Universitatea din Craiova
Facultatea de Automatică, Calculatoare și Electronică

Lanul de porumb

Popa Mihai Radu
Calculatoare Română
Anul I
Grupa 1.2B

Mai 2020

1 Enunțul problemei

Într-un lan de porumb,cantitatea de porumb ce va putea fi recoltată depinde de distanțele dintre fiecare doi coceni consecutivi aflați pe același rand. Dacă această valoare este mai mică decât o limită minimă prestabilită d atunci recolta nu va mai fi optimă deoarece porumbii nu se vor mai putea dezvolta corespunzător. Din acest motiv un agricultor trebuie să elimine o parte din porumbii de pe același rând astfel încat distanța dintre oricare doi coceni succesivi să fie cel puțin d. Se consideră că porumbii au fost plantați pe un singur rând. Totodată, agricultorul va trebui să elimine cât mai puțini coceni deoarece prin eliminarea unui cocean producția se va micșora. Se cunosc: i) distanța minimă care trebuie păstrată între oricare doi coceni consecutivi; ii) coordonatele cocenilor plantați pe același rând. Să se determine: i) numărul minim de tulpini care trebuie eliminate astfel încat să se respecte distanța minimă între doi coceni consecutivi; ii) coordonatele cocenilor care trebuie păstrați.

2 Algoritmi

ELIMINARE-TULPINI(VECTOR, DIMENSIUNE, DISTANTA)

```
1:  for i ← 0, dimensiune − 1 do
2:      if vector[i] = 1 then
3:          for j ← i + 1, i + distanta − 1 do
4:              if vector[j] = 1 then
5:                  eliminari ← eliminari + 1
6:              vector[j] ← 0
7:          i ← i + distanta − 1
8:          tulpini_ramase ← tulpini_ramase + 1
9:  if vector[dimensiune] = 1 then
10:     tulpini_ramase ← tulpini_ramase + 1
```

ELIMINARE-TULPINI(VECTOR, DIMENSIUNE, DISTANTA)

```
1:  eliminari ← 0
2:  tulpini_ramase ← 0
3:  for i ← 0, dimensiune − 1 do
4:      if vector[i + 1] − vector[i] < distanta then
5:          for j ← i + 1, dimensiune − 1 do
6:              vector[j] ← vector[j + 1]
7:          i ← i − 1
8:          dimensiune ← dimensiune − 1
9:          eliminari ← eliminari + 1
10: aux ← dimensiune
11: while eliminari <= nr_coordonate do
12:     tulpini_ramase ← tulpini_ramase + 1
13:     eliminari ← eliminari + 1
```

2.1 Algoritmul 1

2.1.1 Descrierea variabilelor

Am folosit contorii i și j pentru a parcurge tabloul unidimensional. Variabila dimensiune reține dimensiunea tabloului unidimensional, variabila tulpini_ramase reține numărul de tulpini rămase, variabila distanta reține distanța necesară dezvoltării corespunzătoare dintre doi coceni de porumb consecutivi, iar variabila eliminari reține numărul de tulpini eliminate.

2.1.2 Descrierea algoritmului

Algoritmul parcurge cu ajutorul contorului i tabloul unidimensional (lan) completat cu 1 și 0, unde 1 reprezintă existența unei tulpini de porumb, iar 0 lipsa acesteia. Daca pe poziția i se află o tulpină, parcurgem cu ajutorul contorului j tabloul unidimensional până la următoarea tulpină de porumb care respectă distanța minimă necesară dezvoltării corespunzătoare a porumbilor. Daca pe poziția j se află o tulpină, incrementăm numărul de tulpini eliminate. Apoi eliminăm tulpinile parcurse (marcarea cu 0 în tabloul unidimensional). Sărim peste tulpinile parcurse și incrementăm numărul de tulpini rămase cu o unitate. Dacă pe ultima poziție a tabloului unidimensional se află o tulpină de porumb, incrementăm numărul de tulpini rămase cu o unitate.

Linie	Cost	Timp de execuție	Notație asimptotică
1			
2	c1	$\sum_{i=0}^{n-1} c1$	$O(n)$
3			
4	c2	$\sum_{i=0}^{n-1} \sum_{j=i+1}^{i+n-1} c2$	$O(n^2)$
5	c3	$\sum_{i=0}^{n-1} \sum_{j=i+1}^{i+n-1} c3$	$O(n^2)$
6	c4	$\sum_{i=0}^{n-1} \sum_{j=i+1}^{i+n-1} c4$	$O(n^2)$
7	c5	$\sum_{i=0}^{n-1} c5$	$O(n)$
8	c6	$\sum_{i=0}^{n-1} c6$	$O(n)$
9	c7	c7	$O(1)$
10	c8	c8	$O(1)$

$T(n) = O(n) + O(n^2) + O(n^2) + O(n^2) + O(n) + O(n) + O(1) + O(1)$
$\implies T(n) = O(n^2)$

Din punct de vedere al timpului de execuție algoritmul are complexitatea $O(n^2)$, iar din punct de vedere al memoriei are complexitatea $O(n)$ sau $O(1)$.

2.2 Algoritmul 2

2.2.1 Descrierea variabilelor

Ceea ce apare în plus față de primul algoritm sunt cele două variabile: aux în care reținem dimensiunea tabloului unidimensional după eliminarea tulpinilor și nr_coordonate în care reținem numărul de coordonate.

2.2.2 Descrierea algoritmului

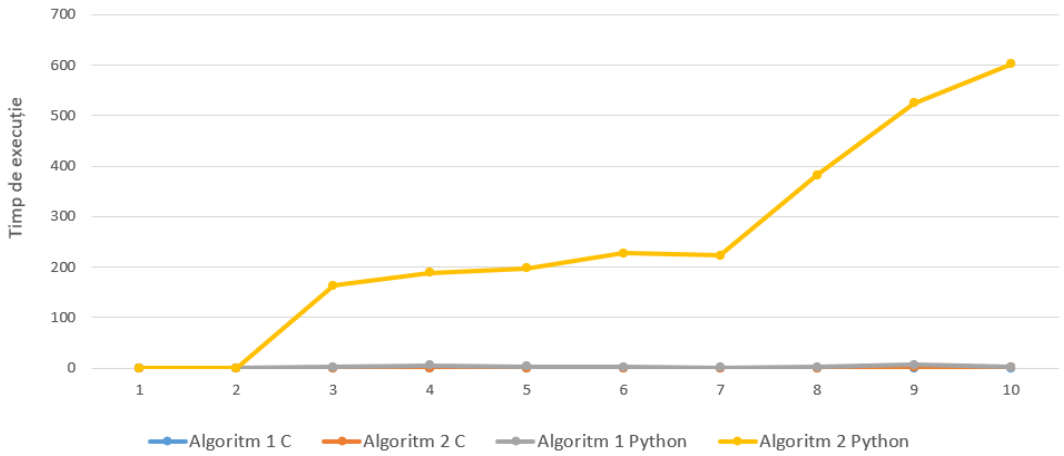
Algoritmul parcurge cu ajutorul contorului i tabloul unidimensional (coordonate) sortat strict crescător(cu ajutorul algoritmului quicksort), care stochează coordonatele tulpinilor de porumb. Dacă diferența dintre două coordonate alăturate este strict mai mică decât distanța necesară dezvoltării corespunzătoare a porumbilor se elimină tulpina de pe poziția i+1, deplasând tabloul unidimensional cu o poziție spre stânga. Decrementăm cu o unitate valoarea contorului i pentru a rămâne la poziția curentă. Incrementăm numărul de eliminări cu o unitate și decrementăm dimensiunea tabloului unidimensional cu o unitate deoarece am eliminat o tulpină. Reținem în variabila auxiliară aux dimensiunea tabloului unidimensional după eliminarea tulpinilor. Cu ajutorul instrucțiunii while determinăm numărul de tulpini rămase.

Din punct de vedere al timpului de execuție algoritmul de sortare quicksort are complexitatea $O(nlogn)$, iar din punct de vedere al memoriei are complexitatea $O(n)$.

Linie	Cost	Timp de execuție	Notație asimptotică
1	c1	c1	$O(1)$
2	c2	c2	$O(1)$
3			
4	c3	$\sum_{i=0}^{n-1} c3$	$O(n)$
5			
6	c4	$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} c4$	$O(n^2)$
7	c5	$\sum_{i=0}^{n-1} c5$	$O(n)$
8	c6	$\sum_{i=0}^{n-1} c6$	$O(n)$
9	c7	$\sum_{i=0}^{n-1} c7$	$O(n)$
10	c8	c8	$O(1)$
11			
12	c9	$\sum_{eliminari=0}^{nr\_coordonate} c9$	$O(n)$
13	c10	$\sum_{eliminari=0}^{nr\_coordonate} c10$	$O(n)$

$T(n) = O(1) + O(1) + O(n) + O(n^2) + O(n) + O(n) + O(n) + O(1) + O(n) + O(n)$
$\implies T(n) = O(n^2)$

Din punct de vedere al timpului de execuție algoritmul are complexitatea $O(n^2)$, iar din punct de vedere al memoriei are complexitatea $O(n)$.



Exemple de rulaj:

Input
10428 3430 3648 1159 1706 6936 9796 3112 3101 ... ... 4585 1042 399 8382 1321 6526 9085

Output
Numarul minim de tulpini eliminate: 10424 Coordonatele cocenilor ramasi sunt: 0 3430 6860 10290 Durata testului: 0.1429999999999s

Input
5 77 60 40 13 34 3

Output
IMPOSIBIL Durata testului: 0.0000000000000s

3 Date experimentale

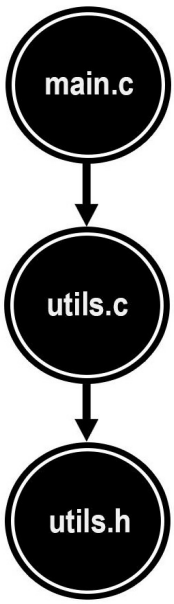
Pentru datele experimentale am folosit un algoritm care generează aleatoriu zece seturi de date de intrare, fiecare set având pe prima linie două numere ce reprezintă *numărul de coordonate* și *distanța minimă* care trebuie păstrată între oricare doi coceni consecutivi, iar pe liniile următoare *nr* numere ce reprezintă *coordonatele* cocenilor de porumb. Parametrul *n* reprezintă limita maximă până la care pot fi generate aleator valori.

GENERATOR(N)

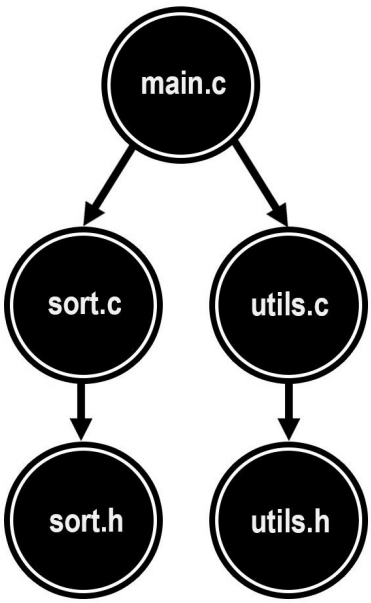
```
1: for i ← 1, 10 do
2:     nr ← random(1, n)
3:     distanta_minima ← random(1, n) + 1
4:     print(nr, distanta_minima)
5:     for j ← 1, nr do
6:         print(random(1, n))
```

4 Proiectarea aplicației experimentale

4.1 Structura de nivel înalt a aplicației



Aplicația 1 - Structura pe nivel înalt



Aplicația 2 - Structura pe nivel înalt

4.2 Descrierea mulțimii datelor de intrare

Programul citește dintr-un fișier text nr_coordonate (numărul de coordonate), distanta_minima (distanța care trebuie păstrată între oricare doi coceni consecutivi) și cele nr_coordonate (coordonatele cocenilor de porumb).

Input
62 58 16 40 54 51 84 61 34 71 53 15 85 0 20 86 80 24 61 34 89 9 69 86 94 46 40 34 6 66 3 45 45 7 32 48 49 78 65 43 88 96 6 25 10 48 66 29 26 4 43 8 93 53 40 36 50 51 76 75 62 72 65 49

4.3 Descrierea ieșirilor/rezultatelor

Programul afișează într-un fișier text numărul minim de tulpini eliminate, coordonatele cocenilor de porumb rămași și timpul de execuție.

Output
Numarul minim de tulpini eliminate: 60 Coordonatele cocenilor ramasi sunt: 0 61 Durata testului: 0.0000000000000s

4.4 Modulele aplicației

Aplicația 1 conține următoarele module:

- ***main.c*** - utilizat pentru interfața utilizatorului
- ***utils.c***, ***utils.h*** - utilizat pentru funcțiile de generare, afișare și eliminare

Aplicația 2 conține următoarele module:

- ***main.c*** - utilizat pentru interfața utilizatorului
- ***sort.c***, ***sort.h*** - utilizat pentru funcția de sortare și funcțiile auxiliare
- ***utils.c***, ***utils.h*** - utilizat pentru funcțiile de generare, afișare și eliminare

4.5 Funcțiile aplicației

Aplicația 1

- ***utils.c*** conține următoarele funcții:

    1. ***generator*** - generează aleatoriu datele de intrare
    Funcția conține un singur parametru *n* ce reprezintă limita maximă până la care pot fi generate aleator valori.
    2. ***afisare_coordonate*** - afișează coordonatele cocenilor de porumb rămași
    Funcția conține doi parametri: *vector* și *dimensiune*. Parametrul *vector* este o mulțime de valori care reprezintă lanul de porumb, iar parametrul *dimensiune* reprezintă numărul de elemente ale mulțimii.
    3. ***eliminare_tulpini*** - elimină tulpinile de porumb care nu respectă distanța minimă
    Funcția conține trei parametri: *vector*, *dimensiune*, *distanta*. Parametrul *vector* este o mulțime de valori care reprezintă lanul de porumb, parametrul *dimensiune* reprezintă numărul de elemente ale mulțimii, iar parametrul *distanta* reprezintă distanța minimă ce trebuie păstrată între doi coceni consecutivi.

Aplicația 2

- ***sort.c*** conține următoarele funcții:

    1. ***interchange_values*** - interschimbă două elemente
    Funcția conține doi parametri: *address_1* si *address_2*. Parametrul *address_1* este un pointer la adresa primului element, iar parametrul *address_2* este un pointer la adresa celui de-al doilea element.
    2. ***partition*** - ia ultimul element ca pivot, plasează elementul pivot pe poziția sa corectă în vectorul sortat, toate elementele mai mici decât pivotul la stânga pivotului, iar toate elementele mai mari la dreapta pivotului
    Funcția conține trei parametri: *vector*, *left* și *right*. Parametrul *vector* reprezintă un pointer la mulțimea de valori care urmează să fie sortată, parametrul *left* reprezintă indexul de pornire, iar parametrul *right* reprezintă imdexul de oprire.
    3. ***quick_sort*** - funcția principală ce implementează quicksort
    Funcția conține trei parametri: *vector*, *left* și *right*. Parametrul *vector* reprezintă un pointer la mulțimea de valori care urmează să fie sortată, parametrul *left* reprezintă indexul de pornire, iar parametrul *right* reprezintă imdexul de oprire.

- ***utils.c*** conține următoarele funcții:

    1. ***generator*** - generează aleatoriu datele de intrare
    Funcția conține un singur parametru *n* ce reprezintă numărul maxim până la care pot fi generate aleator valori.
    2. ***afisare_coordonate*** - afișează coordonatele cocenilor de porumb rămași
    Funcția conține doi parametri: *vector* și *dimensiune*. Parametrul *vector* este o mulțime de valori care reprezintă coordonatele cocenilor de porumb, iar parametrul *dimensiune* reprezintă numărul de elemente ale mulțimii.
    3. ***eliminare_tulpini*** - elimină tulpinile de porumb care nu respectă distanța minimă
    Funcția conține trei parametri: *vector*, *dimensiune*, *distanta*. Parametrul *vector* este o mulțime de valori care reprezintă coordonatele cocenilor de porumb, parametrul *dimensiune* reprezintă numărul de elemente ale mulțimii, iar parametrul *distanta* reprezintă distanța minimă ce trebuie păstrată între doi coceni consecutivi.

5 Rezultate & concluzii

5.1 Rezultate

Mai jos se poate observa o analiză comparativă a timpilor de execuție obținuți în urma rulării celor doi algoritmi implementați. Se observă faptul că algoritmul 1 este mai eficient decât algoritmul 2. Limbajul C este mult mai rapid decât limbajul Python, deoarece Python este un limbaj interpretat, iar C este un limbaj compilat.

Nr. coordonate	Algoritm 1 C	Algoritm 2 C	Algoritm 1 Python	Algoritm 2 Python
5	0.00000	0.00000	0.00000	0.00000
62	0.00000	0.00000	0.10935	0.06290
6180	0.00000	0.03100	1.79851	161.13638
7868	0.00000	0.06300	5.38579	183.15937
8562	0.01600	0.06900	3.30632	194.37593
9982	0.00000	0.09800	1.60569	225.48246
10428	0.00000	0.09300	1.26585	222.27484
17008	0.01500	0.25700	1.97549	379.91833
24394	0.00000	0.56900	5.76400	518.74625
27699	0.00000	0.71900	1.63756	599.63924

5.2 Concluzii

Când am început să scriu codul pentru acest proiect , obiectivul meu principal a fost să-l scriu cât de simplu se poate. Aceasta a fost probabil cea mai provocatoare parte a proiectului. După terminarea acestuia pot spune că am învățat că uneori este mai important cum găsești rezultatul decât rezultatul în sine. Acest proiect a fost foarte interesant pentru mine. Ca direcții viitoare, voi încerca să reduc numărul de caractere pe care le oferă programul, astfel încât acesta să poată rula mai repede.

Referinţe

[1] Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms*. MIT Press, 3rd Edition, 2009.

[2] w3schools – `https://www.w3schools.com/python/`, accesat în Aprilie 2020.

[3] overleaf – `https://www.overleaf.com/learn/latex/Main_Page`, accesat în Aprilie 2020.

[4] wikibooks – `https://en.wikibooks.org/wiki/LaTeX/Algorithms`, accesat în Aprilie 2020.

[5] stackexchange – `https://tex.stackexchange.com/`, accesat în Aprilie 2020