Colocação de Livros
Resolução de Problemas de Otimização utilizando Programação em Lógica com Restrições

Francisco Rodrigues (up2013305627) and Marta Lopes (ei12106)

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal
http://www.fe.up.pt/

Resumo Este artigo foi realizado para explicar o desenvolvimento do segundo projeto da Unidade Curricular de PLOG. O tema escolhido foi Colocação de Livros e é um problema de optimização. *abstract* environment.

Keywords: livros, otimizar, organizar, mieic, prolog, plog, feup

1 Introdução

O objetivo deste projeto é consolidar o conhecimento que foi adquirido na segunda parte da matéria de Programação em Lógica, sendo esta sobre Restrições. Foram-nos dados vários temas em que seriam Problemas de Decisão ou Otimização e nós optamos por escolher um problema de Otimização. Inicialmente achamos que este tema seria bastante fácil mas ao longo do desenvolvimento do projeto fomos encontrado imensas dificuldades, algumas delas não foram ultrapassadas.

Neste artigo vamos então descrever de forma detalhada, a forma como decidimos abordar este problema de organização de livros de forma otimizada.

2 Descrição do Problema

Pretende-se organizar livros por várias prateleiras. Cada prateleira vai ter uma largura e uma altura que não podem ser ultrapassados pelos livros que forem colocados lá. Devemos minimizar a largura por preencher em cada prateleira. Em cada prateleira os livros com o mesmo autor terão que ficar juntos, e preferencialmente ordenados por data de publicação. Devemos otimizar a ordenação de livros de acordo com várias medidas.

3 Abordagem

Para a resolução deste problema foram tentadas várias abordagens sendo que tivemos alguma dificuldade em definir como seria a melhor maneira de organizar o problema de modo a que pudéssemos aplicar as restrições necessárias para uma solução válida.

Sendo assim, tentamos abordar o problema de duas formas: uma em que teríamos em conta que os livros teriam larguras diferentes entre si, e outra em que os livros teriam todos largura de 1.

3.1 Primeira Abordagem

Como já foi dito, nesta abordagem, os livros teriam larguras diferentes entre si o que nos permitiu colocar os livros nas prateleiras verificando se a largura da prateleira não era ultrapassada. Também era possível verificar se a altura do livro não ultrapassaria a altura da prateleira, se ultrapassasse o livro não seria colocado nas prateleiras, mas iria ser apresentada uma solução na mesma, sem esse livro. No entanto nesta abordagem não foi possível organizar os livros por autores ou data de publicação mas foi feita a minimização das larguras por preencher em cada prateleira.

Os livros são então criados como demonstrados em baixo.

```
%id, nome, autor, tema, ano, largura, altura
book(1,'EnsaioSobreACegueira','Saramago','Romance',1995,3,15).
book(2,'Lusiadas','Camoes','Classic',1572,3,15).
book(3,'OsMaias','Eca_de_Queiros','Classic',1888,2,15).
book(4,'AViagemDoElefante','Saramago','Romance',2008,2,15).
book(5,'OEvangelhoSegundoJesusCristo','Saramago','Romance'
    ↪ ,1991,2,15).
book(6,'ACaverna','Saramago','Romance',2000,2,15).
```

3.1.1 Variáveis de Decisão

O espaço ocupado pelo livro em cada prateleira vai depender da largura desse livro, sendo que cada espaço na prateleira corresponde a 1 unidade, se um livro tiver 2 de largura, irá ocupar duas posições na prateleira, então, como o que é tido em conta é apenas se o livro cabe ou não numa determinada prateleira, e se a largura dessa prateleira está minimizada, a **variável de decisão** irá ser a prateleira para onde o livro vai e quantas posições nessa prateleira ocupa.

A **variável de domínio** será o número de prateleiras usadas que vai desde 1 até ao número de prateleiras existentes: *domain(ShelveSpacesFlattened, 1, NrShelves).*

3.1.2 Organização da lista de livros

Apesar de não ser aplicado como restrição é feita uma organização dos livros que vão ser usados para serem aplicadas as restrições, dado que altura de cada livro não pode ultrapassar a altura máxima da prateleira *MaxHeight*, vai ser tido em conta se a altura do livro, se ultrapassar a altura da prateleira não vai ser adicionado à lista de livros nas prateleiras.

```
checkBookHeight([],[],_).
checkBookHeight([Book|T], BookResult, MaxHeight):-
        nth0(3, Book, BookHeight),
        BookHeight > MaxHeight,
        nth0(0, Book, ID),
        book(ID, Name, _, _,_,_,Height),
        write('The book '),
        write(Name ),
        write(' was not added because it has an height of '),
        write(Height),
        checkBookHeight(T, BookResult, MaxHeight).


checkBookHeight([Book|T], [BResH|BResT],MaxHeight):-
        BResH = Book,
        checkBookHeight(T, BResT, MaxHeight).
```

3.1.3 Restrições

A largura do conjunto de livros na prateleira não pode ultrapassar a largura máxima da prateleira *MaxWidthShelf*

```
initializeShelves(_,Max, _, Max).
initializeShelves(SSpaces, NrShelves, MaxWidthShelf, CurrShelf
    ↪ ):-
        NextShelf is CurrShelf + 1,
        count(NextShelf, SSpaces, #=<, MaxWidthShelf),
        initializeShelves(SSpaces, NrShelves, MaxWidthShelf,
            ↪ NextShelf).
```

Na restrição **count** nós definimos que a prateleira a ser atribuída tem que ser uma prateleira que ainda tenha espaço disponível para o livro.

3.1.4 Função de Avaliação

A largura por preencher em cada prateleira terá que ser mínima

```
getCostShelfUsage(ShelveSpacesFlattened, MaxWidthShelf,
    ↪ NrShelves, Res):-
        length(ShelfList, NrShelves),
```

```
getEmptyShelveSpaces(ShelveSpacesFlattened, 1,
    ↪ ShelfList, MaxWidthShelf),
removeEmpty(ShelfList, MaxWidthShelf, Res).
```

Nesta função é calculado o custo. Em *getEmptyShelveSpaces* vai buscar os espaços vazios em cada prateleira e depois em *removeEmpty* vai calcular o espaço vazio por cada prateleira, se a prateleira estiver completamente vazia o espaço desta não vai ser contabilizado. O custo é depois devolvido em *Res*.

3.1.5 Estratégia de Pesquisa

A organização dos livros vai ser então para que a largura restante em cada prateleira seja minima, sendo assim teremos que minimizar o custo calculado em *getCostShelfUsage*. Vão ser dadas as prateleiras para cada livro de acordo com este *minimize*

```
getCostShelfUsage(ShelveSpacesFlattened, MaxWidthShelf,
    ↪ NrShelves, Cost),

labeling([minimize(Cost), time_out(60000,_)],
    ↪ ShelveSpacesFlattened).
```

3.2 Segunda Abordagem

Nesta abordagem os livros terão todos largura de 1, teremos então em conta o autor do livro, a altura e o ano de publicação. Apesar de existir informação relativa ao tema, esta não é usado. Se a altura de um livro ultrapassar a altura da prateleira, nesta abordagem, não existirá nenhuma solução pois definimos que todos os livros têm que ser adicionados.

Os livros são então criados como demonstrados em baixo.

```
book(1,'EnsaioSobreACegueira',1,'Romance',1995,16).
book(2,'Lusiadas',2,'Classic',1572,13).
book(3,'OsMaias',3,'Classic',1888,14).
book(4,'AViagemDoElefante',1,'Romance',2008,15).
book(5,'OEvangelhoSegundoJesusCristo',1,'Romance',1991,15).
book(6,'ACaverna',1,'Romance',2000,15).

author(1, 'Saramago').
author(2, 'Camoes').
author(3, 'Eca_de_Queiros').
```

3.2.1 Variáveis de Decisão

A **variável de decisão** vai ser que posição o livro vai ter nas prateleiras, sendo que apenas existe uma lista que engloba todas as prateleiras e assumimos que desde x até x+*WidthShelf* será uma prateleira. A **variável de domínio** será desde 1 até ao número de espaços nas prateleiras.

3.2.2 Restrições

A posições dadas a cada livro terão que ser diferentes usando *all_distinct(BooksRes)*. Sendo que *BookRes* é a lista dos livros com a posição associada.

Os livros do mesmo autor têm que estar todos juntos, e os autores estarão organizados por ordem alfabética.

```
sortByAuthor([A1, A2 | ARest]):-
        A1 #=< A2,
        sortByAuthor([A2|ARest]).
sortByAuthor([_]).
```

É passada dos autores de todos os livros e vai ser verificado que o autor do próximo livro terá que ser o mesmo ou o próximo autor tendo em conta a ordem alfabética.

Dentro dos livros do mesmo autor, estes terão que estar organizados por alturas, otimizando também as variações de altura.

```
sortByHeight([H1, H2 | HRest], [A1, A2 | ARest]):-
        (A1 #= A2 #=> H1 #=< H2),
        sortByHeight([H2 | HRest], [A2 | ARest]).
sortByHeight([_], [_]).
```

É passada uma lista dos autores e das alturas de todos os livros e vai ser verificado se o autor é o mesmo nos livros adjacentes, se for, organiza os livros por alturas.

Se o autor e altura dos livros for a mesma, vai ordenar por ano de publicação.

```
sortByHeight([H1, H2 | HRest], [A1, A2 | ARest]):-
        (A1 #= A2 #=> H1 #=< H2),
        sortByHeight([H2 | HRest], [A2 | ARest]).
sortByHeight([_], [_]).
```

É passada uma lista dos autores, das alturas e dos anos de publicação de todos os livros e vai ser verificado se o autor e a altura são iguais nos livros adjacentes, se forem, organiza por data de publicação.

3.2.3 Estratégia de Pesquisa

A organização dos livros vai ser então ordenada por autores, alturas entre autores, e ano de publicação entre alturas, não é usada nenhuma função para minimizar o maximizar o curso, porque não foi necessário nesta abordagem. De acordo com estas especificações vai ser então atribuída a posição a cada livro.

```
labeling([time_out(60000,_)], BooksRes)
```

4 Visualização da Solução

4.1 Primeira abordagem

Nesta primeira abordagem, será mostrado a informação sobre os livros que foram adicionados, dizendo em que prateleira é que foram adicionados e a sua largura. Vai também referir o espaço que vai sobrar em cada prateleira, para mostrar a minimização. Se um dos livros ultrapassar a altura máxima e não for adicionado, vai mostrar uma mensagem a dizer que livro não foi adicionado. Podemos ver no exemplo em baixo.

```
The book A Caverna was not added because it has an height of 25

Book 1
Ensaio Sobre a Cegueira by Saramago is in shelf 1 with a witdth of 3

Book 2
Lusiadas by Camoes is in shelf 1 with a witdth of 3

Book 3
Os Maias by Eca de Queiros is in shelf 1 with a witdth of 2

Book 4
A Viagem do Elefante by Saramago is in shelf 1 with a witdth of 2

Book 5
O Evangelho segundo Jesus Cristo by Saramago is in shelf 1 with a witdth of 2


Shelf 1 has 8 empty spaces.

Shelf 2 has 20 empty spaces.

Shelf 3 has 20 empty spaces.

Shelf 4 has 20 empty spaces.
```

4.2 Segunda abordagem

Na segunda abordagem, vai ser mostrada a prateleira e dentro mostrará os livros que estão lá, organizados.

```
------------------------------------------------------------------------------------------
Shelf 1
[O Evangelho segundo Jesus Cristo, escrito por:Saramago, em:1991, com altura de:15]
[A Caverna, escrito por:Saramago, em:2000, com altura de:15]
------------------------------------------------------------------------------------------
Shelf 2
[A Viagem do Elefante, escrito por:Saramago, em:2008, com altura de:15]
[Ensaio Sobre a Cegueira, escrito por:Saramago, em:1995, com altura de:16]
------------------------------------------------------------------------------------------
Shelf 3
[Lusiadas, escrito por:Camoes, em:1572, com altura de:13]
[Os Maias, escrito por:Eca de Queiros, em:1888, com altura de:14]
------------------------------------------------------------------------------------------
```

5 Resultados

Para este projeto criamos as estatísticas para ambas as abordagens, e pudemos verificar que eram influenciadas por diferentes fatores. *Resumptions* será o número de vezes que uma restrição foi concluida, *Entailments* será o número de vezes que uma consequência foi detetada por uma restrição, *Prunings* será o número de opções inviáveis, *Bactracks* é o número de vezes que uma contradição foi encontrada e ele teve que encontrar novas soluções e *Constraints* será o número de restrições criadas.

5.1 Primeira abordagem

A partir dos resultados obtidos conseguimos verificar que o tempo de resolução depende do número de prateleiras e do número de livros a verificar.

	Tempo (segundos)	Resumptions	Entailments	Prunings	Backtracks	Constraints
1000 Prateleiras, 2 Livros	276.45	533968296	300056841	300745347	65750	48024
10 Prateleiras, 2 Livros	1.91	12573	6768	7798	99	168
10 Prateleiras, 24 Livros	64.32	280443363	85947156	148231915	2380252	1196

Tabela 1. Estatisticas na primeira abordagem

5.2 Segunda abordagem

Nesta segunda abordagem podemos observar que o número de prateleiras não vai influenciar as estatísticas.

	Tempo (segundos)	Resumptions	Entailments	Prunings	Backtracks	Constraints
1000 Prateleiras, 6 Livros	0.08	1379	320	625	25	60
4 Prateleiras, 6 Livros	0.08	1379	320	625	25	60
30 Prateleiras, 24 Livros	188.99	996891974	314050679	591063710	16935889	774

Tabela 2. Estatisticas na segunda abordagem

6 Conclusões

6.1 Conclusões

Com este projeto podemos então concluir que usar restrições e funções de avaliação em *Prolog* facilita a resolução de problemas complexos de uma forma mais rápida e simples. Apesar das dificuldades em organizar o pensamento de resolução no inicio deste projeto, conseguimos abordar o problema de duas formas, que apesar de incompletas, acabam por se completar uma a outra. Podemos dizer que aprendemos bastante com este problema de otimização e acreditamos que se tivéssemos conseguido organizar logo de inicio todo o pensamento de resolução de uma maneira mais simples, seria possível ter arranjado uma abordagem que incluísse todas as restrições e otimizações que foram propostas no enunciado.

6.2 Trabalho Futuro

No futuro, gostaríamos de juntar as duas abordagens numa, de forma a que o problema de otimização fosse resolvido por completo.

Referências

1. SICStus Prolog, `https://sicstus.sics.se/sicstus/docs/4.0.1/html/sicstus/index.html`
2. SWI-Prolog documentation, `http://www.swi-prolog.org/pldoc/index.html`

7 Anexos

7.1 Primeira abordagem

```
:-use_module(library(clpfd)).
:-use_module(library(lists)).

%id, nome, autor, tema,
book(1, 'Ensaio sobre a Cegueira',
    ↪                              'Saramago',                  '
    ↪ Romance',       1995, 3, 15).
book(2, 'Lusiadas',
    ↪                                        'Camoes',
    ↪                        'Classic',       1572, 3, 16).
book(3, 'Os Maias',
    ↪                                        'Eca de Queiros',
    ↪           'Classic',       1888, 2, 15).
book(4, 'A Viagem do Elefante',
    ↪                               'Saramago',                  '
    ↪ Romance',       2008, 2, 19).
book(5, 'O Evangelho segundo Jesus Cristo',
    ↪             'Saramago',                'Romance',       1991,
    ↪ 2, 15).
book(6, 'A Caverna',
    ↪                                        'Saramago',
    ↪             'Romance',       2000, 2, 25).


listBooks(List, MaxHeight) :-
        findall([Id, Author,BookW, BookH],
                        book(Id, _, Author, _, _, BookW, BookH
                            ↪ ),
                        ListTemp),
        checkBookHeight(ListTemp, ListTemp2, MaxHeight),
        addShelfSpace(ListTemp2, List).

checkBookHeight([],[],_).
checkBookHeight([Book|T], BookResult, MaxHeight):-
        nth0(3, Book, BookHeight),
        BookHeight > MaxHeight,
        nth0(0, Book, ID),
        book(ID, Name, _, _,_,_,Height),
        nl,
        write('The book '),
        write(Name ),
        write(' was not added because it has an height of '),
        write(Height),nl,
        checkBookHeight(T,  BookResult, MaxHeight).
```

```
checkBookHeight([Book|T], [BResH|BResT],MaxHeight):-
        BResH = Book,
        checkBookHeight(T, BResT, MaxHeight).



getSizeOfBook(SizeOfBook, [_,_,SizeOfBook,_]).

setBookSpaces(Spaces, SizeOfBook):-
        length(Spaces, SizeOfBook),
        startSetEquals(Spaces).

setEquals(_, []).
setEquals(Value, [LH | LT]) :-
        Value #= LH,
        setEquals(Value, LT).

startSetEquals([LH |LT]) :-
        setEquals(LH, LT).



addShelfSpace([], []).
addShelfSpace([LH |LT], [ListH | ListT]) :-
        getSizeOfBook(SizeOfBook, LH),
        setBookSpaces(Spaces, SizeOfBook),
        ListH = [LH, Spaces],
        addShelfSpace(LT, ListT).

getShelvesSpaces([],[]).
getShelvesSpaces([[_, ShelfH] | BooksT], [ShelfH |ShelfT]) :-
        getShelvesSpaces(BooksT, ShelfT).


getEmptyShelveSpaces(_, _, [], _).
getEmptyShelveSpaces(ShelveSpacesFlattened, NrShelf, [ShelfH |
    ↪ ShelfT], MaxWidthShelf):-
        NrShelves1 is NrShelf+1,
        getShelfEmpty(ShelveSpacesFlattened, NrShelf,
            ↪ MaxWidthShelf, ShelfH),
        getEmptyShelveSpaces(ShelveSpacesFlattened, NrShelves1
            ↪ , ShelfT, MaxWidthShelf).


getShelfEmpty(ShelveSpacesFlattened, Shelf, MaxWidthShelf, R)
    ↪ :-
        count(Shelf, ShelveSpacesFlattened, #=, Res),
        R #= MaxWidthShelf - Res.
```

```
removeEmpty(ShelfList, MaxWidthShelf, Res):-
        count(MaxWidthShelf, ShelfList, #=, CounterTemp),
        sum(ShelfList, #=, Total),
        Res #= Total - (CounterTemp * MaxWidthShelf).

getCostShelfUsage(ShelveSpacesFlattened, MaxWidthShelf,
    ↪ NrShelves, ShelfList, Res):-
        length(ShelfList, NrShelves),
        getEmptyShelveSpaces(ShelveSpacesFlattened, 1,
            ↪ ShelfList, MaxWidthShelf),
        removeEmpty(ShelfList, MaxWidthShelf, Res).

livros(ShelveSpaces, NrShelves, MaxWidthShelf, MaxHeight):-
        %length(Books, N),
        listBooks(Books, MaxHeight),

        getShelvesSpaces(Books, ShelveSpaces),

        flatten(ShelveSpaces, ShelveSpacesFlattened),

        getCostShelfUsage(ShelveSpacesFlattened, MaxWidthShelf
            ↪ , NrShelves, EmptySpaces, Cost),

        domain(ShelveSpacesFlattened, 1, NrShelves),

        initializeShelves(ShelveSpacesFlattened, NrShelves,
            ↪ MaxWidthShelf, 0),
        !,
        labeling([minimize(Cost),time_out(60000,_)],
            ↪ ShelveSpacesFlattened),
        printBookInfo(Books, 1),nl,
        printShelfInfo(EmptySpaces, 1),nl,
        write('###_STATISTICS_###'),nl,
        print_time,
    fd_statistics,nl.


/*flatten(Res, List), write(Res),
domain(List, 1, N),


all_distinct(List),

labeling([], List),
write(Res)*/


flatten([],[]).
flatten([LH|LT], Flattened) :-
```

```
        is_list(LH),
        flatten(LH, FlattenedTemp),
        append(FlattenedTemp, LT2, Flattened),
        flatten(LT, LT2).

flatten([LH | LT], [LH | FlattenedT]) :-
        \+is_list(LH),
        flatten(LT, FlattenedT).


initializeShelves(_,Max, _, Max).
initializeShelves(SSpaces, NrShelves, MaxWidthShelf, CurrShelf
    ↪ ):-
        NextShelf is CurrShelf + 1,
        count(NextShelf, SSpaces, #=<, MaxWidthShelf),
        initializeShelves(SSpaces, NrShelves, MaxWidthShelf,
            ↪ NextShelf).


getBookShelf([_, [Shelf|_]], Shelf).


printBookInfo([], _).
printBookInfo([Book|Rest], IdCount):-
        Idnew is IdCount+1,
        getBookShelf(Book, Shelf),
        book(IdCount, Title, Author, _, _, Width, _),
        nl,
        write('Book '),
        write(IdCount),nl,
        write(Title),
        write(' by '),
        write(Author),
        write(' is in shelf '),
        write(Shelf),
        write(' with a witdth of '),
        write(Width),nl,
        printBookInfo(Rest, Idnew).

printShelfInfo([], _).
printShelfInfo([H|T], IdCount):-
        Idnew is IdCount+1,
        nl,
        write('Shelf '),
        write(IdCount),
        write(' has '),
        write(H),
        write(' empty spaces.'), nl,
        printShelfInfo(T, Idnew).
```

```
reset_timer :- statistics(walltime,_).
print_time :-
        statistics(walltime,[_,T]),
        TS is ((T//10)*10)/1000,
        nl, write('Time:_'), write(TS), write('s'), nl, nl.
```

7.2 Segunda abordagem

```
:-use_module(library(clpfd)).
:-use_module(library(lists)).


book(1, 'Ensaio_Sobre_a_Cegueira',
    ↪                                1, 'Romance',    1995, 16).
book(2, 'Lusiadas',
    ↪                                     2, 'Classic',    1572,
    ↪ 13).
book(3, 'Os_Maias',
    ↪                                     3, 'Classic',    1888,
    ↪ 14).
book(4, 'A_Viagem_do_Elefante',
    ↪                                1, 'Romance',    2008, 15).
book(5, 'O_Evangelho_segundo_Jesus_Cristo',
    ↪           1, 'Romance',    1991, 15).
book(6, 'A_Caverna',
    ↪                                     1, 'Romance',    2000,
    ↪ 15).


author(1, 'Saramago').
author(2, 'Camoes').
author(3, 'Eca_de_Queiros').


listBooks(List) :-
        findall(Id,
                        book(Id, _, _, _, _, _),
                        List).

getHeightList([], _, _, _).
getHeightList([H | Rest], HeightList, I, IMax):-
        book(I, _, _, _, _, BookHeight),
        I1 is I + 1,
        element(H, HeightList, BookHeight),
        getHeightList(Rest, HeightList, I1, IMax).

sortByHeight([H1, H2 | HRest], [A1, A2 | ARest]):-
        (A1 #= A2 #=> H1 #=< H2),
        sortByHeight([H2 | HRest], [A2 | ARest]).
```

```
sortByHeight([_], [_]).

checkHeight([], _).
checkHeight([H|T], MaxHeight):-
        H #=< MaxHeight,
        checkHeight(T, MaxHeight).

getAuthorList([], _, _, _).
getAuthorList([A | Rest], AuthorList, I, IMax):-
        book(I, _, Author, _, _, _),
        I1 is I + 1,
        element(A, AuthorList, Author),
        getAuthorList(Rest, AuthorList, I1, IMax).

sortByAuthor([A1, A2 | ARest]):-
        A1 #=< A2,
        sortByAuthor([A2|ARest]).
sortByAuthor([_]).


getYearList([], _, _, _).
getYearList([Y | Rest], YearList, I, IMax):-
        book(I, _, _, _, Year, _),
        I1 is I + 1,
        element(Y, YearList, Year),
        getYearList(Rest, YearList, I1, IMax).

sortByYear([Y1, Y2 | YRest], [H1, H2 | HRest], [A1,A2 | ARest
    ↪ ]):-
        ((H1 #= H2 #/\ A1 #= A2) #=> Y1 #=< Y2),
        sortByYear([Y2|YRest], [H2 | HRest],[A2 | ARest]).
sortByYear([_], [_], [_]).



livros(BooksRes, NrShelves, WidthShelf, MaxHeight):-
        AllShelfsSize is NrShelves*WidthShelf,
        listBooks(Books),
        length(Books, Size),
        AllShelfsSize >= Size,
        length(BooksRes, Size),
        all_distinct(BooksRes),
        domain(BooksRes, 1, Size),
        length(HeightList, Size),
        length(AuthorList, Size),
        length(YearList, Size),
        getHeightList(BooksRes, HeightList, 1, Size),
        getAuthorList(BooksRes, AuthorList, 1, Size),
        getYearList(BooksRes, YearList, 1, Size),
        sortByAuthor(AuthorList),
```

```
        sortByHeight(HeightList, AuthorList),
        checkHeight(HeightList, MaxHeight),
        sortByYear(YearList, HeightList, AuthorList),
        reset_timer,
        labeling([time_out(60000,_)], BooksRes),
        printShelfs(BooksRes, 0, Size, Res),
        printResult(Res, 0, WidthShelf),
        nl,write('### STATISTICS ###'),nl,
        print_time,
    fd_statistics,nl,!.

printShelfs(_, C, C, []).
printShelfs(Books, C, Size, [LH | LT]):-
        C1 is C + 1,
        nth1(Index, Books, C1),
        book(Index, Name, Author, _,YearPub, Height),
        author(Author, AuthorName),
        LH = [Name, ' escrito por': AuthorName, ' em': YearPub
            ↪ , ' com altura de': Height],
        printShelfs(Books, C1, Size, LT).

printResult([],_,_):-
        write('
            ↪ ------------------------------------------------------------------------------
            ↪ n').
printResult([LH | LT], Counter, Max) :-
        Mod is Counter mod Max,
        Mod == 0,
        write('
            ↪ ------------------------------------------------------------------------------
            ↪ n'),
        NShelf is Counter div Max + 1,
        C is Counter + 1,
        write('Shelf '),
        write(NShelf), nl,
        write(LH), nl,
        printResult(LT, C, Max).

printResult([LH | LT], Counter, Max) :-
        C is Counter + 1,
        write(LH), nl,
        printResult(LT, C, Max).

reset_timer :- statistics(walltime,_).
print_time :-
        statistics(walltime,[_,T]),
        TS is ((T//10)*10)/1000,
        nl, write('Time: '), write(TS), write('s'), nl, nl.
```