Programmeeropdracht 1 — Aqualin
Algoritmiek, voorjaar 2025

Inleiding

Brian en Robin kunnen niet meer op het veld uitvechten wie nu de beste is. Daarom hebben ze hun toevlucht genomen tot het bordspel Aqualin. Hierbij leggen de twee spelers om de beurt een steen op een rechthoekig bord. Elke steen wordt gekenmerkt door een kleur en een vorm. Speler 0 (Brian) legt de eerste steen. Hij probeert clusters van stenen op het bord te krijgen van dezelfde kleur. Speler 1 (Robin) probeert juist clusters van stenen op het bord te krijgen van dezelfde vorm. De speler die aan het eind van het spel de grootste cluster van zijn voorkeur (kleur of vorm) op het bord heeft, wint.

De belangen zijn groot, en zeker Brian heeft iets te bewijzen. Hij vraagt jou dan ook om hulp: kun je hem helpen om in elke stand van het spel de optimale zet te vinden? Ook Robin kan wel hulp gebruiken, maar hij beseft dat het vinden van de optimale zet wel eens te moeilijk kan zijn. Wellicht zijn er heuristieken die toch in ieder geval een goede zet opleveren. Toevallig komt ook hij bij jou terecht om dit idee uit te werken. Je vindt het geen probleem om twee heren te dienen, en gaat aan de slag.

Je representeert een steen door twee getallen, waarbij het eerste getal de kleur voorstelt, en het tweede getal de vorm. Een compleet gevuld bord kan er dan bijvoorbeeld uitzien als in Figuur 1.

	0	1	2	3	4	5
0	(0,1)	(3,4)	(3,5)	(2,1)	(1,3)	(5,5)
1	(4,3)	(2,4)	(4,4)	(0,3)	(1,2)	(0,5)
2	(0,2)	(3,2)	(5,3)	(5,4)	(1,0)	(0,0)
3	(4,2)	(4,1)	(2,3)	(3,0)	(5,0)	(2,2)
4	(2,0)	(4,5)	(2,5)	(4,0)	(1,4)	(0,4)
5	(3,1)	(1,5)	(1,1)	(5,1)	(3,3)	(5,2)

?figurename? 1: Voorbeeld van een volledig bedekt 6×6 Aqualin-bord. Brian heeft hier een cluster van drie stenen van kleur 3 (in rood), terwijl Robin een cluster van vijf stenen van vorm 0 (in blauw) heeft.

Uitwerking spelregels

Het bord kent h rijen en b kolommen, met $1 \leq h, b \leq 10$. Rijen zijn van boven naar beneden genummerd van 0 t/m $h-1$. De kolommen van links naar rechts van 0 t/m $b-1$. De spelers mogen elke steen die ze leggen niet op een vakje naar keuze leggen. Om het spel wat eenvoudiger te maken is er namelijk een vaste *vulvolgorde* van de vakjes op het bord. Brian moet als speler 0 een eerste steen op vakje 0 in deze volgorde leggen, daarna legt Robin als speler 1 een steen op vakje 1 in deze volgorde, dan Brian weer op vakje 2, enzovoort. De spelers kunnen wel kiezen welke steen uit hun hand ze op het vastgelegde vakje leggen.

Er zijn net zoveel stenen als er vakjes op het bord zijn. Alle stenen zijn verschillend. Als we het aantal verschillende kleuren k noemen, en het aantal verschillende vormen v, dan hebben we de volgende stenen tot onze beschikking: $(0,0), (0,1), \ldots, (0,v-1), (1,0), (1,1), \ldots, (1,v-$

$1), (2,0), (2,1), \ldots$ Dus de kleuren heten $0, 1, \ldots, k-1$ en de vormen $0, 1, \ldots, v-1$. Er geldt dat $1 \leq k, v \leq 10$. Het is niet per se zo dat $k \cdot v = h \cdot b$. Als we bijvoorbeeld een 4×5 bord hebben, met $v = 6$ verschillende vormen, dan zijn er $k = 4$ kleuren, waarbij de laatste steen in bovenstaande opsomming de steen $(3, 1)$ is.

Elke speler heeft een hand met een aantal stenen waar hij uit kan kiezen. Om variatie mogelijk te maken, zijn de handen niet per se even groot. Brian (speler 0) heeft maximaal k_0 stenen in zijn hand, terwijl Robin (speler 1) er maximaal k_1 heeft. We noemen deze aantallen de *keuzeaantalllen* van de spelers. Als bijna alle stenen op het bord liggen, kan het zo zijn dat Brian en Robin minder stenen in hun hand hebben dan hun keuzeaantal toe zou staan, gewoon omdat er niet genoeg stenen meer over zijn.

De stenen zitten aanvankelijk in de pot. Daar komen ze in een vaste volgorde uit. Bij het begin van het spel worden de eerste a stenen al direct op het bord gelegd, in de volgorde waarin ze uit de pot komen, op de eerste a vakjes van de vulvolgorde. De volgende k_0 stenen (als die er zijn) uit de pot komen in de hand van Brian, de volgende k_1 stenen komen in de hand van Robin. Wanneer een speler in zijn beurt een steen op het bord legt, krijgt hij een nieuwe steen uit de pot, uiteraard zo lang de voorraad strekt.

	0	1	2	3	4
0	11	8	13	14	12
1	6	5	9	4	1
2	10	15	19	16	2
3	7	17	0	3	18

?figurename? 2: Een 4×5 bord met volgorde waarin de vakjes bedekt moeten worden met stenen.

Stel dat we een 4×5 bord hebben, waarbij de vulvolgorde is aangegeven in Figuur 2. Stel verder dat we $k = 4$ en $w = 6$ hebben, en dat de stenen in de volgende volgorde uit de pot komen: (0,5), (2,2), (1,1), (2,4), (2,1), (0,4), (2,0), (0,2), (2,3), (1,3), (2,5), (0,3), (3,1), (3,0), (1,0), (0,0), (0,1), (1,5), (1,2), (1,4). Stel tenslotte dat de eerste $a = 5$ stenen direct op het bord gelegd worden, en dat $k_0 = 2$ en $k_1 = 3$. Dan ziet het bord er aan het begin van het spel uit als in Figuur 3, en heeft Brian de stenen (0,4), (2,0) in zijn hand, terwijl Robin de stenen (0,2), (2,3), (1,3) in zijn hand heeft.

	0	1	2	3	4
0	-	-	-	-	-
1	-	-	-	(2,1)	(2,2)
2	-	-	-	-	(1,1)
3	-	-	(0,5)	(2,4)	-

?figurename? 3: Mogelijke beginstand op het 4×5 bord uit Figuur 2 met $a = 5$.

Brian en Robin hebben bij het spel volledige informatie. Ze weten in het bijzonder ook welke stenen de andere speler in zijn hand heeft, en in welke volgorde de stenen uit de pot komen.

Het spel eindigt wanneer de speler die aan de beurt is, geen steen meer in zijn hand heeft. In theorie is het mogelijk dat de andere speler dan nog wel stenen in zijn hand heeft, namelijk wanneer zijn keuzeaantal hoger is dan dat van de speler die aan de beurt is, zodat hij ook met een groter aantal stenen in de hand is begonnen. Als we een eindstand hebben bereikt, wordt

de score van het spel bepaald, door de grootste cluster van Brian te vergelijken met de grootste cluster van Robin.

Invoer

Een spel voor Brian en Robin kan worden ingelezen uit een tekstbestand. Zo'n tekstbestand heeft het volgende formaat:

- Een regel met twee gehele getallen: h en b, voor de hoogte en de breedte van het bord.
- h regels, overeenkomend met achtereenvolgens rij 0, rij 1, ..., rij $h-1$ van het bord. Elk van deze regels bevat b gehele getallen, voor de vakjes in de rij van links naar rechts. Deze getallen zijn de posities van de betreffende vakjes in de vulvolgorde van het bord. Als bijvoorbeeld het vierde getal in de tweede rij een 4 is, betekent dit dat vakje (1,3) op de vierde plek in de vulvolgorde staat. Merk op dat het hier niet om vakje (2,4) gaat, maar om vakje (1,3), want rijen en kolommen beginnen te nummeren bij 0.
- Een regel met één geheel getal: het aantal vormen v.
- Een regel met $h \cdot b$ gehele getallen, die de stenen voorstellen in de volgorde waarin ze uit de pot komen. Als bijvoorbeeld $v = 6$ en deze regel begint met de getallen 5, 14, 7, 16, dan zijn de eerste vier stenen die uit de pot komen achtereenvolgens (0,5), (2,2), (1,1), (2,4), want $5 = 0 \cdot 6 + 5$, $14 = 2 \cdot 6 + 2$, $7 = 1 \cdot 6 + 1$, $16 = 2 \cdot 6 + 4$.
- Een regel met één geheel getal: het aantal stenen a dat direct op het bord wordt gelegd.
- Een regel met twee gehele getallen k_0 en k_1, voor de keuzeaantallen van Brian en Robin.

Getallen op een zelfde regel van de invoer worden gescheiden door een spatie. In Figuur 4 is de inhoud van het invoerbestand te zien dat overeenkomt met het voorbeeld in en voor Figuur 3.

```
4 5
11 8 13 14 12
6 5 9 4 1
10 15 19 16 2
7 17 0 3 18
6
5 14 7 16 13 4 12 2 15 9 17 3 19 18 6 0 1 11 8 10
5
2 3
```

?figurename? 4: Inhoud van invoerbestand, overeenkomend met het voorbeeld in en voor Figuur 3.

Voor u beschikbaar

Op de website van het vak

`https://liacs.leidenuniv.nl/~vlietrvan1/algoritmiek/`

is een skeletprogramma beschikbaar, waarin een klasse `Aqualin` wordt gedefinieerd, die door het hoofdprogramma gebruikt wordt om het spel te spelen en experimenten te doen. Het programma wordt onder Linux gecompileerd met het commando `make`. Vervolgens kun je het met het commando `./Aqualin` runnen. Je krijgt dan een menu aangeboden, met de keuze om een nieuw spel te starten, een spel in te lezen uit een tekstbestand, of experimenten te doen. Bedoeling is om de `TODO`'s in de gegeven bestanden, in het bijzonder in `aqualin.cc`, `aqualin.h` en `main.cc` uit te voeren.

In de bestanden `standaard.h` en `standaard.cc` zijn enkele standaardfuncties uitgewerkt: `integerInBereik`, `randomGetal` en `genereerRandomPermutatie`. Je kunt die gebruiken om te testen of een integer tussen bepaalde grenzen ligt (al dan niet met een foutmelding), om een random getal tussen bepaalde grenzen te genereren, en om een random permutatie te genereren.

Bij het skeletprogramma zijn ook twee voorbeelden van een geldig invoerbestand gegeven: `spel1.txt` en `spel2.txt`. Het tweede bestand komt overeen met Figuur 3 en Figuur 4.

De meeste functies die je moet implementeren worden toegelicht in het skeletprogramma, speciaal in `aqualin.h`. Goed om te weten: als er in het commentaar boven een functie `Pre:` staat, dan volgt de **preconditie** voor die functie: een aantal aannames waar je vanuit mag gaan als je in die functie binnenkomt. Je hoeft dat dus in de functie niet meer te controleren. De gebruiker van de functie moet daar van tevoren voor zorgen. Net zo staat `Post:` voor de **postconditie** van de functie: zaken die na afloop van de functie moeten gelden. Daar moet je in de functie dus voor zorgen.
Van een aantal functies volgt nog een nadere toelichting:

- `bool Aqualin::leesInSpel (...)`

  Deze functie leest een spel in vanuit een tekstbestand. Dit bestand heeft het formaat zoals hierboven beschreven onder het kopje Invoer. Je moet nog wel controleren of het tekstbestand daadwerkelijk te openen is voor lezen.

  Uiteraard betekent *in*lezen dat het uit het bestand gelezen spel op een of andere wijze wordt opgeslagen in het betreffende object van de klasse `Aqualin`.

  Als je in C++ een `ifstream fin` gebruikt voor je invoer-tekstbestand, dan kun je heel eenvoudig een geheel getal inlezen, bijvoorbeeld met

  ```
  fin >> getal;
  ```

  waarbij `getal` bijvoorbeeld gedeclareerd is als:

  ```
  int getal;
  ```

  Op deze wijze wordt automatisch over spaties en newlines in het tekstbestand heengesprongen. Je hoeft het getal dus niet karakter-voor-karakter op te bouwen.

- ... (volgt nog)

Enkele tips bij het programmeren

- Kijk in het skeletprogramma waar allemaal `TODO` staat, voor met name de functies die geïmplementeerd moeten worden.

- In het skeletprogramma wordt gebruik gemaakt van `pair`, om een tweetal getallen in op te slaan. Als je niet weet hoe je die gebruikt, zoek het op in de C++ reference.

  Handig om te weten: je kunt een nieuw pair aanmaken met de functie `make_pair`. En als de variabele `paar` een pair voorstelt, dan kun je met respectievelijk `paar.first` en `paar.second` de afzonderlijke onderdelen van het pair benaderen.

- Het is verstandig om eerst na te denken over hoe je de inhoud van het spel (bord, vulvolgorde, pot, handen met stenen) op kunt slaan, en vervolgens enkele eenvoudige functies te implementeren, bijvoorbeeld de getters, `leesInSpel`, `drukAf`, de constructor met parameters, en `eindstand`.

- Als je bij de vereiste controles in de memberfuncties een fout ontdekt, geef dan ook een passende foutmelding aan de gebruiker.

- Om je programma te testen (voordat je het inlevert) komen er mogelijk nog nadere instructies, op de website van het vak.

Let op: het is niet toegestaan om de headers van de gegeven public memberfuncties in `aqualin.h` te veranderen. Dan zou onze automatische test (met een ander main programma) bij de beoordeling namelijk niet goed kunnen werken. Je mag wel functies toevoegen. Verder kan het inleveren van lelijke, niet-elegante, of erg inefficiënte code 0.5 punt aftrek opleveren.

Algemene opmerkingen

- Maak / behoud een verstandige klassenstructuur.

- Je klassen mogen alleen `public` membervariabelen en methoden hebben die buiten de klasse bekend moeten zijn. Andere membervariabelen en methoden moeten `private` zijn.

- Als er in je klassen dynamisch geheugen wordt gealloceerd, denk dan ook aan een destructor.

- Functies mogen niet te lang zijn (maximaal 35 regels).

- Gebruik constantes waar dat zinvol is. Kijk bijvoorbeeld in bestand `constantes.h` in het skeletprogramma.

- Het werkende programma mag er op het scherm eenvoudig uitzien, maar moet wel duidelijk zijn. De enige te gebruiken headerfiles zijn in principe `iostream`, `sstream`, `iomanip`, `fstream`, `cstring`, `string`, `vector`, `utility`, `set`, `unordered_set`, `cstdlib`, `ctime` en `climits`. Wil je een andere headerfile gebruiken, vraag de docent. De headerfile `algorithm` mag in ieder geval niet gebruikt worden.

- Boven elke functie moet een commentaarblokje komen met daarin een korte beschrijving van wat de functie doet. Noem daarin tevens de gebruikte parameters: geef hun betekenis en geef aan hoe ze eventueel veranderd worden door de functie. Geef bij memberfuncties ook aan wat deze met de membervariabelen van het object doen. Let verder op de layout (consequent inspringen) en op het overige commentaar bij de programmacode (zinvol en kort).

- Het programma moet onder Linux bij LIACS getest zijn en werken. Dat kan vanuit huis bijvoorbeeld op de huisuil, zie de instructie op de website.

Verslag

Het verslag moet getypt zijn in LaTeX, en moet bevatten:

- . . . (volgt nog)

Je hoeft in je verslag geen 'appendix' met je complete programma (alle .cc/.h bestanden) op te nemen. We printen het toch niet meer uit.

Aanvullingen / tips / vragen

Eventuele verdere aanvullingen of tips bij de programmeeropdracht komen op de website van het vak

`https://liacs.leidenuniv.nl/~vlietrvan1/algoritmiek/`

Daar is ook een subpagina met onder andere een template voor het LaTeX-verslag. De behaalde cijfers komen te zijner tijd in Brightspace te staan.

Heb je vragen over de opdracht, dan kun je die uiteraard stellen tijdens de practicumbijeenkomsten van het vak. Je kunt ook emailen naar algoritmiek@liacs.leidenuniv.nl

In te leveren

Via Brightspace:

- je programma (alle .cc/.h bestanden en Makefile voor Linux bij LIACS)
- en een PDF van je verslag

samen in één .zip, .tgz of .tar.gz bestand. Vermeld overal duidelijk de namen van de makers.

Uiterste (!) inleverdatum: dinsdag 1 april 2025, 23.59 uur. Je kunt maximaal twee weken te laat inleveren, maar dan gaat er wel per (deel van een) week een punt van het cijfer af. Zie de website voor de exacte regeling.

Normering: . . . (volgt nog)

Het is niet toegestaan om je opdracht te laten maken door een large language model als ChatGPT of DeepSeek. Bij opvallende inzendingen kunnen de makers worden uitgenodigd om hun oplossing toe te lichten.