Algoritmen met een balans

Tom Verhoeff*

Juni 2005

Figuur 1: Roos en Iris op een wipbalans

Samenvatting

Je leert al doende iets over de manier waarop met computers (wiskundige) vragen beantwoord kunnen worden. Denk aan het sorteren van een stel getallen. De beschrijving van zo'n manier om antwoorden te maken wordt een *algoritme* genoemd. Een algoritme moet natuurlijk altijd het juiste antwoord geven. En het moet liefst ook snel zijn.

Je werkt eerst aan problemen waarbij in een groep kinderen één of meer kinderen gevonden moeten worden met behulp van een balans, zoals bijvoorbeeld de één-na-lichtste. Vervolgens ga je enkele manieren onderzoeken om de kinderen te sorteren naar gewicht.

*Faculteit Wiskunde en Informatica van de Technische Universiteit Eindhoven, <T.Verhoeff@TUE.NL>

Voorwoord

Ik heb dit document geschreven ter ondersteuning van mijn workshop op Zomerkamp A (bovenbouw basisschool plus brugklas middelbare school) van de Stichting Vierkant voor Wiskunde[1] in augustus 2005.

Mijn doel is de kinderen iets te leren over *algoritmen*:

- wat is een algoritme,
- hoe wordt een algoritme uitgevoerd,
- wat betekent het dat een algoritme *correct* is,
- wat betekent het dat een algoritme *efficiënt* is,
- hoe analyseer je een algoritme,
- hoe redeneer je over een algoritme.

Dat lijkt misschien hoog gegrepen, maar ik hoop toch dat mijn aanpak hier iets van weet over te brengen. Het gaat er om dat ze deze dingen op een toegankelijke manier onder ogen krijgen en er zelf met plezier mee bezig zijn.

Ik besteed bewust geen aandacht aan *notaties* voor algoritmen. Je kan daarbij te makkelijk verzanden in allerlei details, die de aandacht afleiden.

De opgaven zijn bedoeld om onder begeleiding aan te werken.

[1] `http://www.vierkantvoorwiskunde.nl/`

Inhoudsopgave

1 Inleiding

Iedereen weet wel dat **computers** een belangrijke rol spelen in ons leven. Ze zitten tegenwoordig bijna overal in: PCs, TVs, horloges, mobieltjes, allerlei spelletjes, MP3-spelers, camera's, wasmachines, magnetrons, auto's, verkeerslichten, vliegtuigen, robots in fabrieken, medische apparatuur in ziekenhuizen, etc.

Een computer is een **automaat**, die **invoer** kan ontvangen en dan bijbehorende **uitvoer** produceert.

Denk aan een horloge, waarbij het computertje elke duizendste seconde een kloktik ontvangt van een trillend kristal en dan aan het schermpje vertelt welke cijfertjes getoond moeten worden. Via knopjes kun je ook invoer geven, bijvoorbeeld om de tijd en datum in te stellen, of de stopwatch te bedienen.

De **processor** is het computeronderdeel dat de berekeningsstappen doet, zoals bepalen welke dag van de week het is. De computer heeft **geheugen** om informatie te onthouden, zoals de tijd dat de wekker dient af te gaan.

Je kunt de invoer zien als de beschrijving van een **probleem** en de uitvoer als de bijbehorende **oplossing**. Of nog eenvoudiger gezegd: de invoer stelt een **vraag** en de uitvoer geeft het **antwoord**.

De computer zelf is niet slim. De computer voert domweg een (veelal lange) reeks eenvoudige **opdrachten** uit. We noemen die reeks opdrachten wel het **programma**. Het programma staat in het geheugen. Het bestaat uit hele kleine stapjes die razendsnel kunnen worden uitgevoerd. Het bedenken van zo'n programma heet **programmeren**.

We zullen ons hier bezig houden met de **wiskunde van het programmeren**. Je kunt programma's bestuderen net zo als getallen of meetkundige figuren. Zulke wiskundige programma's heten **algoritmen**.

Een algoritme beschrijft hoe je bij een (wiskundige) vraag een antwoord kan bepalen. Op de lagere school leer je bijvoorbeeld een algoritme om twee getallen, elk gegeven als rijtje van cijfers, op te tellen:

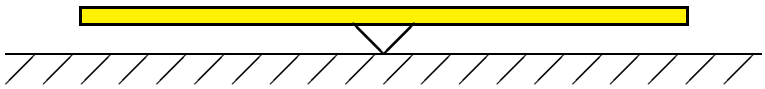
$$\begin{array}{r} 1995 \\ 10 \\ \hline 2005 \end{array} \; +$$

Dat algoritme bestaat uit een aantal eenvoudige stapjes, waaronder het optellen van twee onderelkaarstaande cijfers en het werken met 'één' onthouden'. Je moet ze in de juiste volgorde uitvoeren om het goede antwoord te bepalen. Kun je de stappen in het algoritme preciezer beschrijven? En welke soorten stapjes moet je doen voor het vermenigvuldigen van twee getallen?

In deze workshop ga je algoritmen bestuderen waarbij het gewicht van kinderen een rol speelt.

2 Balans

Het gewicht van twee kinderen kun je vergelijken met behulp van een **balans**. Op de workshop gebruiken we daarvoor een soort wipje. Om ongelukken te voorkomen en om een goede weging te doen mag de wipbalans alleen onder begeleiding gebruikt worden.



Figuur 2: Wipbalans

Met de balans kun je vaststellen wie van twee kinderen de lichtste is en wie de zwaarste. Door de uitslag van een aantal wegingen handig samen te nemen, kun je zo een aantal vragen over een groep kinderen beantwoorden. Hoe je dat precies moet doen hangt af van de vraag en wordt vastgelegd in het algoritme.

We zullen de volgende vragen beschouwen:

1. bepaal het lichtste kind
2. bepaal het lichtste èn het zwaarste kind
3. bepaal het lichtste en het één-na-lichtste kind
4. sorteer de kinderen op gewicht
5. zoek de plaats waar een nieuw kind op volgorde van gewicht moet komen te staan, gegeven een gesorteerde rij van kinderen
6. bepaal het kind met het middelste gewicht (in een groep met een oneven aantal kinderen)

We bestuderen algoritmen om deze vragen te beantwoorden. Zo'n algoritme moet natuurlijk **correct** zijn. Dat wil zeggen, het moet bij goede uitvoering altijd het juiste antwoord geven. Verder zullen we letten op de **snelheid**. We meten die snelheid door te tellen hoeveel wegingen nodig zijn.

Elk algoritme wordt beschreven als een verhaaltje waarin staat welke stappen gedaan moeten worden. Zo'n algoritme zullen we met een klein groepje van kinderen naspelen. Eén van de andere kinderen houdt bij welke stap van het algoritme aan de beurt is. Een ander kind houdt bij hoeveel keer is gewogen. Bij het naspelen is het nodig om de kinderen in groepjes of rijtjes op te delen en deze een naam te geven. We gebruiken als namen de grote letters A, B en C.

3 Lichtste

We beginnen met een makkelijk probleem, waar je zelf waarschijnlijk wel een algoritme voor kan verzinnen:

> Gegeven een groep van kinderen, bepaal met behulp van de balans wie het lichtste kind is.
> Je mag aannemen dat de groep niet leeg is en dat elk paar kinderen verschilt in gewicht.

Figuur 3: (**Probleem L**) Bepaal de lichtste

Opgave 1 Kun je beschrijven hoe je dit probleem zou oplossen? Dat kan zelfs op allerlei verschillende manieren. Probeer het eventueel eerst eens op papier met een groep van vier kinderen. Welke kinderen worden vegeleken en hoe gebruik je de uitslag van een weging bij de volgende wegingen?

Omdat het in deze workshop niet gaat om het verzinnen van de algoritmen, geven we er hier een:

> 1. Zet de kinderen op een rij, die we A noemen.
>   Het maakt niet uit in welke volgorde.
> 2. Groep B begint leeg.
> 3. Herhaal zolang de rij A meer dan één kind bevat:
>   (a) Vergelijk de eerste twee kinderen van de rij A.
>   (b) Het lichtere kind van de twee sluit *achteraan* de rij A.
>   (c) Het zwaardere kind gaat naar groep B.
> 4. De rij A bevat nu precies één kind; dit is de lichtste van allemaal; alle andere kinderen zitten in groep B.

Figuur 4: (**Algoritme L1**) Bepaal de lichtste

Opgave 2 Begrijp je wat je moet doen om dit algoritme uit te voeren? Speel Algoritme L1 maar eens na met een groep van vier kinderen. Eén van de andere kinderen houdt bij welke stap van het algoritme aan de beurt is. Let er op dat iemand het aantal wegingen telt tijdens de uitvoering. Hoeveel wegingen zijn er gedaan?

Probeer het ook eens met vijf andere kinderen. Hoeveel wegingen zijn er nu gedaan?

Laten we even stil staan bij een paar belangrijke zaken:

- Waarom is algoritme L1 *correct*? Dat wil zeggen, waarom eindigt het en vindt het inderdaad altijd het lichtste kind?
- Hoe *snel* is het algoritme? Dat wil zeggen, hoeveel wegingen zijn zo nodig om het lichtste kind te vinden?

Opgave 3 Kun je uitleggen waarom het algoritme altijd eindigt? De enige stap om je zorgen over te maken is de herhaling in stap 3. Die zou eindeloos door kunnen gaan. Echter, de rij A wordt bij elke herhaling van stap 3 *ééntje korter*. Mijn oma zei vroeger al: waar af gaat en niet bij komt, dat wordt minder. En raakt een keer op.

Verder is het zo dat in groep B alleen kinderen komen die zeker *niet* de lichtste zijn, omdat ze in een weging zwaarder waren dan een ander kind. Groep B bevat de **afvallers**. Alle kinderen op één na komen in groep B. De enige overblijver is dus de lichtste.

Opgave 4 Weet je nu ook hoeveel wegingen in het algemeen nodig zijn? Bij elke weging valt er precies één kind af. Op het einde zijn alle kinderen op één na afgevallen. Dus het aantal wegingen is één minder dan het aantal kinderen in de groep. Bij vier kinderen kost het drie wegingen en bij vijf kost het vier wegingen. Maak maar een tabel van het aantal wegingen voor 1 t/m 8, 32 en 1000 t/m 1002 kinderen.

In het algemeen geven we het aantal kinderen in de groep aan met de letter N. We hebben dus gevonden dat Algoritme L1 precies $N-1$ wegingen doet om de lichtste te vinden in een groep van N kinderen.

$N =$ aantal kinderen

Opgave 5 Stel we voegen één nieuw kind aan de groep toe. Hoeveel *meer* wegingen heeft Algoritme L1 dan nodig om de lichtste te vinden?

3.1 Afvaltoernooi

Je kunt het algoritme om de lichtste te vinden ook zien als een **afvaltoernooi**. Elke weging is een **wedstrijd** en de lichtste bij de weging is de **winnaar**. De lichtste van allemaal is de winnaar van het hele toernooi: de **kampioen**. Het vinden van de lichtste kun je dus ook zien als het organiseren van een toernooi om (alleen) de kampioen te bepalen (verliezers vallen meteen af).

Opgave 6 Figuur 5 geeft een wat ander algoritme. Speel Algoritme L2 ook eens na met vier kinderen. Hoeveel wegingen waren nodig? Waarom zijn bij Algoritme L1 en Algoritme L2 altijd evenveel wegingen nodig?

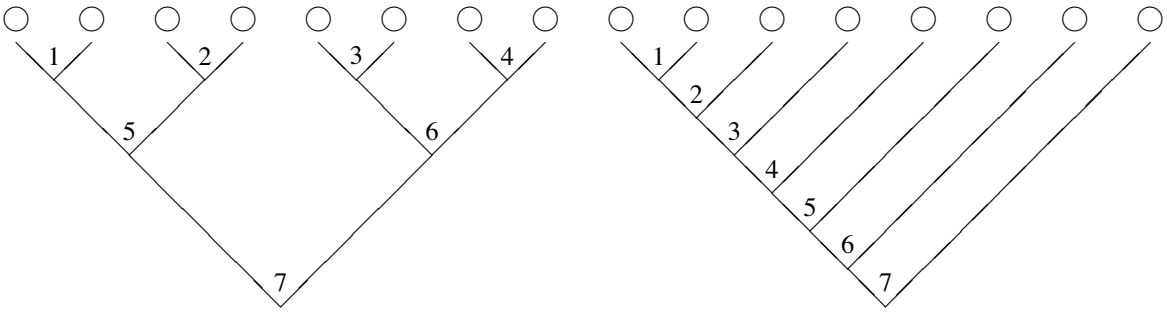
Algoritme L2 ken ik van de judotrainingen, waar we oefentoernooitjes deden met een stuk of 10 judoka's. De eerste uit de rij gaat op de mat staan. Degene die op de mat staat draait een wedstrijdje tegen de eerstvolgende uit de rij. De

1. Zet de kinderen op een rij, die we A noemen.
   Het maakt niet uit in welke volgorde.
2. Groep B begint leeg.
3. Herhaal zolang de rij A meer dan één kind bevat:
   (a) Vergelijk de eerste twee kinderen van de rij A.
   (b) Het lichtere kind van de twee sluit *vóóraan* de rij A.
   (c) Het zwaardere kind gaat naar groep B.
4. Rij A bevat nu precies één kind: dit is de lichtste van allemaal; alle andere kinderen zitten in groep B.

Figuur 5: (**Algoritme L2**) Bepaal de lichtste

winnaar van een wedstrijdje blijft telkens op de mat staan. Op het einde staat de kampioen op de mat. Je kent het zelf misschien wel bij een eigen sport of spel.

Opgave 7 Waarom is Algoritme L1 voor een echt toernooi met meer deelnemers geschikter dan Algoritme L2? Denk er aan dat op een echt toernooi meer judomatten beschikbaar zijn en er dus meer wedstrijden op hetzelfde tijdstip gedaan kunnen worden. Beschouw eens een toernooi met acht kinderen en bekijk figuur 6. Elk rondje stelt een kind voor en elke V een (genummerde) wedstrijd met één winnaar, die dan weer een volgende westrijd doet, totdat onderaan de kampioen staat.



Figuur 6: Toernooi volgens Algoritme L1 (links) en Algoritme L2 (rechts)

Opgave 8 Teken de wedstrijdschema's bij Algoritmen L1 en L2 voor *vijf* kinderen.

Opgave 9 Hoe kun je overigens het *zwaarste* kind in de groep vinden (Probleem Z)? Beschrijf hiervoor een algoritme. Hoeveel wegingen kost dat dan?

4 Lichtste en zwaarste

We maken het nu iets moeilijker:

> Gegeven een groep van kinderen, bepaal met behulp van de balans wie het lichtste kind is èn wie het zwaarste kind is. Je mag aannemen dat de groep ten minste twee kinderen bevat en dat elk paar kinderen verschilt in gewicht.

Figuur 7: (**Probleem LZ**) Bepaal de lichtste èn de zwaarste

Hoe zou je dit zelf oplossen? Hoeveel wegingen worden daarbij gedaan? **Opgave 10**
Het ligt voor de hand om eerst de lichtste te vinden en vervolgens de zwaarste:

> 1. Bepaal de lichtste van de groep met Algoritme L1 uit de vorige paragraaf.
> 2. Bepaal de zwaarste van de groep (zie opgave 9).

Figuur 8: (**Algoritme LZ1**) Bepaal de lichtste èn de zwaarste

Je hoeft dit niet na te spelen. Kun je beredeneren hoeveel wegingen bij Algoritme LZ1 worden gedaan bij een groep van vier kinderen? En hoeveel wegingen in het algemeen? Vul de tabel van opgave 4 aan met een extra kolom. **Opgave 11**

Het kan wel wat slimmer, want bij stap 2 van Algoritme LZ1 is het niet nodig om in de hele groep te zoeken naar de zwaarste:

> 1. Bepaal de lichtste van de groep met Algoritme L1 uit de vorige paragraaf.
> 2. Bepaal de zwaarste van de groep *zonder* de lichtste gevonden in stap 1 (zie opgave 9).

Figuur 9: (**Algoritme LZ2**) Bepaal de lichtste èn de zwaarste

Hoeveel wegingen worden bij Algoritme LZ2 gedaan bij een groep van vier kinderen? En hoeveel wegingen in het algemeen? Vul de tabel aan met een kolom. **Opgave 12**

Stel we voegen één nieuw kind aan de groep toe. Hoeveel *meer* wegingen heeft Algoritmen LZ1 dan nodig om de lichtste èn de zwaarste te vinden? Hoe zit dat met Algoritme LZ2? **Opgave 13**

Het kan nog slimmer. In Algoritmen LZ1 en LZ2 wordt de kennis opgedaan bij de wegingen in stap 1 niet hergebruikt in stap 2. Als we in stap 2 van Algoritme LZ1 de tegenhanger van Algoritme L1 gebruiken (laten we dat Algoritme Z1 noemen) en daarbij de kinderen *in dezelfde volgorde* zetten als bij stap 1, dan worden bij stap 1 en stap 2 in de *eerste ronde* (denk aan het afvaltoernooi en het wedstrijdschema bij opgave 7) precies dezelfde wegingen gedaan. Die wegingen kun je dus uitsparen in stap 2. Maar je moet dan wel uitleggen hoe je dat precies organiseert. Dat kan zo (figuur 10):

1. Zet de kinderen op een rij, die we A noemen.
   Het maakt niet uit in welke volgorde.
2. Groepen B en C beginnen leeg.
3. Herhaal zolang de rij A meer dan één kind bevat:
   (a) Vergelijk de eerste twee kinderen van de rij A.
   (b) Het lichtere kind gaat van rij A naar groep B.
   (c) Het zwaardere kind gaat van rij A naar groep C.
4. Bepaal de lichtste van rij A en groep B samen.
5. Bepaal de zwaarste van rij A en groep C samen.

Figuur 10: (**Algoritme LZ3**) Bepaal de lichtste èn de zwaarste

Opgave 14 Speel Algoritme LZ3 maar eens na met een groep van vier kinderen. Eén van de andere kinderen houdt bij welke stap van het algoritme aan de beurt is. Let er op dat iemand het aantal wegingen telt tijdens de uitvoering. Hoeveel wegingen zijn er gedaan? Probeer het ook eens met vijf andere kinderen. Hoeveel wegingen zijn er nu gedaan?

Opgave 15 Rij A bevat na stap 3 van Algoritme LZ3 ten hoogste één kind. Bij welke startgroepen is rij A na stap 3 *leeg* en bij welke bevat rij A dan *precies één* kind.

Opgave 16 Als in stap 5 van Algoritme LZ3 rij A *niet* meegenomen wordt en alleen maar de zwaarste van groep C wordt bepaald, dan is het algoritme *fout*. Hoe zou je dat kunnen merken? Kun je een voorbeeld geven van een groep van drie kinderen waarbij het algoritme dan het foute antwoord geeft?

Opgave 17 Hoeveel wegingen worden bij Algoritme LZ3 gedaan bij een groep met een even aantal kinderen? Hoeveel bij een oneven aantal? Vul de tabel weer aan.

Opgave 18 Stel we voegen *twee* nieuwe kinderen aan de groep toe. Hoeveel *meer* wegingen heeft Algoritme LZ3 dan nodig om de lichtste èn de zwaarste te vinden? Hoeveel wegingen extra is dat *gemiddeld* voor één kind extra?

5 Lichtste en één-na-lichtste

We maken het nu nog iets moeilijker:

> Gegeven een groep van kinderen, bepaal met behulp van de balans wie het lichtste kind is èn wie het één-na-lichtste (het tweede) kind is.
> Je mag aannemen dat de groep ten minste twee kinderen bevat en dat elk paar kinderen verschilt in gewicht.

Figuur 11: (**Probleem LT**) Bepaal de lichtste en de één-na-lichtste

Omdat 'één-na-lichtste' zo'n lang woord is, zeggen we liever 'tweede'.

Dit probleem lijkt wel wat op het vorige. Hoe zou je het zelf oplossen? Hoeveel wegingen worden daarbij gedaan? **Opgave 19**

De volgende oplossing lijkt sterk op Algoritme LZ2:

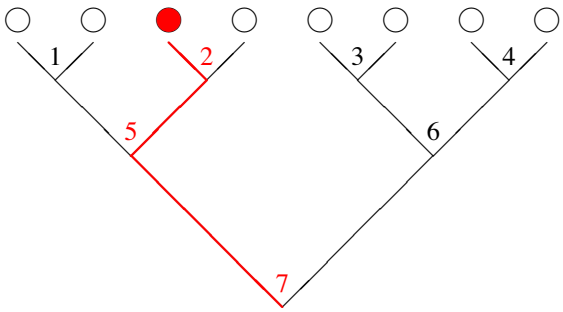
> 1. Bepaal de lichtste van de groep (bijvoorbeeld met Algoritme L1 uit §3).
> 2. Bepaal de lichtste van de groep *zonder* de lichtste gevonden in stap 1. Dit is de tweede (de één-na-lichtste) van de hele groep.

Figuur 12: (**Algoritme LT1**) Bepaal de lichtste èn de tweede

Het aantal wegingen is hetzelfde als bij Algoritme LZ2. Maar ook in dit geval kan het nog wel beter. Toch is dit een ander probleem dan het vorige, omdat in stap 2 weer de lichtste bepaald wordt en niet de zwaarste.

We willen in stap 2 van Algoritme LT1 zoveel mogelijk wegingen uit stap 1 hergebruiken (en niet opnieuw doen). Dat kan door in stap 2 hetzelfde wedstrijdschema aan te houden als in stap 1, maar *zonder* de lichtste gevonden in stap 1. We vervangen daarom *in gedachte* die lichtste door een heel zwaar iemand, die alle wedstrijden verliest. De kampioen van het tweede toernooi is dan de tweede van de hele groep.

Laten we een voorbeeld bekijken. Begin met een groep van acht kinderen. In figuur 13 staat het wedstrijdschema voor stap 1 van Algoritme LT1 gedaan met Algoritme L1 om de lichtste te bepalen. Stel dat het derde kind van links (rood gekleurd) de lichtste is. Deze wint al haar wedstrijden (2, 5 en 7). In stap 2 vervangen we dit kind door 'iemand' die heel zwaar is. De rode wedstrijden (2, 5 en 7) moeten daarom opnieuw gedaan worden, de anderen zijn al gedaan in



Figuur 13: Wedstrijdschema bij Algoritme LT1 als het derde kind van links de lichtste is

stap 1. Wedstrijd 2 is op voorhand al verloren door het zware rode kind. Twee extra wedstrijden volstaan dus.

Je kan dit ook op een andere manier begrijpen. De rode wedstrijden werden in stap 1 gewonnen door het lichtste kind. De verliezers van die wedstrijden hebben alleen verloren van de kampioen. Ze zijn de enige die in aanmerking komen voor de tweede plaats. Immers, de tweede wint alle wedstrijden, *behalve* tegen de kampioen. Om te weten wie op de tweede plaats komt, moet dus de winnaar bepaald worden van de (drie) verliezers van de kampioen. Dat kan in twee wegingen. In totaal zijn dan $7 + 2 = 9$ wegingen gedaan.

In het algemeen komt dat neer op het volgende algoritme:

1. Bepaal de lichtste van de groep met Algoritme L1 (uit §3). *en laat elk kind onthouden van wie verloren werd*. Vul daartoe het wedstrijdschema in.

2. Bepaal de lichtste van de groep bestaande uit de *verliezers van de lichtste* gevonden in stap 1. Dit is de tweede (de één-na-lichtste) van de hele groep.

Figuur 14: (**Algoritme LT2**) Bepaal de lichtste èn de tweede

Opgave 20 Speel Algoritme LT2 maar eens na met een groep van vier kinderen. Probeer het ook eens met vijf andere kinderen. Hoeveel wegingen zijn er gedaan?

Opgave 21 Hoeveel wegingen worden bij Algoritme LT2 gedaan bij een groep met een aantal kinderen dat een *macht van twee* is? Een macht van twee is te schrijven als $2 \times 2 \times \cdots \times 2$, zoals $2, 4, 8$ en ook 32 en 1024. Het wedstrijdschema voor stap 1 is dan heel regelmatig. Hoeveel als het geen macht van twee is? Let op dat het aantal dan ook afhangt van de volgorde in de rij. Het aantal kan één schelen. Vul de tabel weer aan (met de grotere waarden, als er twee zijn).

6 Sorteren

Als je ook de derde, vierde, enzovoorts wilt vinden, dan kun je de kinderen misschien net zo goed op gewicht sorteren:

> Gegeven een groep van kinderen, zet ze met behulp van de balans op volgorde van gewicht.
> Je mag aannemen dat de groep niet leeg is en dat elk paar kinderen verschilt in gewicht.

Figuur 15: (**Probleem S**) Sorteer op gewicht

Als je kan sorteren, dan zijn daarmee de problemen uit de vorige paragrafen ook meteen op te lossen. Sorteren is dus algemener en daarom vermoedelijk ingewikkelder dan de vorige problemen. Maar misschien heb je toch een idee hoe je het kan aanpakken. Hoe zou je de kinderen sorteren? Kun je iets zeggen over het aantal wegingen dat daarbij wordt gedaan? **Opgave 22**

Een van de eenvoudigste algoritmen om te sorteren bestaat uit herhaaldelijk de lichtste vinden:

> 1. Plaats alle kinderen in één groep, die we A noemen.
> 2. Rij B begint leeg. Dit wordt het antwoord.
> 3. Herhaal zolang de groep A niet leeg is:
>   (a) Bepaal de lichtste van groep A, bijvoorbeeld met Algoritme L1 (uit §3).
>   (b) Verwijder dit lichtste kind uit groep A en plaats het achteraan rij B.
> 4. Rij B is nu de gesorteerde rij kinderen.

Figuur 16: (**Algoritme S1**) Sorteer op gewicht

Speel Algoritme S1 maar eens na met een groep van vier kinderen. Let er op dat bij het uitvoeren van Algoritme L1 in stap 3(a) een 'eigen' rij A en groep B een rol spelen. Hoeveel wegingen zijn er in totaal gedaan? **Opgave 23**

Hoeveel wegingen worden in het algemeen bij Algoritme S1 gedaan? Vul de tabel weer aan met een extra kolom. **Opgave 24**

Sorteren kan wel sneller dan met Algoritme S1. Dat is misschien niet zo makkelijk zelf te ontdekken. Daarom beschrijven we hier zonder omhaal een beter

Algoritme S2. Als de groep uit één kind bestaat, dan is dat meteen de gesorteerde rij. We gaan er nu van uit dat er ten minste twee kinderen in de groep zitten:

1. Splits de groep in twee aparte niet-lege groepen A en B, die qua grootte zo min mogelijk schelen.
2. Sorteer groep A tot rij A met Algoritme S2.
3. Sorteer groep B tot rij B met Algoritme S2.
4. Voeg de gesorteerde rijen A en B samen tot één gesorteerde rij. Dit is apart beschreven als Algoritme V1.

Figuur 17: (**Algoritme S2**) Sorteer op gewicht (ten minste twee kinderen)

In stap 4 worden wegingen gedaan om de twee gesorteerde rijen samen te voegen. We beschouwen dit als een apart probleem:

Gegeven twee gesorteerde rijen van kinderen, voeg ze met behulp van de balans samen tot één gesorteerde rij.
Je mag aannemen dat beide rijen niet leeg zijn en dat elk paar kinderen verschilt in gewicht.

Figuur 18: (**Probleem V**) Voeg twee gesorteerde rijen samen

Het volgende algoritme lost Probleem V op:

1. Noem de twee gegeven gesorteerde rijen A en B.
2. Rij C begint leeg. Dit wordt het antwoord.
3. Herhaal zolang rij A èn rij B *beide* niet leeg zijn:
   (a) Vergelijk het eerste kind van rij A met het eerste kind van rij B.
   (b) Het lichtere kind sluit achteraan rij C.
   (c) Het zwaardere kind blijft staan.
4. Rij A of rij B is nu leeg. Plaats de ander achter rij C.

Figuur 19: (**Algoritme V1**) Voeg twee gesorteerde rijen samen

Opgave 25 Speel Algoritme S2 na met een groep van vier kinderen. Let er op dat bij het

uitvoeren van Algoritme S2 in stappen 2 en 3 weer ‘eigen’ groepen A en B een rol spelen. Hoeveel wegingen zijn er in totaal gedaan?

Opgave 26

Hoeveel wegingen worden in het algemeen bij Algoritme V1 gedaan in het ‘slechtste’ geval? En hoeveel voor Algoritme S2? Werk eerst het geval uit dat het aantal kinderen een macht van twee is: $2 \times 2 \times \cdots \times 2$. Vul de tabel weer aan met een extra kolom.

Sorteren is een veelvoorkomende activiteit van computers. Een van de redenen om te sorteren is dat iets veel sneller is te vinden in een gesorteerde lijst dan in een ongesorteerde lijst. (Weet je hoe?)

Sorteren is niet alleen nuttig om telefoonboeken en woordenboeken te maken, waarin de namen en woorden op alfabetische volgorde staan. Er zijn ook allerlei andere toepassingen van sorteren. We noemen er hier een paar:

- Controleren of er dubbelen zijn (na sorteren staan ze naast elkaar);
- makkelijker kunnen tellen hoe vaak elk ding vóórkomt (na sorteren staan gelijken naast elkaar);
- rangorde bepalen, bijvoorbeeld om de derde of de middelste te vinden.

Opgave 27

Sorteer nu alle kinderen van de workshop op gewicht met Algoritme S2. Stappen 2 en 3 kunnen daarbij gelijktijdig worden uitgevoerd als er genoeg balansen zijn.

7 Slot

Je hebt kennisgemaakt met een aantal algoritmen waarbij een balans gebruikt wordt. Je hebt gezien dat hetzelfde probleem op verschillende manieren opgelost kan worden en dat deze oplossingen (veel) kunnen verschillen in het aantal wegingen dat ze vergen. Bekijk de tabel met het aantal wegingen per algoritme nog maar eens.

Algoritmen zijn ingewikkeldere dingen dan getallen of meetkundige figuren. Maar je hoeft niet veel te weten om er wat mee te doen en van te leren.

Het bestuderen van algoritmen is een onderdeel van het vak *Informatica*. Wiskunde speelt daarbij een belangrijke rol.

A Notatie voor algoritmen

We hebben geen formele notatie voor algoritmen gebruikt. De belangrijkste ingredienten van de beschouwde algoritmen zijn:

- groepjes van kinderen (volgorde *niet* van belang):
  - een (leeg) groepje vormen
  - het aantal kinderen erin bepalen
  - een of meer kinderen erin aanwijzen
  - kinderen toevoegen of verwijderen
- rijtjes van kinderen (volgorde *wel* van belang, inclusief onderscheid vóór- en achteraan):
  - een (leeg) rijtje vormen
  - het aantal kinderen erin bepalen
  - kinderen vóór- of achteraan toevoegen of verwijderen
- kinderen, groepjes en rijtjes benoemen;
- (het gewicht van) twee kinderen vergelijken met de balans;
- de volgende stap is pas aan de beurt als de vorige geheel af is;
- afhankelijk van de balansuitslag (of een andere conditie) het één of het ander doen;
- iets herhalen zolang een zekere conditie geldt;
- een ander (benoemd) algoritme activeren om een deelprobleem op te lossen;
- het algoritme zelf nogmaals (recursief) activeren (maar dan voor een 'kleinere' versie van hetzelfde probleem)

B Aanwijzingen voor begeleiders

B.1 Gebruik van de wipbalans

Zo gebruik je de wipbalans om het gewicht van twee kinderen te vergelijken:

1. Schuif —zonder kinderen erop— eerst de blokjes links en rechts eronder.
2. Laat de twee kinderen opstappen met hun neus naar het midden.
3. Zorg ervoor dat hun voeten op de 'voetafdrukken' staan met hun hakken tegen de achterzijde.
4. Wacht tot ze rechtop stilstaan.
5. Verwijder de blokjes gelijktijdig. Hiervoor is hulp nodig, bijvoorbeeld van twee begeleiders (of andere kinderen). Let er op dat ze hun vingers niet onder de plank houden.
6. De wip zakt door aan de kant van het zwaardere kind.

Figuur 20: Gebruik van de wipbalans

Er zijn een paar redenen om de blokjes te plaatsen:

- De blokjes houden de balans in evenwicht en laten daardoor het opstappen veiliger en vlotter verlopen.
- De blokjes zorgen er voor dat de wipbalans in het midden (de metastabiele toestand) begint en dan uitslaat naar de zwaardere kant, ook als het verschil in gewicht klein is. Omdat het draaipunt van de wipbalans er*onder* ligt, heeft de wipbalans, in het geval het gewicht van de kinderen weinig verschilt, *twee* stabiele toestanden. Als ze één voor één opstappen, hangt de uitslag af van de opstapvolgorde.
- De blokjes maken de uitslag ook betrouwbaarder, omdat de kinderen op een gestabiliseerde balans eerder geneigd zullen zijn om op de juiste plaats tegaan staan en een 'neutrale' houding aan te nemen.

Houd bij het gebruik van de wipbalans de volgende beperkingen in gedachte.

- De uitslag is niet noodzakelijk *reproduceerbaar*. D.w.z. hetzelfde paar kinderen kan bij een tweede weging een andere uitslag geven. Mogelijke redenen hiervoor zijn:
  - Ze staan verwisseld op de balans (de wipbalans zelf is niet precies symmetrisch).
  - De voeten staan anders (m.n. de hakken niet tegen de achterzijde).
  - Ze nemen een andere houding aan (hangen vóór- of achterover, of proberen te balanceren).
  - Ze hebben tussendoor gegeten of gedronken of zijn naar het toilet geweest.
  - Ze hebben iets extra's meegenomen op de balans (bijv. in de broekzak) of juist weggelegd.
- De uitslagen zijn mogelijk niet *transitief*. D.w.z. als $a < b$ en $b < c$, dan mogelijk niet $a < c$.
- De balans is niet geschikt om vast te stellen dat twee kinderen *hetzelfde* gewicht hebben.

Hier is weinig aan te doen. De ervaring moet nog uitwijzen of dit aanleiding geeft tot ernstige problemen.

Een ander aspect dat niet is meegenomen in de algoritmen is de aanwezigheid van verschillende kinderen met hetzelfde gewicht.

B.2 Algoritmen uitvoeren

Benodigdheden:

- beschrijving van het algoritme, eventueel op groot vel of geprojecteerd
- wipbalans met twee blokjes
- goedgedefinieerde groep kinderen, als invoer voor het algoritme
- eventueel stukken touw en lettervellen (A4-tjes met grote A, B, C erop) om groepen/rijen mee af te bakenen en te benoemen (N.B. via activeren van subalgoritmen kunnen verschillende groepen bestaan met dezelfde (lokale) naam)
- kinderen om te helpen:
  - twee voor de wipbalans om de blokjes te bedienen,

  - één om bij te houden welke stap in het algoritme aan de beurt is (bij activatie van subalgoritmen en bij recursie: één extra per activatie kan nuttig zijn),
  - één om het aantal wegingen te turven.

Let er op

- dat alle stappen van het algoritme precies zo worden worden uitgevoerd als beschreven;
- dat geen stappen worden overgeslagen;
- dat de voorgeschreven volgorde van de stappen wordt aangehouden;
- dat ook wegingen die al eens eerder zijn gedaan gewoon opnieuw worden uitgevoerd en meegeteld.

B.3 Achtergronden bij de algoritmen

B.3.1 Lichtste

Een algemener algoritme om de lichtste te bepalen staat in fig. 21.

De algoritmen die we besproken hebben zijn een specialisatie hiervan. Door de kinderen op een rij te zetten, wordt de vrijheid ingeperkt. De kinderen kunnen echter bij Algoritme L2 het gevoel hebben dat dit *niet eerlijk* is. De reden om het vrijere algoritme niet te presenteren, is juist omdat er zoveel vrijheid in zit. Dat geeft mogelijk conflicten bij het naspelen, omdat er dan telkens gekozen moet worden wie er op de wip mogen. (Aan de andere kant moet er bij Algoritmen L1 en L2 toch een rijvolgorde gekozen worden, maar dat is eenmalig.)

Het is eenvoudig in te zien dat er bij N kinderen ten minste $N - 1$ wegingen nodig zijn om de lichtste te vinden. Het wedstrijdschema moet een samenhangende graaf zijn op N knopen. Deze heeft derhalve minstens $N - 1$ takken. De besproken algoritmen zijn dus optimaal.

B.3.2 Lichtste èn zwaarste

Strictgenomen kan dit probleem ook gesteld worden voor een groep met maar één kind. In dat geval zijn de lichtste en de zwaarste hetzelfde kind. We hebben dat toch uitgesloten om verwarring te voorkomen, omdat er dan sprake is van *aliasing* (twee verschillende aanduidingen voor dezelfde entiteit).

1. Noem de groep kinderen A
2. Groep B begint leeg.

   (Invariant: Groepen A en B overlappen niet en bevatten samen alle kinderen. Verder bevat groep B de afvallers, die zeker niet de lichtste zijn. De lichtste van groep A is de lichtste van de hele groep.)
3. Herhaal zolang groep A meer dan één kind bevat:
   (a) Kies twee kinderen uit groep A.
   (Hier heb je een hoop vrijheid.)
   (b) Vergelijk ze met de balans.
   (c) Verplaats het zwaardere kind van A naar B.
4. Groep A bevat nu precies één kind, en wel de lichtste van allemaal; alle anderen zitten immers in B en zijn dus afgevallen.

Figuur 21: (**Algoritme L3**) Bepaal de lichtste

Algoritme LZ1 lijkt misschien overbodig om op te nemen, omdat menigeen meteen het efficiëntere Algoritme LZ2 bedenkt. De reden om het toch op te nemen is dat Algoritme LZ3 directer verwant is aan Algoritme LZ1 dan aan Algoritme LZ2.

In Algoritme LZ3 bestaat groep B uit afvallers voor de *zwaarste*, terwijl groep C juist bestaat uit afvallers voor de *lichtste*. Bij een oneven aantal kinderen is het overgebleven kind in rij A nog kandidaat voor zowel lichtste als zwaarste.

Er is ook een fraai *recursief* algoritme dat evenveel wegingen vergt als Algoritme LZ3. Het is gebaseerd op de volgende observatie.

> Als je van twee groepen elk de lichtste en zwaarste kent, dan kun je met *twee* wegingen de lichtste en zwaarste van de twee groepen samen bepalen: vergelijk de twee lichtsten en ook de twee zwaarsten.

Voor $N = 1$ en $N = 2$ is het probleem 'triviaal'. Voor $N > 2$ los je het probleem eerst op voor $N-2$ kinderen. De 2 andere kinderen zijn met *één* weging onderling te ordenen en met nog eens *twee* wegingen samen te voegen bij de overige $N-2$.

Dit is ook *iteratief* goed te beschrijven (maar in mijn ogen toch lastiger dan Algoritme LZ3): Houdt de lichtste en zwaarste bij van een groep, die leeg begint en steeds met *twee* kinderen wordt uitgebreid in *drie* wegingen. Op het einde moet mogelijk nog een resterend kind worden meegenomen (in twee wegingen).

B.3.3 Lichtste en één-na-lichtste

Hier is het echt nodig dat de groep ten minste twee kinderen bevat, omdat met minder kinderen het probleem zou vragen naar iets dat niet bestaat.

Er zijn twee complicerende factoren bij het efficiëntere algoritme LT2:

- Er moet informatie onthouden worden, die van een ingewikkeldere vorm is dan bij de voorgaande algoritmen. We hebben dit geformuleerd door elk kind te laten onthouden van wie verloren werd. In programmeertalen als Pascal, C/C++ en Java kan dat bijv. met een *array*.

- Het aantal wegingen hangt niet alleen af van de groepsgrootte. Er is een verschil tussen *best case* en *worst case*. Door in stap 1 van Algoritme LT2 de lichtste te bepalen met Algoritme L1 blijft dat verschil beperkt tot hooguit één weging.

  Als Algoritme L2 zou zijn gebruikt, dan loopt dat verschil op:

  Best case vergt dan $N - 1$ wegingen (wat beter is dan de best case bij gebruik van Algoritme L1), nl. als de lichtste achteraan rij A staat. Er is dan maar één kind dat verliest van de lichtste en daarmee is de tweede meteen bepaald.

  Worst case vergt dan $N - 1 + N - 2 = 2N - 3$ wegingen, nl. als de lichtste vóóraan rij A staat. De lichtste wordt dan met elk ander kind vergeleken, waardoor ze allemaal nog kandidaat zijn voor de tweede plaats.

  Het 'compactere' wedstrijdschema van Algoritme L1 zorgt er voor dat de worst case geminimaliseerd wordt. Dit gaat echter ten koste van een slechtere best case.

Merk op dat Algoritme LT2 'aanzienlijk' sneller is dan Algoritme LZ3, maar dat er tussendoor ook meer informatie onthouden moet worden. Als je het met minder geheugen wilt doen, dan kan dat wel en kost het evenveel als Algoritme LZ3. Dit berust op een soortgelijke observatie als hierboven:

> Als je van twee groepen elk de lichtste en tweede kent, dan kun je met *twee* wegingen de lichtste en tweede van de twee groepen samen bepalen: vergelijk eerst de twee lichtsten en vergelijk vervolgens de zwaardere met de tweede van de andere groep.

Iteratief de lichtste en tweede bijhouden van een groep, die leeg begint en telkens met één kind wordt uitgebreid vergt worst case ca. $2N$ wegingen.

Opmerkelijk is verder nog dat het bepalen van alleen maar de één-na-lichtste niet sneller gaat. Je vind dan toch ook de lichtste. Dit in tegenstelling tot het probleem van de lichtste èn de zwaarste.

B.3.4 Sorteren

Sorteren kan op zeer vele manieren en is uitgebreid bestudeerd. De kennismaking die ik hier bied gaat aan vele zaken voorbij.

Herhaaldelijk de lichtste selecteren (Algoritme S1) staat bekend als *Selection Sort*. Ik heb dit algoritme gekozen omdat het direct aansluit bij de voorgaande algoritmen. Bovendien is het aantal wegingen alleen afhankelijk van de groepsgrootte.

Ik heb verder gekozen voor *Merge Sort* (Algoritme S2), omdat het efficiënt is en eenvoudig te beschrijven en te begrijpen binnen de contekst van kinderen wegen. Bovendien is het een algoritme waarin recursie tot zijn recht komt, omdat er meer dan één recursieve activatie een rol speelt.

In stap 1 van Algoritme S2 (Merge Sort) is voor de correctheid alleen van belang dat de deelgroepen A en B *niet leeg* zijn. Dat ze zo min mogelijk verschillen qua grootte is alleen van belang om het aantal wegingen te minimaliseren. Als je een van de deelgroepen laat bestaan uit één kind, dan reduceert het algoritme tot *Insertion Sort*, waarbij een gesorteerde rij kinderen telkens met één wordt uitgebreid door deze op de juiste plaats tussen te voegen.

Als je Algoritme LT2 verder zou uitwerken om achtereenvolgens ook de derde, vierde, etc. efficiënt te bepalen, dan kom je op *Heapsort*, met een datastructuur waaruit je in circa $\log_2 N$ wegingen telkens de lichtste kan verwijderen.

B.4 Antwoorden en kanttekeningen bij de opgaven tussendoor

Opgave 1 (Eigen algoritme voor de lichtste) Sta hier niet te lang bij stil, want het gaat er niet om dat ze zelf algoritmen verzinnen en formuleren. Maar moedig iemand met een idee wel aan het kort uit te leggen.

Opgave 2 (Speel Algoritme L1 na) Probeer ze dit zo serieus mogelijk te laten naspelen. Het is de eerste keer en zet dus de toon voor de volgende keren. Ze moeten de stappen van Algoritme L1 *precies* volgen.

Opgave 3 (Waarom eindigt Algoritme L1?) Dit is de eerste keer dat ze een vraag over eindiging krijgen voorgelegd. Het antwoord wordt meteen erna gegeven. Belangrijk is dat ze zich realiseren dat eindiging niet zo vanzelfsprekend is als het misschien lijkt. Overigens is de redenering in de tekst niet volledig: Ook van belang is dat de rij bij aanvang ten minste één kind bevat.

Opgave 4 (Hoeveel wegingen doet Algoritme L1?) Dit is de eerste keer dat ze een vraag over het aantal wegingen krijgen voorgelegd. Het antwoord wordt meteen erna gegeven. Formules als $N - 1$ hebben ze op school

waarschijnlijk nog niet gehad. Maar ik denk dat ze het met wat aanmoediging wel kunnen lezen. Het helpt ook om ze een tabel te laten maken:

	aantal wegingen bij	
N	L1	...
1	0	
2	1	
3	2	
4	3	
5	4	
6	5	
7	6	
8	7	
32	31	
1000	999	
1001	1000	
1002	1001	

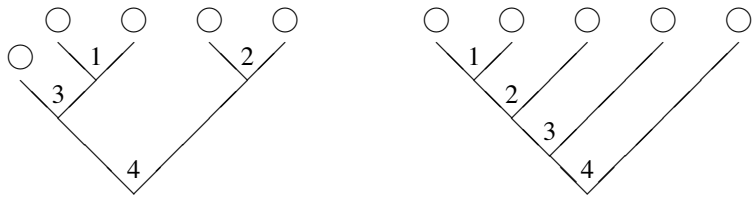
Opgave 5 (Hoeveel wegingen doet Algoritme L1 extra voor één kind extra?) De vraag heeft misschien wat toelichting nodig. Het gaat om het aantal extra wegingen t.o.v. het aantal wegingen voor de groep zonder het extra kind. Eén kind extra kost één weging extra. Dit is ook goed in de tabel bij opgave 4 te zien. Dit benadrukt het lineaire karakter van het aantal wegingen met coëfficiënt één.

Opgave 6 (Waarom doen Algoritmen L1 en L2 altijd evenveel wegingen?) Uiteraard is van belang dat beide algoritmen op dezelfde groep werken. De redenering over het aantal wegingen bij Algoritme L1 gaat ook op voor Algoritme L2: bij elke weging valt precies één kind af.

Opgave 7 (Waarom is Algoritme L1 geschikter dan L2 voor een grotere groep met meer balansen?) Je kan bij Algoritme L1 de rij opdelen in paren, die allemaal *tegelijk* vergeleken kunnen worden (als er genoeg balansen zijn). Bij acht kinderen kunnen er vier wegingen gedaan worden in de 1e ronde, die vier 'winnaars' oplevert. Deze vier leveren in de 2e ronde met twee wegingen weer twee 'winnaars'. Tenslotte komt uit de laatste weging in de 3e ronde de 'kampioen'.

Bij Algoritme L2 zijn er bij acht kinderen *zeven* rondes nodig (en maar één balans). Dit kun je eenvoudig uitleggen aan de hand van figuur 6.

Opgave 8 (Wedstrijdschema's voor toernooi met 5 kinderen.) Zie figuur 22. Let op dat bij Algoritme L1 het achterste kind in de rij pas in wedstrijd 3 aan de beurt komt.



Figuur 22: Toernooi volgens Algoritme L1 (links) en Algoritme L2 (rechts)

Opgave 9 (Vind de zwaarste) Dat gaat natuurlijk net als de lichtste, maar dan is telkens het zwaardere kind de 'winnnaar'. Zo vind je Algoritmen Z1 en Z2. Bij N kinderen worden hierbij derhalve ook $N-1$ wegingen gedaan.

Opgave 10 (Eigen algoritme voor lichtste èn zwaarste) Zie opgave 1.

Opgave 11 (Hoeveel wegingen doet Algoritme LZ1?) Twee minder dan twee keer het aantal kinderen in de groep: $(N-1)+(N-1) = 2N-2$. Bij vier kinderen vergt dit zes wegingen. Vul de tabel bij opgave 4 aan met extra kolommen (zie ook opg. 17).

Opgave 12 (Hoeveel wegingen doet Algoritme LZ2?) Drie minder dan twee keer het aantal kinderen in de groep: $(N-1)+(N-1-1) = 2N-3$. Bij vier kinderen vergt dit vijf wegingen.

Opgave 13 (Hoeveel wegingen doen Algoritmen LZ1 en LZ2 extra voor één kind extra?) Bij beide kost één kind extra *twee* wegingen extra. Dit benadrukt het lineaire karakter van het aantal wegingen met coëfficiënt twee.

Opgave 14 (Speel Algoritme LZ3 na) Ze moeten de stappen van Algoritme LZ3 *precies* volgen. Rij A bevat na afloop van stap 3 nul of één kind(eren). Bij stappen 4 en 5 moeten ze een algoritme van de vorige paragraaf gebruiken.

Opgave 15 (Rij A na stap 3 in Algoritme LZ3) Als Algoritme LZ3 met een even aantal kinderen start, dan is rij A na stap 3 leeg en bij een oneven aantal bevat rij A precies één kind.

Opgave 16 (Toon fout aan in variant van Algoritme LZ3) Als de rij A in stap 5 van Algoritme LZ3 niet zou meedoen, dan gaat het fout als het laatste kind het zwaarste is. Want die doet dan niet mee in stap 5 en derhalve levert het algoritme een verkeerd antwoord voor de zwaarste.

Opgave 17 (Hoeveel wegingen doet Algoritme LZ3?) Als N even is, zeg $N = 2M$, dan worden er M wegingen in stap 3 gedaan, en in stappen 4 en 5 elk nog $M-1$. In totaal dus $3M-2$.

Als N oneven is, zeg $N = 2M + 1$, dan worden er M wegingen in stap 3 gedaan, en in stappen 4 en 5 elk nog M. In totaal dus $3M$.

Je kunt deze gevallen nog samennemen. De besparing t.o.v. Algoritme LZ1 is $\lfloor N/2 \rfloor$. In totaal dus $\lceil \frac{3N}{2} \rceil - 2 = \lfloor \frac{3}{2}(N-1) \rfloor$ wegingen, maar dat voert te ver.

	aantal wegingen bij			
N	L1	LZ1	LZ2	LZ3
1	0	–	–	–
2	1	2	1	1
3	2	4	3	3
4	3	6	5	4
5	4	8	7	6
6	5	10	9	7
7	6	12	11	9
8	7	14	13	10
32	31	62	61	46
1000	999	1998	1997	1498
1001	1000	2000	1999	1500
1002	1001	2002	2001	1501

Algoritme LZ3 is dus 'aanzienlijk' (ca. $N/2$) sneller dan Algoritme LZ2.

Opgave 18 (Hoeveel wegingen doet Algoritme LZ3 extra voor twee kinderen extra?) Elke *twee* kinderen extra kost *drie* wegingen extra: één bij elk van de stappen 3, 4 en 5. Gemiddeld *anderhalve* weging extra per extra kind.

Opgave 19 (Eigen algoritme voor lichtste èn tweede) Zie opgave 1.

Opgave 20 (Speel Algoritme LT2 na) Ze moeten de stappen van Algoritme LT2 *precies* volgen. Bij stap 1 moeten ze Algoritme L1 gebruiken. Het invullen van het wedtrijdschema is nodig, omdat er toch enkelen kunnen vergeten van wie ze verloren hebben. Bij stap 2 maakt het niet uit welk algoritme gebruikt wordt voor het vinden van de lichtste.

Opgave 21 (Hoeveel wegingen doet Algoritme LT2?) In stap 1 worden altijd $N - 1$ wegingen gedaan. Als $N = 2^K$, dan zijn er K verliezers van de lichtste en kost het nog $K - 1$ wegingen om daaronder de tweede te vinden.

Als N geen macht van twee is, dan geldt $2^{K-1} < N < 2^K$. Afhankelijk van de positie van het lichtste kind in de rij, heeft deze $K - 1$ of K wedstrijden gedaan met evenzoveel verliezers, die kandidaat zijn voor de tweede plaats.

In het algemeen, als $2^{K-1} < N \leq 2^K$, dan kost het in het slechtste geval $N - 1 + K - 1 = N + K - 2$ wegingen ($K = \lceil \log_2 N \rceil$). Het vinden van

de lichtste èn de tweede kan dus vlugger dan het vinden van de lichtste èn de zwaarste.

Opgave 22 (Eigen algoritme voor sorteren op gewicht) Zie opgave 1.

Opgave 23 (Speel Algoritme S1 na) Ze moeten de stappen van Algoritme S1 *precies* volgen. Het kan verwarrend zijn dat in stap 3(a) bij uitvoering van Algoritme L1 ook de namen A en B weer gebruikt worden, maar voor andere dingen dan in Algoritme S1. Het zijn *lokale* variabelen. Daarom is het goed een ander kind de leiding te geven over uitvoering van Algoritme L1.

Verder kan het zijn dat er wegingen optreden die al eerder zijn gedaan. Deze moeten gewoon opnieuw uitgevoerd en meegeteld worden.

Opgave 24 (Hoeveel wegingen doet Algoritme S1?) Groep A begint met N kinderen en wordt bij elke herhaling van stap 3(b) precies één kleiner. Algoritme L1 wordt dus precies N keer uitgevoerd in stap 3(a), telkens met een groep A die één kleiner is. Het aantal wegingen gedaan bij Algoritme L1 is één minder dan de grootte van groep A. Het totaal aantal wegingen komt daarmee op

$$(N-1)+(N-2)+\cdots+2+1+0 \;=\; \frac{1}{2}N(N-1)$$

Dit zijn de driehoeksgetallen. Eén kind extra kost N wegingen extra.

Opgave 25 (Speel Algoritme S2 na) Merk op dat beide groepen A en B *echt kleiner* zijn dan de oorspronkelijke groep. Daardoor zal het splitsen ooit ophouden. Algoritme S2 is *recursief*. Daarom is het van belang om de afzonderlijke activaties ervan goed in de gaten te houden. Bij dit algoritme valt dat nog wel mee, omdat de volgorde van stappen 2 en 3 er niet toe doet.

Opgave 26 (Hoeveel wegingen doet Algoritme V1 en hoeveel Algoritme S2?) Dit is best lastig. Stel rij A bestaat uit M kinderen en rij B uit N kinderen. In stap 3 van Algoritme V1 wordt bij elke herhaling één weging gedaan en verhuist één kind naar rij C.

Het 'slechtste' geval doet zich voor als de herhaling in stap 3 pas eindigt als de ene rij leeg is en de ander nog maar *één* kind bevat. Algoritme V1 vergt op zijn 'slechtst' dus $M+N-1$ wegingen.

Het 'beste' geval doet zich voor als stap 3 al eindigt zodra de *kortste* rij leeg is terwijl de andere rij onveranderd is gebleven. Dit kost dan $\min\{M, N\}$ wegingen.

Laat $W(N)$ het 'worst case' aantal wegingen bij Algoritme S2 zijn (deze notatie is niet van belang voor de uitleg; die kun je beter direct aan de tabel ophangen). Dan is $W(1) = 0$. Verder is $W(2N) = 2W(N) + 2N - 1$. Hiermee kun je de tabel al invullen voor de machten van twee. Er geldt dan $W(2^K) = (K-1)2^K + 1$. Met name geldt $W(1024) = 9217$.

Voor het 'best case' aantal wegingen $B(N)$ vind je net zo $B(2N) = 2B(N) + N$ en derhalve $B(2^K) = K2^{K-1}$.

Verder geldt $W(2N+1) = W(N) + W(N+1) + 2N$ en $B(2N+1) = B(N) + B(N+1) + N$. Daarmee zijn de kleine gevallen in de tabel in te vullen.

Voor $2^{K-1} \leq N < 2^K$ geldt $W(N+1) = W(N) + K$. Of equivalent: voor $2^{K-1} < N \leq 2^K$ geldt $W(N-1) = W(N) - K$. Je kan dat illustreren aan de hand van de tabel. Met name geldt $W(1000) = W(2^{10}) - 24 \times 10$. $W(N)$ loopt dus 'iets' harder op dan lineair. Na elke macht van twee neemt de onderlinge afstand met één toe. Voor $B(N)$ ligt dat wat lastiger.

Opgave 27 (Sorteer alle kinderen met Algoritme S2) Reken vooraf even uit hoeveel wegingen dit maximaal vergt. Zorg voor voldoende balansen om de sorteertijd in de perken te houden.

Samenvatting van de tellingen (voor het slechtste geval):

	aantal wegingen bij Algoritme ...							
N	L1	LZ1	LZ2	LZ3	LT2	S1	S2	S2 best
1	0	–	–	–	–	0	0	0
2	1	2	1	1	1	1	1	1
3	2	4	3	3	3	3	3	2
4	3	6	5	4	4	6	5	4
5	4	8	7	6	6	10	8	5
6	5	10	9	7	7	15	11	7
7	6	12	11	9	8	21	14	9
8	7	14	13	10	9	28	17	12
32	31	62	61	46	35	496	129	80
1000	999	1998	1997	1498	1008	499500	8977	4932
1001	1000	2000	1999	1500	1009	500500	8987	4938
1002	1001	2002	2001	1501	1010	501501	8997	4945

Invultabel bij *Algoritmen met een balans*

	Aantal wegingen bij Algoritme …							
N	L1	LZ1	LZ2	LZ3	LT2	S1	S2	S2 best
1		—	—	—	—			
2								
3								
4								
5								
6								
7								
8								
32								
1000								
1001								
1002								

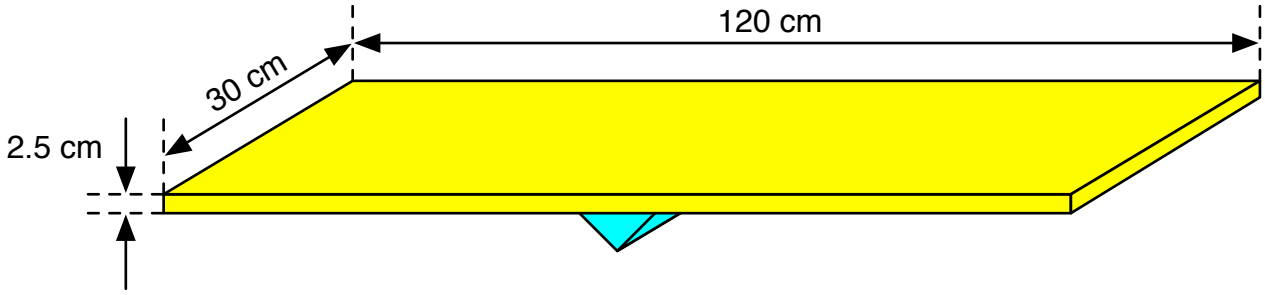
Bouwtekening voor wipbalans

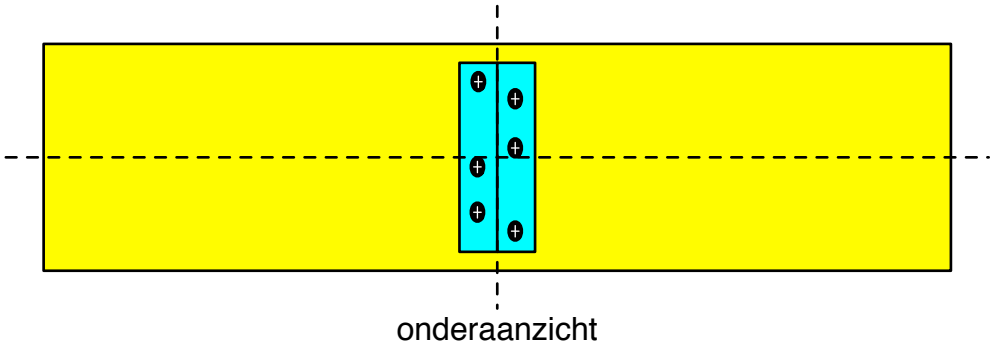
Benodigdheden:

- houten plank circa 120 cm x 30 cm x 2.5 cm
- stalen hoekprofiel 25 cm x 4 cm x 4 cm
- 6 houtschroeven

Eventueel nog

- zelfklevend plastic beschermfolie
- rubber stootkussentjes





onderaanzicht

zijaanzicht