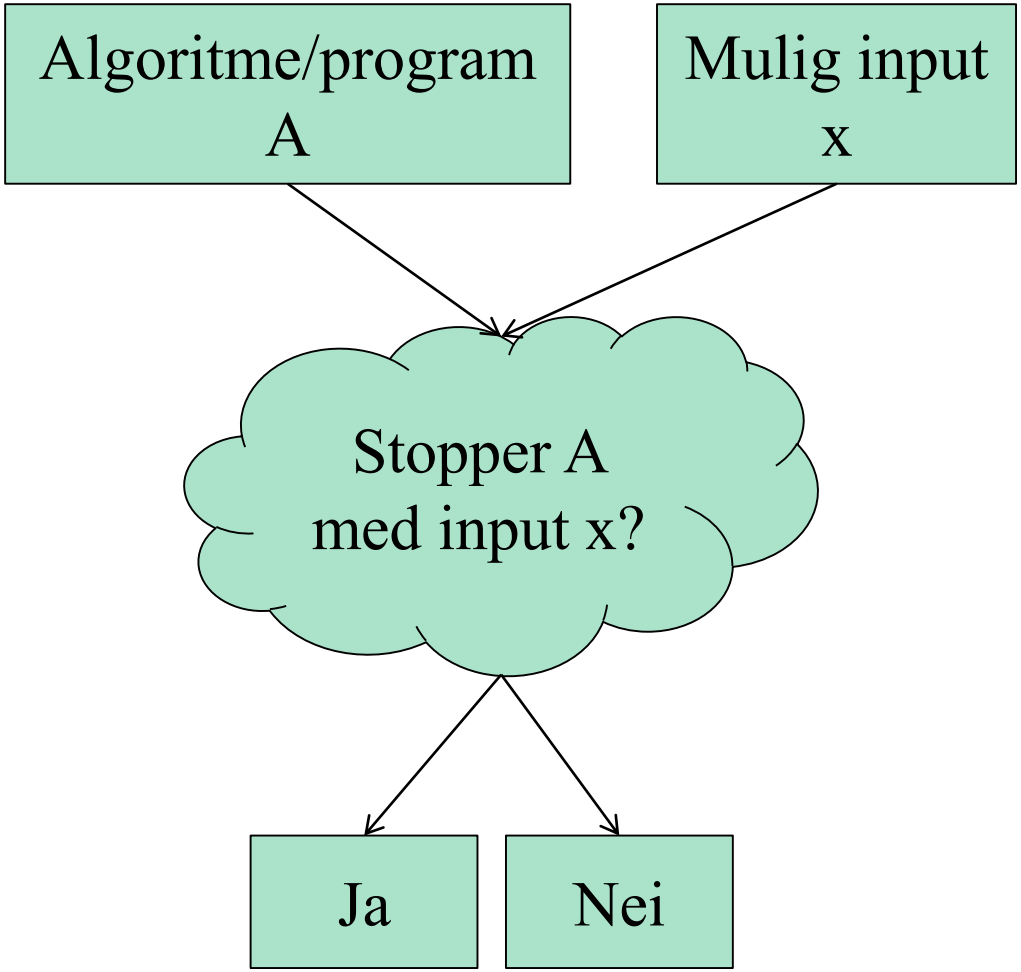


BEREGNBARHET OG KOMPLEKSITET

Klasser av problemer



Et uavgjørbart problem: stoppeproblemet



Klassene P og NP

- P = polynomial-time
- Et problem er i P dersom en løsning kan finnes i polynomisk tid.

Klassene P og NP

- P = polynomial-time
- Et problem er i P dersom en løsning kan finnes i polynomisk tid.

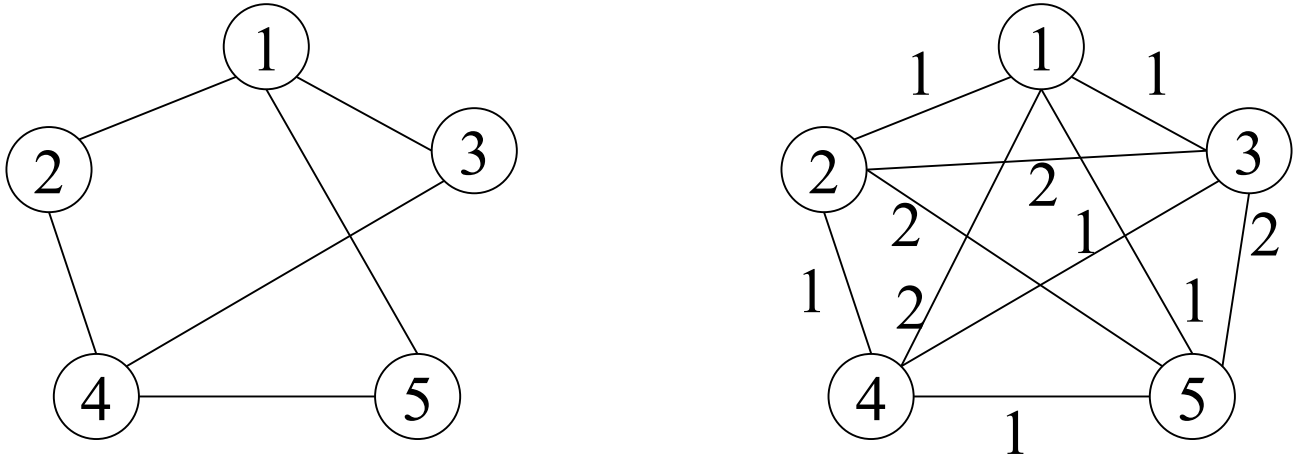
- NP = nondeterministic polynomial-time.
- Et problem er i NP dersom en gitt løsning kan sjekkes i polynomisk tid.

NP-kompletthet

De NP-komplette problemene er de vanskeligste problemene i NP.

Ethvert problem i NP kan reduseres til et NP-komplett problem i polynomisk tid.

Eksempel: Hamiltonsk løkke → Traveling Salesman



VERTEX-COVER

- Gitt en graf, finnes det et delmengde på max k noder slik at alle kanter har minst en node i delmengden.
- *Eksempel:*
  Gitt et nettverk hvor nodene representerer rutere og kantene fysiske forbindelser. Vi ønsker å oppgradere nettverket med (dyre) spesial-rutere for bedre monitorering av nettverket. Er det tilstrekkelig å kjøpe k nye rutere?

SUBSET-SUM

- Gitt et heltall k og en mengde S med n heltall, finnes det er delmengde av S slik at summen av tallene er lik k?
- *Eksempel:*
  Anta at vi har en Web-server og et antall download-forespørsler. For hver forespørsel kan vi lett bestemme størrelsen på den aktuelle filen. Vi ønsker å finne et antall forespørsler slik at summen tilsvarer kapasiteten serveren har i løpet av ett minutt.
  - *NB: Siden problemet er NP-komplett, blir problemet vanskeligere å løse jo mer kapasitet serveren har!*

P = NP???

Neste forelesning: 16. november

REPETISJON