OSM-S Opdracht "Scheduling" 2023-2024 S1-P1

Bram Knippenberg Joost Kraaijeveld Jorg Visch

13 september 2023

1 Inleiding

Planning en scheduling zijn twee belangrijke onderwerpen binnen de technische informatica. *Planning* gaat over *WAT* er gedaan moet worden. Deze opdracht gaat over *scheduling*: *WANNEER* het gedaan moet worden. De opdracht bestaat uit twee delen, een programma (in tweetallen) en het schrijven van een essay (individueel).

2 Het programma (tweetallen)

De opdracht luidt als volgt: *Schrijf een programma dat job-shop scheduling problemen oplost met behulp van het "Minimum" of "Least Slack" algoritme.* Als inspiratie voor het algoritme kan je (Russell en Norvig 2003) gebruiken, zie "Planning and acting in the real world" in de bijlage. Ook biedt wikipedia enige hulp:

- http://nl.wikipedia.org/wiki/Job_sequencing (Wikipedia-bijdragers 2017)
- http://en.wikipedia.org/wiki/Least_slack_time_scheduling (Wikipedia contributors 2019)

2.1 Input

Het programma bevat tenminste de classes JobShop, Job en Task, waarbij de JobShop Jobs heeft en een Job Tasks heeft. De gegevens (jobs, tasks en aantal machines) moeten vanuit een bestand worden ingelezen. De naam van het bestand wordt meegegeven als argument aan de applicatie en mag niet hardgecodeerd in de applicatie zitten. Gebruik voor het lezen van het bestand de C++-standaard `std::ifstream` class. Zie (Stroustrup 2013, sec. 10.7) en (cppreference 2020) voor uitleg in het gebruik. Het inputbestand heeft een vast formaat, de definitie van de input staat beschreven in de bijlage "Input in ABNF-notatie". Hieronder een voorbeeld van zo'n bestand en toelichting.

```
6 6
2  1  0  3  1  6  3  7  5  3  4  6
1  8  2  5  4 10  5 10  0 10  3  4
2  5  3  4  5  8  0  9  1  1  4  7
1  5  0  5  2  5  3  3  4  8  5  9
2  9  1  3  4  5  5  4  0  3  3  1
1  3  3  3  5  9  0 10  4  4  2  1
```

De eerste regel geeft aan hoeveel jobs en machines er zijn (hier in beide gevallen 6, maar dat varieert per casus). De overige regels bevatten telkens de taken van een job. Jobs hebben als id het volgnummer in de lijst, i.e. de eerste job (beginnend met `2 1 0`…) heeft 0 als id, de laatste job (beginnend met `1 3 3`…) heeft id 5. Job-regels worden gescheiden met new-lines.

De job-regel bevat de taken. De taken staan in de volgorde van uitvoering. Taken hebben als id het volgnummer in de regel, de machine waarop deze moet worden uitgevoerd (eerste getal) en de tijdsduur van de bewerking van die machine (tweede getal), i.e. de eerste job (`2 1 0 3 1 6 3 7 5 3 4 6`) heeft als eerste taak taak 0 op machine 2 met een tijdsduur van 1 en als tweede taak taak 1 op machine 0 met een tijdsduur van 3. De taken en de tijdsduur van de bewerking worden gescheiden door een willekeurige hoeveelheid "whitespace". Op OnderwijsOnline vind je bij deze opdracht ook een tekstbestand met een aantal voorbeeldconfiguraties (`voorbeeldconfiguraties.txt`).

Je mag alleen gebruik maken van classes en functies die standaard bij een C++-compiler (C++17) zitten. Er mag dus geen gebruik gemaakt worden van externe libraries als Boost. Ook mag er geen platform (Windows/Linux/mac) afhankelijke code gebruikt worden. Er zijn pluspunten te verdienen voor zij die de `std::regex` classen gebruiken (zie (Stroustrup 2013, sec. 9.4))

2.2 Output

De output [1] van het programma moet een overzicht per job zijn van de start- en eindtijden van de jobs in volgorde van het job-id, alles gescheiden door whitespace (spaties en/of tabs). Deze output moet exact volgens de gegeven beschrijving zijn, dus geen eigen extra regels of waarden toevoegen. De definitie van de output staat beschreven in de bijlage "Output in ABNF-notatie"

Een voorbeeld van zo'n output kan het volgende zijn (dit is niet de output van het voorbeeld hierboven, maar van een andere configuratie):

```
0  1  288
1  8  291
2  5  258
3  5  169
4  9  100
5  0  300
```

De starttijd is hierbij het tijdstip van het starten van de eerste taak van een job waarbij de start van de allereerste taak van de alleerste job geldt als tijdstip 0. Het tijdstip van het einde van de taak is het tijdstip waarop de laatste taak van de job eindigt. Als tijdseenheid wordt dezelfde eenheid gebruikt als waar de taak-duur in is aangegeven.

3 Het essay (individueel)

In de literatuur over scheduling wordt vaak een onderscheid gemaakt in een tweetal categoriën "scheduling policies":

- "fair scheduling policies"
- "priority scheduling policies"

In detail verschillen deze "policies" op diverse manieren, zoals beschreven staat in (Douglass 2004):

> Determining a scheduling strategy is crucial for efficient scheduling of real-time systems. Systems no more loaded than 30 percent have failed because of poorly chosen scheduling policies. Scheduling policies may be *stable*, *optimal*, *responsive*, and/or *robust*. A *stable* policy is one in which, in an overload situation, it is possible to a priori predict which task(s) will miss their timeliness requirements. Policies may also be *optimal*[2]. A *responsive* policy is one in which incoming events are handled in a timely way. Lastly, by *robust*, we mean that the timeliness of one task is not affected by the misbehavior of another…

Beantwoord in een essay de volgende vragen over het onderwerp "scheduling policies".

1. Beschrijf bondig en in eigen bewoordingen het fundamentele verschil tussen "fair scheduling policies" en "priority scheduling policies".
2. Beschrijf met eigen woorden wat de termen "stable", "optimal", "responsive" en "robust" betekenen binnen de context van scheduling.
3. Kies van beide twee policies (fair en priority) tenminste twee varianten (dus vier in totaal), uitgezonderd het "Least Laxity" of "Minimum/Least Slack" algoritme, want dat is het onderwerp van deze opdracht. Beschrijf de door jou gekozen algoritmen op systematische wijze op hun kenmerken. Doe dat aan de hand van de volgende stappen:
   a. Een korte algemene beschrijving van het schedulingalgoritme.
   b. Beschrijf systematisch hoe het algoritme scoort op de termen "stable", "optimal", "responsive" en "robust".

[1] Met de *output* wordt bedoeld de *output naar de console*.
[2] An optimal policy is one that can schedule a task set if it is possible for any other policy to do so. (Douglass 2004)

c. Beschrijf voor welke problemen het gekozen algoritme bij uitstek *geschikt* is en voor welke problemen het *ongeschikt* is. Beargumenteer dat aan de hand van de eerder gegeven termen ("stable", "optimal" etc.).

Beantwoord de bovenstaande vragen over "scheduling policies" kort, in ongeveer 4 bladzijden (richtlijn). Gebruik hierbij *niet* een (letterlijke) vertaling van het eerste de beste Wikipedia-lemma of een andere, waarschijnlijk "random" via Google gevonden, webpagina. Geef voorbeelden uit het domeingebied van de programmeeropdracht (met jobs, machines en taken). Kortom, verras jezelf en doe iets wilds: *lees* een aantal verschillende webpagina's of wellicht boeken en *denk na* om tot slot *in eigen woorden* iets op te schrijven. Naast dat je werk inhoudelijk voldoende moet zijn, dient het ook aan de gangbare richtlijnen te voldoen (AIM-controlekaart, bronvermeldingen in APA, etc.).

4 Inleveren

Het inleveren (in iSAS) kan uiterlijk op de vrijdag in lesweek 5, zie het toetsrooster in iSAS voor de definitieve deadline.

Lever je code in als een export van een *eclipse-project*, zodat de docent die kan importeren in zijn Eclipse-omgeving. Je code compileert zonder errors en onnodige warnings. Je code wordt gecompileerd met de warning levels `-Wall -Wextra -Wconversion`, als je code een warning geeft, dan is uit het commentaar in de code op te maken waarom deze warning niet is opgelost. Daarnaast wordt natuurlijk je code gecompileerd met de C++17-standaard.

Het essay lever je in als *PDF*, zodat deze op elk platform te lezen is. Het staat je vrij om een text editor van eigen keuze te gebruiken.

Te laat inleveren betekent dat je werk niet wordt nagekeken, dus een onvoldoende cijfer.

5 Bijlagen

5.1 ABNF input en output van de applicatie

5.1.1 Input in ABNF-notatie

De definitie in ABNF(IETF Tools 2008) (Wikipedia contributors 2020) van de input is:

```
input           = config jobs   ; full input of the jobshop scheduling

config          = job-count 1*WSP machine-count *WSP line-feed
                                ; First line of input file

jobs            = job-line *(line-feed job-line) [line-feed]
                                ; Job configuration

job-count       = 1*DIGIT       ; number of jobs in input file
machine-count   = 1*DIGIT       ; number of available machines

job-line        = task *(WSP task) *WSP
                                ; Configuration of one job (sequence of tasks)

task            = *WSP machine-id 1*WSP duration
                                ; configuration of a task

machine-id      = 1*DIGIT       ; id of a machine
duration        = 1*DIGIT       ; duration of a task

line-feed       =   LF / CRLF   ; line feeds

DIGIT           =  %x30-39      ; 0-9
WSP             =  SP / HTAB    ; white space
CRLF            =  CR LF        ; newline
```

```
CR              =  %x0D         ; carriage return
LF              =  %x0A         ; linefeed
SP              =  %x20         ; space
HTAB            =  %x09         ; horizontal tab
```

(DIGIT, WSP, CRLF etc. zijn core rules van de ABNF standaard), zie https://tools.ietf.org/html/rfc5234#page-13.

5.1.2 Output in ABNF-notatie

De definitie in ABNF[IETF Tools (2008)](Wikipedia contributors 2020) van de output is:

```
output      =   1*job-result    ; full output of the jobshop scheduling

job-result  =   job-id 1*WSP start-time 1*WSP end-time *WSP line-feed
                                ; per job the start and end time

job-id      =   1*DIGIT         ; id of the job
start-time  =   time-stamp      ; start time of the scheduled job
end-time    =   time-stamp      ; end time of the scheduled job

time-stamp  =   1*DIGIT         ; time stamp
line-feed   =   LF / CRLF       ; line feeds

DIGIT       =  %x30-39          ; 0-9
WSP         =  SP / HTAB        ; white space
CRLF        =  CR LF            ; newline
CR          =  %x0D             ; carriage return
LF          =  %x0A             ; linefeed
SP          =  %x20             ; space
HTAB        =  %x09             ; horizontal tab
```

(DIGIT, WSP, CRLF etc. zijn core rules van de ABNF standaard), zie https://tools.ietf.org/html/rfc5234#page-13.

5.2 Planning and acting in the real world

Zie het artikel “Planning and acting in the real world” (Douglass 2004) voor informatie over het minimum/least slack Algorithm. Dit artikel is te vinden op OnderwijsOnline.

Bronnenlijst

cppreference. 2020. ‘Input/output library - cppreference.com’. 25 april 2020. http://en.cppreference.com/w/cpp/io.

Douglass, Bruce Powel. 2004. *Real time uml: Advances in the uml for real-time systems (3rd edition)*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

IETF Tools. 2008. ‘Augmented BNF for Syntax Specifications: ABNF’. januari 2008. https://tools.ietf.org/html/rfc5234.

Russell, Stuart, en Peter Norvig. 2003. *Artificial intelligence: A modern approach*. 2nd edition. Prentice-Hall, Englewood Cliffs, NJ.

Stroustrup, B. 2013. *A tour of c++ (second edition)*. C++ in-depth series. Pearson Education.

Wikipedia contributors. 2019. ‘Least slack time scheduling’. 24 juni 2019. http://en.wikipedia.org/wiki/Least_slack_time_scheduling.

———. 2020. ‘Augmented Backus–Naur form’. 27 juni 2020. https://en.wikipedia.org/wiki/Augmented_Backus-Naur_form.

Wikipedia-bijdragers. 2017. ‘Job sequencing’. 17 september 2017. http://nl.wikipedia.org/wiki/Job_sequencing.