UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Implementarea unui algoritm de tip greedy ce rezolva problema bancnotelor in Dafny

propusă de

Veronica Elisa Chicoș

Sesiunea: iunie, 2022

Coordonator științific

Conf. Dr. Ciobaca Stefan

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Implementarea unui algoritm de tip greedy ce rezolva problema bancnotelor in Dafny

Veronica Elisa Chicoș

Sesiunea: iunie, 2022

Coordonator științific

Conf. Dr. Ciobaca Stefan

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Ciobaca Stefan.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Chicoș Veronica Elisa** domiciliat în **România, jud. Galați, com. Matca, str. 1 decembrie 1918, nr. 7**, născut la data de **09 decembrie 2000**, identificat prin CNP **6001209171714**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2022, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Implementarea unui algoritm de tip greedy ce rezolva problema bancnotelor in Dafny** elaborată sub îndrumarea domnului **Conf. Dr. Ciobaca Stefan**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Implementarea unui algoritm de tip greedy ce rezolva problema bancnotelor in Dafny**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Veronica Elisa Chicoș**

Data: ..... Semnătura: .....

Cuprins

Motivație

Motivul alegerii acestei teme deriva din dorinta de a aprofunda subiectul optimizarii. In timpul facultatii, am invatat ca un cod trebuie sa fie optimizat pentru a creste eficineta acestuia. Drept urmare, consider ca orice actiune, cat de simpla, poate fi optimizata astfel incat sa se realizeze intr-un timp cat mai scurt si cu un numar minim de resurse.

Intentie

In cadrul lucrarii voi face o scurta intorducere in paradigma de programare Greedy, urmata de mai multe probleme care se rezolva cu ajutorul acesteia, printre care si Problema Bancnotelor.

Totodata, voi prezenta limbajul Dafny cat si implementarea algoritmului pentru rezolvarea problemei alese de mine.

Introducere

Lucrarea va fi structurata in felul urmator:

1. Paradigma Greedy - in acest capitol voi prezenta paradigma Greedy impreuna cu cateva probleme si aplicatiile acestora
2. Problema Bancnotelor - in acest capitol voi descrie problema pe care am ales-o pentru licenta
3. Limbajul de programare Dafny - in acest capitol voi prezenta o introducere in limbajul Dafny
4. Detalii de implementare - in acest capitol voi prezenta in detaliu algoritmul implementat

Chapter 1

Paradigma Greedy

Greedy este o strategie de rezolvare a problemelor de optimizare. Metoda presupune luarea unei decizii definitive la fiecare pas in functie de informatiile cunoscute in prezent fara a ne ingrijora de efectul acesteia in viitor.

1.1 Ce este o problema de optimizare si cum lucreaza metoda Greedy cu aceasta?

O problema de optimizare este o problema care are un input oarecare dar outputul ei trebuie sa fie o valoare maxima sau minima. In functie de ce ne cere problema, aceasta poate fi de doua feluri:

1. Problema de minimizare

Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply.

2. Problema de maximizare

Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply.

In functie de decizia facuta de metoda Greedy, vom crea solutii posibile. O solutie posibila este o submultime care satisface cerintele problemei, in timp ce o solutie optima respecta cerintele date si are cost minim sau castig maxim.

Dacă mai multe soluții îndeplinesc criteriile date, atunci acele soluții vor fi considerate ca fiind posibile, in timp ce solutia optima este cea mai buna dintre toate solutiile.

Drept urmare, aceasta metoda este folosita pentru a determina solutia oprima si a rezolva corect problema de optimizare.

1.2 Algoritmi Greedy

1.2.1 Problema selectiei activitatilor

Problema activitatilor este o problema de maximizare care presupune alegerea cat mai multor activitati care pot fi realizate intr-un anumit interval de timp fara ca acestea sa se suprapuna.

O posibila formulare a acestei probleme este:

Input:

- n - numarul de activitati,
- aStart[0..n-1] - vector care contine timpul la care incep activitatile,
- aSfarsit[0..n-1] - vector ce contine timpul la care se termina actvitatile astfel incat aStart[i] $<$ aSfarsit[i] pentru orice 0 $\leq$ i $\leq$ n-1

Output:

- A $\subseteq$ { 0, ..., n-1 }, unde A este o multime de activitati care nu se suprapun si este de cardinal maxim;

Spre exmplu putem avea aceste multimi de activitati reprezentate in tabelul de mai jos:

Nr	Activitate	Inceput	Final
1	Curs ML	8	10
2	Seminar AI	10	12
3	Curs Pian	11	1
4	Antrenament Inot	12	17
5	Curs CN	16	18
6	Consultatii RPA	17	19
7	Laborator CN	18	20

Solutia optima este $A = \{1, 2, 4, 6\}$

1.2.2 Problema Codurilor Huffman

1.2.3 Problema Bin-packing

1.2.4 Algoritmul Dijkstra

1.2.5 Problema arborelui partial de cost minim

1.3 Avantaje si dezavantaje ale algoritmilor Greedy

Chapter 2

Problema Bancnotelor

2.1 Ce este Problema Bancnotelor?

Problema bancnotelor este o problema de minimizare de care ne lovim zilnic. Stiind ca avem o multime de bancnote $B = \{b_1, b_2, b_3\}$ si o suma S de platit, rezultatul problemei va fi multimea de bancnote cu cardinal minim pe care o putem folosi pentru a plati suma S.

2.1.1 Formularea problemei

In limbaj natural, Problema Bancnotelor poate fi formulata astfel:

Se da o multime de bancnote $B = \{b_1, b_2, b_3\}$ si o suma S pe care trebuie sa o platim. Se cere sa se afiseze o multime de bancnote care trebuie sa indeplineasca urmatoarele conditii:

- cardinalul multimii sa fie minim,
- suma elementelor multimii sa fie egata cu S;

Formularea computationala a problemei:

Input:

- un numar natural n - suma care trebuie platita,

Output:

- numerele $n_{500}, n_{200}, n_{100}, n_{50}, n_{20}, n_{10}, n_5, n_1$ (n_i - numarul de bancnote i folosite), astfel incat $\sum_{i \in \{500,200,100,50,20,10,5,1\}} n_i$ sa fie minima si $n = \sum_{i \in \{500,200,100,50,20,10,5,1\}} i \times n_i$.

Spre exemplu, fiind date bancnotele $B = \{100, 50, 10, 5, 1\}$ si $s = 157$ suma care trebuie platita.

Solutia optima pentru aceasta formulare a problemei este $\{1, 1, 0, 1, 2\}$ deoarece $s = 1 \cdot 100 + 1 \cdot 50 + 0 \cdot 10, 1 \cdot 5 + 1 \cdot 2$.

2.2 Strategia Greedy aleasa

Strategia Greedy consta in alegerea, la fiecare pas, a bancnotei celei mai mari, care este mai mica sau egala cu suma pe care trebuie sa o platim. In functie de alegerea facuta, va trebui sa scadem valoarea bancnotei alese din suma care trebuie platita pentru a ne asigura ca avansam progresiv catre obiectivul final.

Folosind exemplul de mai sus, prima decizie facuta de strategia noastra este alegerea bancnotei de 100 deoarece $100 \leq 157$. Decizia facuta are ca rezultat adaugarea valorii 1 la solutia finala iar suma pe care trebuie sa o platim scade cu 100, astfel incat la pasul urmator va trebui sa o platim 57. Acest algoritm se va repeta pana cand suma va fi 0.

// exemplu desen pasi

2.3 Limitarea Strategiei Greedy

Limitarea strategiei alese este ca exista posibilitatea ca acesta sa nu ofere solutia optima pentru toate datele de intrare.

De exemplu, daca multimea de bancnote este formata din $B = \{1, 6, 9\}$ iar suma care trebuie platita este $s = 12$, solutia optima generata de strategia aleasa va fi $\{3, 0, 1\}$. Cu toate acestea, solutia cu adevarat optima va fi $\{0, 2, 0\}$.
exemplu

Motivul pentru care se produce aceasta "eroare" este ca solutia este construita pas cu pas. La fiecare pas, se alege bancnota care va crea cel mai mic cost pentru solutia curenta chiar daca in viitor ar exista o alta solutie cu cost mai mic.

Chapter 3

Dafny

3.1 Limbajul de programare Dafny

Dafny este un limbaj de programare functional si imperativ care prin intermediul adnotarilor sale permite crearea unui program care sa nu contina erori la runtime si sa faca ceea ce isi doreste programatorul

Un exemplu de adnotare este $requires\ a >= 10$. Astfel, suntem siguri ca variabila a nu va fi mai mica decat 10 scutindu-ne de o structura de decizie pe care ar fi trebui sa o utilizam pentru a verifica acest lucru. Alte erori care pot fi evitate prin folosirea adnitarilor sunt erorile de indexare, impartirea la 0, valori nule etc.

3.2 Avantajele folosirii limbajului Dafny

Preconditiile si postconditiile din acest limbaj de programare stabilesc ce conditii trebuie sa fie adevarate la intrare, respectiv, la iesirea din metoda. Astfel, programele in Dafny sunt verificate pentru corectitudinea globala astfel incat fiecare rulare se va termina si se va ajunge la rezultatul dorit. In plus,adaugarea acestor conditii duc la o intelegere in detaliu a codului.

Chapter 4

Detalii de implementare

In acest capitol voi prezenta detaliile de implementare a algoritmului cat si modul in care am demonstrat corectitudinea codului utilizand Dafny.

4.1 Reprezentarea datelor de intrare si a celor de iesire

4.1.1 Varibilele folosite, tipurile si semnificatia acestora

- suma : $int \rightarrow$ reprezinta suma pe care trebuie sa o platim
- rest : $int \rightarrow$ reprezinta suma care ne-a ramas de platit dupa ce am ales o bancnota
- solutieFinala : $seq < int > \rightarrow$ reprezinta solutia finala a problemei
- solutieCurenta : $seq < int > \rightarrow$ reprezinta solutia creata pana la pasul curent
- solutieOarecare : $seq < int > \rightarrow$ reprezinta o solutie pe care am folosit-o pentru a demonstra corectitudinea programului

4.1.2 Descrierea solutiei rezultate

Bancnotele folosite de mine pentru rezolvarea acestei probleme sunt $B = \{1, 5, 10, 20, 50\}$, astfel, solutia problemei va fi o secventa de numere naturale $solutie = \{n_1, n_5, n_{10}, n_{20}, n_{50}\}$ unde n_i va numarul de i bancnote folosite iar

$$n_1 \cdot 1 + n_5 \cdot 5 + n_{10} \cdot 10 + n_{20} \cdot 20 + n_{50} \cdot 50 == suma.$$

4.2 Implementarea algoritmului Greedy

4.2.1 Predicate si functii

Inainte de a trece la implementarea propriu-zisa a algoritmului, o sa incep prin a prezenta predicatele care asigura corectitudinea rezultatului intors de catre metoda ce genereaza solutia optima pentru suma pe care trebuie sa o platim.

Predicatele sunt metode care returneaza o valoare de adevar si sunt folosite pentru a verifica o anumita proprietate sau mai multe prin intermediul unei singure instructiuni.

In Dafny, functiile sunt asemanatoare cu functiile matematice. Functiile contin o singura expresie care returneaza o valoare de tipul declarat in antetul functiei.

1. Predicatul $esteSolutieValida(solutie : seq < int >) \rightarrow$ acest predicat ne asigura ca solutia primita ca parametru este de lungime 5 si ca fiecare element este mai mare sau egal cu 0. Conditii mentionate anterior sunt foarte importante deoarece, avand 5 bancnote (1, 5, 10, 20, 50), inseamna ca va trebuie sa existe un elemnt al solutiei pentru fiecare dintre ele.

```
predicate esteSolutieValida(solutie : seq<int>)
{
  |solutie| == 5 && solutie[0] >= 0 && solutie[1] >= 0 &&
     solutie[2] >= 0 && solutie[3] >=0 && solutie[4] >=0
}
```

2. Predicatul $esteSolutie(solutie : seq < int >, suma : int) \rightarrow$ acest predicat verifica faptul ca suma elementelor secventei $solutie$ inmultime cu bancnotele folosite in problema este egala cu $suma$. Daca cele doua sunt egale, atunci secventa $solutie$ devine solutie posibila pentru $suma$.

```
predicate esteSolutie(solutie : seq<int>, suma : int)
  requires esteSolutieValida(solutie)
{
  solutie[0] * 1 + solutie[1] * 5 + solutie[2] * 10 +
     solutie[3] * 20 + solutie[4] * 50 == suma
}
```

3. Functia $cost(solutie : seq < int >) : int \rightarrow$ aceasta functie calculeaza si returneaza cate bancnote au fost folosite pentru a forma solutia posibila memorate in variabila $solutie$ data ca parametru.

```
function cost(solutie : seq<int>) : int
  requires esteSolutieValida(solutie)
{
  solutie[0] + solutie[1] + solutie[2] + solutie[3] +
     solutie[4]
}
```

4. Predicatul $esteSolutieOptima(solutie : seq < int >, suma : int) \rightarrow$ acest predicat va returnat valoarea de adevar daca solutia primita ca parametru este o solutie posibila pentru suma pe care trebuie sa o platim si orice alta solutie posibila are costul mai mare sau egal. Acest lucru se intampla deoarece Problema Bancnotelor este o problema de minimizare iar solutia trebuie sa aibe costul minim.

```
predicate esteSolutieOptima(solutie : seq<int>, suma : int)
  requires esteSolutieValida(solutie)
{
  esteSolutie(solutie, suma) &&
  forall solutieOarecare ::
     esteSolutieValida(solutieOarecare) &&
     esteSolutie(solutieOarecare, suma)
  ==> cost(solutieOarecare) >= cost(solutie)
}
```

5. Predicatul $INV(rest : int, suma : int, solutieFinala : seq < int >) \rightarrow$ acest predicat va fi adevarat daca pentru orice solutie posibila pentru $rest$, (adica pentru suma care ne-a ramas de platit),

```
predicate INV(rest : int, suma : int, solutieFinala :
   seq<int>)
  requires esteSolutieValida(solutieFinala)
{
  forall solutieCurenta :: esteSolutieValida(solutieCurenta)
     ==>
```

```
        (esteSolutie(solutieCurenta, rest) ==>
        esteSolutie([solutieFinala[0] + solutieCurenta[0],
           solutieFinala[1] + solutieCurenta[1],
        solutieFinala[2] + solutieCurenta[2], solutieFinala[3] +
           solutieCurenta[3], solutieFinala[4] +
           solutieCurenta[4]], suma)) &&
        (esteSolutieOptima(solutieCurenta, rest) ==>
        esteSolutieOptima([solutieFinala[0] + solutieCurenta[0],
           solutieFinala[1] + solutieCurenta[1],
        solutieFinala[2] + solutieCurenta[2], solutieFinala[3] +
           solutieCurenta[3], solutieFinala[4] +
           solutieCurenta[4]], suma))
      }
```

4.2.2 Algoritmul Greedy

```
method nrMinimBancnote(suma : int) returns (solutie : seq<int>)
  requires suma >= 0
  ensures esteSolutieValida(solutie)
  ensures esteSolutie(solutie, suma)
  ensures esteSolutieOptima(solutie, suma)
{
 var rest := suma;
 var s1 := 0;
 var s5 := 0;
 var s10 := 0;
 var s20 := 0;
 var s50 := 0;
 while(rest > 0)
  decreases rest
  invariant 0 <= rest <= suma
  invariant esteSolutie([s1, s5, s10, s20, s50], suma - rest)
  invariant INV(rest, suma, [s1, s5, s10, s20, s50])
 {
  var i := 0;
```

```
var s := gasireMaxim(rest);
if( s == 1)
{
  cazMaxim1(rest, suma, [s1, s5, s10, s20, s50]);
  s1 := s1 + 1;
  assert esteSolutie([s1, s5, s10, s20, s50], suma - (rest - 1));
  assert INV(rest - 1, suma, [s1, s5, s10, s20, s50]);
}
else if(s == 5)
{
  cazMaxim5(rest, suma, [s1, s5, s10, s20, s50]);
  s5 := s5 + 1;
  assert esteSolutie([s1, s5, s10, s20, s50], suma - (rest - 5));
  assert INV(rest - 5, suma, [s1, s5, s10, s20, s50]);

}
else if (s == 10)
{
  cazMaxim10(rest, suma, [s1, s5, s10, s20, s50]);
  s10 := s10 + 1;
  assert esteSolutie([s1, s5, s10, s20, s50],
       suma - (rest - 10));
  assert INV(rest - 10, suma, [s1, s5, s10, s20, s50]);
}
else if(s == 20)
{
  cazMaxim20(rest, suma, [s1, s5, s10, s20, s50]);
  s20 := s20 + 1;
  assert esteSolutie([s1, s5, s10, s20, s50], suma - (rest - 20));
   assert INV(rest - 20, suma, [s1, s5, s10, s20, s50]);
}
else
{
  cazMaxim50(rest, suma, [s1, s5, s10, s20, s50]);
  s50 := s50 + 1;
  assert esteSolutie([s1, s5, s10, s20, s50], suma - (rest - 50));
```

```
    assert INV(rest - 50, suma, [s1, s5, s10, s20, s50]);
  }
   rest := rest - s;
  }
  solutie := [s1, s5, s10, s20, s50];
}
```

Algoritmul propus de mine primeste ca parametru suma pe care va trebui sa o platim si returneaza o solutie care respecta proprietatile discutate mai sus in sectiunea 4.1.2 *Descrierea solutiei rezultate*.

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

1. https://www.jeffyang.io/blog/min_number_of_coins_for_exchange/
2. https://www.baeldung.com/cs/min-number-of-coins-algorithm
3. https://jeffe.cs.illinois.edu/teaching/algorithms/book/04-greedy.pdf
4. https://www.guru99.com/greedy-algorithm.html
5. https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYX Vsd GRvbWFpbnxmaWljb3Vyc2VwYXxneDo2ZTIzNDNhOTZmZmFmZjdk
6. https://brilliant.org/wiki/greedy-algorithm/
7. https://techvidvan.com/tutorials/greedy-algorithm/
8. https://www.hackerearth.com/practice/algorithms/greedy/basics-of -greedy-algorithms/tutorial/
9. https://progressivecoder.com/coin-change-problem-using-greedy-algorithm/
10. https://dafny.org/dafny/OnlineTutorial/guide
11. https://dafny.org/dafny/QuickReference
12. http://www.doc.ic.ac.uk/ scd/Dafny_Material/Lectures.pdf
13. https://www.microsoft.com/en-us/research/project/dafny-a-language-and-program-verifier-for-functional-correctness/
14. https://arxiv.org/pdf/1701.04481.pdf