# De eerste programmeeropgave — Chomp
# Algoritmiek voorjaar 2007, Universiteit Leiden

## Toestand-actie-ruimte en strategie

Het tweepersoonsspel Chomp wordt gespeeld op een rechthoekig rooster, bestaande uit $m$ rijen en $n$ kolommen. De rechthoek stelt een reep chocola voor, bestaande uit $m * n$ blokjes, waarvan het blokje linksboven (met coördinaten $(0, 0)$) giftig is. De spelers moeten om de beurt een (niet lege) *hap* nemen. Degene die het giftige blokje opeet heeft uiteraard verloren. Een hap bestaat uit het aanwijzen van een blokje met coördinaten $(i, j)$, en het vervolgens rechtsonder weghalen (en opeten) van het rechthoekige stuk van de reep met het aangewezen blokje als linksbovenhoek. Iets preciezer: verwijderd wordt dus het rechthoekige stuk bestaande uit alle blokjes met coördinaten $(k, \ell)$ waarvoor geldt: $k \geq i$ EN $\ell \geq j$.

Een voorbeeld van een mogelijke beginsituatie, met $m = 3$ en $n = 4$:

```
*  X  X  X      (1,1)        *  X  X  X
X  X  X  X    ---------->    X
X  X  X  X                   X
```

Hierin geeft `*` het giftige blokje chocola aan, en `X` de gewone, niet-giftige blokjes. Veronderstel dat de speler die begint blokje $(1, 1)$ kiest, dan blijft vervolgens een L-vorm over. De tweede speler kiest daarna bijvoorbeeld $(0, 1)$, hetgeen de volgende situatie overlaat:

```
*
X
X
```

Het is nu duidelijk wat de eerste speler moet doen om te winnen: eet de onderste twee blokjes op, dus kies blokje $(1, 0)$), en laat het giftige blokje liggen voor de tegenstander.
Het is bekend (en niet zo moeilijk in te zien) dat het spel winnend is voor de speler die begint, voor elke $m$ bij $n$ (rechthoekige) reep. Er is echter niet bekend wat in het algemeen de winnende strategie is, dus welke happen de speler moet nemen om zeker te winnen. Voor een aantal gevallen is zo'n winnende strategie echter wel te vinden.

Bij deze programmeeropdracht moet een kort **verslag** gemaakt worden, waarin de volgende vragen/opdrachten beantwoord/uitgewerkt moeten worden. Het verslag is bij voorkeur getypt (bijv. in LaTeX), maar mag ook handgeschreven zijn; als het maar duidelijk is, zowel wat betreft inhoud als leesbaarheid.

**1.** Teken de toestand-actie-ruimte voor het geval $m = 2$ en $n = 3$. Geef bij elke toestand aan of deze winnend is of niet voor de speler die aan de beurt is. Bepaal zo de winnende zet en de winnende strategie.
**2.** Analyseer nu het algemenere geval met $m = 2$ en $n$ willekeurig. Wat is in dit geval de winnende strategie? Geef duidelijk toelichting.
**3.** Als er als tussenstand een L-vorm overblijft is die winnend voor de speler die aan de beurt

is als de poten van de L niet even lang zijn, en anders verliezend. Leg dit uit en geef voor de winnende situatie aan hoe de speler kan winnen.
**3.** Dezelfde vraag als **1.**, maar nu voor het geval $m = 3$ en $n = 3$. Je hoeft hier een toestand waarvan je al gezien of bewezen hebt dat die winnend dan wel verliezend is niet meer helemaal uit te werken.
**4.** Bekijk nu het algemene geval dat de reep een vierkant is (dus $m = n$ met $m, n \geq 2$). Wat is in dit geval de winnende strategie voor de beginnende speler?

## Programma

Er moet een (brute force) programma worden geschreven dat voor Chomp met willekeurige $m$ en $n$ de winnende zet oplevert. De resultaten moeten voor zo veel mogelijk waarden van $m$ en $n$ in een tabelletje in het verslag worden opgenomen. Verder moet het spel gespeeld kunnen worden tegen de computer.

Het programma vraagt de gebruiker om een waarde voor $m$ en $n$ in te voeren. Vervolgens berekent het programma wat een/de winnende zet is door voor alle mogelijke directe vervolgstanden een *recursieve* boolese functie `winst` aan te roepen, die bepaalt of de betreffende stand winnend is *voor de aan de beurt zijnde speler* of niet. Merk op dat een stand winnend is voor degene die aan de beurt is dan en slechts dan als een van zijn directe vervolgstanden niet winnend is voor de tegenstander Druk de/een winnende zet af op het beeldscherm.
Vervolgens kan het spel gespeeld worden tussen de gebruiker (die begint) en de computer. Zetten van de gebruiker worden ingevoerd door de coördinaten van het blokje $(i, j)$ te geven. De computer moet optimaal spelen, dat wil zeggen: als er een winnende zet voor hem bestaat doet hij deze. Voor het vinden van zo'n winnende hap kan de functie `winnendezet` weer worden aangeroepen. Als er in een zekere toestand geen winnende zet bestaat, mag de computer een willekeurige zet doen: bedenk maar wat. Zijn er meer winnende zetten mogelijk, kies er dan een.

Opmerkingen:

- Kies een verstandige klasse-structuur (genaamd `chomp`), met als member-functies in elk geval de recursieve functie `winst`, een functie `winnendezet` die de/een winnende zet bepaalt en `drukaf` (voor het afdrukken van een stand). Denk ook aan de constructor en de destructor (indien nodig).

- Laat de memberfunctie `winnendezet` bijvoorbeeld $(-1, -1)$ opleveren als er geen winnende zet bestaat. Overigens kun je natuurlijk ook de functie `winnendezet` zelf recursief maken en dus recursief een winnende zet laten bepalen; dan heb je geen aparte functie `winst` nodig.

- Implementeer het Chomp-rooster (= de reep) bijvoorbeeld met behulp van een $MaxM$ bij $MaxN$ array, waarbij $MaxM$ en $MaxN$ niet te groot zijn (denk aan orde van grootte van het totaal aantal blokjes 25). Er zijn overigens wel efficiëntere implementaties van het rooster te bedenken.

- Bij het invoeren van $m$ en $n$ en de zetten van de gebruiker mag gewoon `cin` worden gebruikt (je mag aannemen dat er een geheel getal is ingevoerd). Er moet wel worden gecontroleerd dat de gebruiker niet te grote $m$ en $n$ kiest en dat hij een geldige zet invoert,

dus bijvoorbeeld dat $m \leq MaxM$, $0 \leq i \leq m-1$ en dat het blokje $(i, j)$ nog niet opgegeten is.

- Het werkende programma mag er op het scherm eenvoudig uitzien (gebruik bij voorkeur alleen `iostream.h`), maar moet natuurlijk wel duidelijk zijn.
- Boven elke functie moet een commmentaarblokje komen met daarin een (zeer) korte beschrijving van wat de functie doet. Noem daarin ook de gebruikte parameters en geef aan hoe ze eventueel veranderd worden door de functie. Let ook op de layout (consequent inspringen) en op het overige commentaar bij de programmacode (alleen zinvol en kort commentaar).
- Informatie over de programmeeropdracht is te vinden op:
  `http://www.liacs.nl/home/jlaros/edu/alg.shtml`.
  Voor meer informatie over het vak, laatste wijzigingen, etcetera, raadplege men:
  `http://www.liacs.nl/home/graaf/ALGO/algo2007.html`

**Uiterste inleverdatum**: vrijdag 16 maart 2007. Voor elke week te laat inleveren gaat er een punt van het cijfer af.
Het programma per e-mail sturen aan de hoofdnakijker: `jlaros@liacs.nl`. Listing en verslag moeten op papier worden ingeleverd in de daartoe bestemde doos met opschrift Algoritmiek in de postkamer van Informatica, kamer 156. Vermeld overal duidelijk de namen van de makers.

**Normering**: verslag 3; commentaar en layout 1,5; modulaire opbouw en OO: 1,5; werking 4