Hulp bij het Maken van een Verslag voor Algoritmiek

Tijn Witsenburg, Rudy van Vliet

February 17, 2025

1 Introductie

Dit LaTeX document is voor studenten van het vak Algoritmiek een leidraad voor het maken van het verslag behorende bij de practicumopdrachten. Enerzijds is er een suggestie voor een duidelijke verhaallijn en anderzijds worden enkele practische tips gegeven.

2 Probleemstelling

In dit gedeelte leg je uit wat het probleem is en hoe dit opgelost gaat worden. Dit is het algemene probleem en is dus een beschrijving van het spel waarvoor de beste zet gezocht moet worden of de situatie waarvoor een oplossing moet worden gevonden. Leg hierin in ieder geval de volgende zaken uit:

1. de beginsituatie
2. alle regels
3. het uiteindelijke doel

3 Bewijzen uit de Opdracht

Vaak wordt in de opdracht ook een aantal bewijzen gevraagd. Dit is een mooi punt om deze te beantwoorden. Gebruik bijvoorbeeld voor elke vraag een aparte *subsection*.

3.1 Zet hier waar het eerste bewijs over gaat

Het is handig om eerst de te bewijzen stelling te vermelden.

Stelling 1 *In een bepaald soort situatie is het altijd zo dat er iets geldt.*

Nadat je de stelling neergezet hebt, dien je deze te bewijzen. Let op, soms kan het natuurlijk zo zijn dat de gestelde vraag niet letterlijk als stelling te gebruiken is. In dat geval mag je er zelf een stelling van maken. Stel bijvoorbeeld dat de vraag is:

> Toon aan dat er minstens zoveel stukken van dit soort nodig zijn om dit of dat te bereiken.

Dan kun je dit natuurlijk makkelijk omschrijven naar een stelling die te bewijzen is. Zo'n stelling wordt dan bijvoorbeeld:

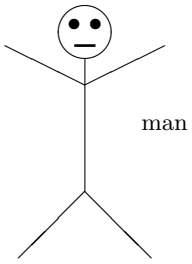

Figure 1: Voorbeeld van een plaatje dat gemaakt is in de *picture*-omgeving

Stelling 2 *Er zijn minstens zoveel stukken van dit soort nodig om dit of dat te bereiken.*

Wat verder nog handig is, is dat je door het gebruik van de commando's `\label` en `\ref` een verwijzing kunt maken naar stelling 1. Ook kun je op dezelfde manier bijvoorbeeld verwijzen naar de spelregels zoals uitgelegd in hoofdstuk 2. Het grote voordeel is dat LaTeX alles automatisch nummert en je je dus geen zorgen hoeft te maken, dat wanneer je ergens iets tussenvoegt, je verwijzingen nog wel kloppen.

3.2 Zet hier waar het tweede bewijs over gaat

Stelling 3 *Soms is iets handiger met een plaatje te verduidelijken.*

Zoals in Figuur 1 te zien is, is stelling 3 waar. Het tekenen in de *picture*-omgeving is vaak veel werk, maar als je eenmaal door hebt hoe het moet, kun je er van alles mee maken. Het is soms een beetje puzzelen hoe alles uitkomt. Vaak is het het handigst om eerst op ruitjespapier even te tekenen wat je wilt hebben en dan overal de coördinaten bij te zetten. Dan kun je makkelijk je plaatje opbouwen en ben je een stuk minder tijd kwijt met steeds een lijn opnieuw trekken, compileren en de *pdf* bekijken net zo lang tot het goed is.

3.3 Zet hier waar het derde bewijs over gaat

Stelling 4 *Een plaatje kan op verschillende manieren worden ingevoegd.*

In paragraaf 3.2 is te zien hoe je een plaatje maakt in de *picture*-omgeving. Figuur 2 laat zien dat er ook een andere methode is om figuren in het verslag te voegen en dus dat stelling 4 waar is.
In sommige versies van LaTeX kun je alleen *eps*-files als plaatje invoeren. Dit heeft te maken met hoe uiteindelijk alles omgezet wordt (van LaTeX naar *dvi*, naar *ps*, naar *pdf*). Check eerst even hoe dit bij jezelf gaat zodat je niet eerst veel tijd kwijt bent met het maken van een bepaald soort plaatje en dat je dan alles opnieuw moet doen omdat een ander soort plaatje moet maken en je het eerste plaatje niet om kunt zetten. Zorg ook dat elk plaatje dat ingevoegd wordt, in dezelfde directory staat, want anders zal deze uiteraard niet gevonden worden of je moet het pad goed opgeven.

3.4 Zet hier waar het vierde bewijs over gaat

Stelling 5 *Soms heb je voor een bewijs wiskundige notaties nodig.*

Figure 2: Voorbeeld van een plaatje dat ingevoegd is met het commando *includegraphics*

Het kan gebeuren dat er in het bewijs een berekening wordt gevraagd. Dan is het mooi dat LaTeX speciaal gemaakt is om mooi en relatief simpel wiskundige notaties te gebruiken. Een en ander wordt duidelijk gemaakt in de volgende formule:

$$f'(x) = \sum_{i=0}^{n} \left(\frac{x^2 + sin(x)}{\sqrt[n-i]{x^2 - x}} + n \cdot i \right) \tag{1}$$

Waarbij duidelijk vermeld dient te worden dat het doel van formule 1 louter is het demonstreren van de vele opties die er zijn, en niet het weergeven van iets wiskundigs interessants.
Verder dient vermeld dat wanneer je graag variabelen zoals i of n in je lopende tekst wil gebruiken, dat je wel moet zorgen dat ze in *math-mode* komen door ervoor en erna een '$' te zetten. Ook kun je een wiskundige afleiding mooi onder elkaar zetten:

$$\begin{aligned} 2x^2 + x + 1 &= x^2 + 2x + 3 \\ x^2 - x - 2 &= 0 \\ (x-2) \cdot (x+1) &= 0 \\ x = 2 &\vee x = -1 \end{aligned}$$

4 Oplossingsmethode

In dit deel kun je omschrijven hoe jij het probleem gaat oplossen. Als eerste kun je de algemene oplossingsmethode bespreken. Let hierbij ook op begin- en eindsituaties en eventuele uitzonderingen. Het zou heel goed kunnen dat je graag wilt verwijzen naar een tekst die je gebruikt hebt om het probleem op te lossen, bijvoorbeeld het Algoritmiekboek van Levitin [1].
Het zou goed kunnen dat je voortvloeiend uit de bewijzen die je in hoofdstuk 3 hebt geleverd, je programma kunt verbeteren. Leg hier ook kort uit wat dit voor je programma betekent. Als je zelf nog meer dingen hebt verzonnen om de prestaties van je programma te verbeteren, dan moet je die hier ook vermelden. Eventueel kun je een en ander verduidelijken door een stuk (pseudo-)code in te voegen zoals te zien is in Figuur 3.

5 Resultaten en Conclusies

De grote klapper van je verslag is natuurlijk het gedeelte met de resultaten. Jullie zitten in het wetenschappelijk onderwijs, en in de wetenschap zijn resultaten erg

```
// De verbatim-omgeving zet alle tekst letterlijk neer
// op een manier dat deze er uit ziet als code.

void functie(int variabele)
{
    int i = 0;
    if (i<variabele)
        cout << "positief" << endl;
    else
        cout << "negatief of een nul" << endl;
}
```

Figure 3: Voorbeeld van een stukje code in de *verbatim*-omgeving.

NAAM	GETAL 1	GETAL 2	GETAL 3	RESULTAAT
Jan	1	2	3	Baard
Pier	1234	119	40	Baard
Walter	14	12	56	Geen Baard
Tjoris	20	78	123	Baard
Jeannette	5	19	14	Geen Baard
Corneel	19	18	1234	Baard

Table 1: Voorbeeld van een tabel in LaTeX.

belangrijk. Deze moeten uiteraard correct zijn en op een duidelijke manier gepresenteerd worden. Een handige manier is dan vaak de tabelvorm zoals te zien is in Tabel 1.
Zet hier ook eventuele andere dingen die je nog opgevallen zijn tijdens het maken van de opdracht en waar eerder geen plaats voor was. Wat bijvoorbeeld op het eind van dit verslag opvalt, is dat het lijkt alsof LaTeX alle figuren en tabellen op de raarste plekken neerzet. Dat is misschien ook wel zo, zeker als je relatief veel figuren en tabellen hebt in verhouding tot de tekst, maar daar hoef je je geen zorgen over te maken. Zeker niet als je de commando's `\label` en `\ref` consequent gebruikt.

References

[1] Anany Levitin. *Introduction to The Design and Analysis of Algorithms.* Pearson Education, Inc., 2012.

A Source code in verslag

Wanneer je je complete programma in je verslag wilt opnemen, kan dat overzichtelijk met het *package* `listings`, bijvoorbeeld als volgt:

```
1 // Say hello to the world.
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
```

```
8     cout << "Hello world" << endl;
9     return 0;
10 }
```

Je hebt hierbij vele mogelijkheden om de weergave van je code aan te passen, zie bijvoorbeeld Wikibooks. Let wel goed op bij het gebruik van tabs in je code. Die zouden soms zomaar kunnen verdwijnen, en dat is waarschijnlijk niet wat je wilt...