# Compresie de imagine folosind arbori quad liniari

## Comparație pentru diferite structuri de date

Ghiurțu Andrei  Olaru Karina

Facultatea de Matematică-Informatică, Universitatea Transilvania, Brașov

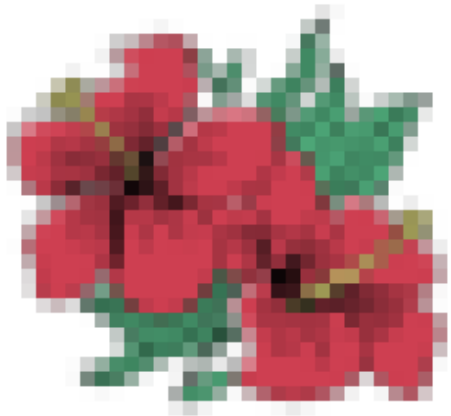
24 iunie 2022

# Cuprins

# Cuprins

# Introducere

Pentru procesarea de imagine, cel mai des, se folosesc arbori quad. În prezentarea ce urmează va fi expus un algoritm de compresie de imagine, construit cu ajutorul unui arbore quad liniar.



# Cuprins

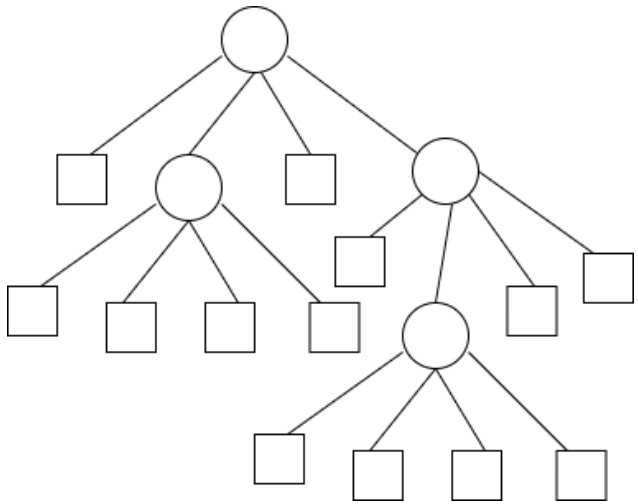
# Compresia de imagine

Compresia de imagine este o formă de reducere a dimensiunii unei imagini, fără a pierde detalii semnificative. Acest lucru este necesar pentru un transfer online, cât mai rapid, de fișiere.
Un exemplu de compresie prin **uniform quantization**:


# Cuprins

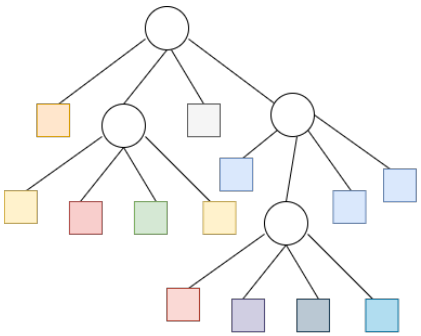
# Arbori quad liniari

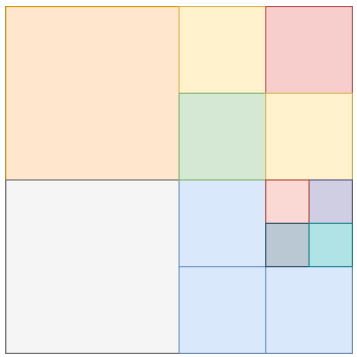
Arbore quad = structură de date arborescentă în care fiecare nod are 0 sau 4 copii
Pentru un arbore quad liniar se reține doar frunzele, deci este mai eficient.

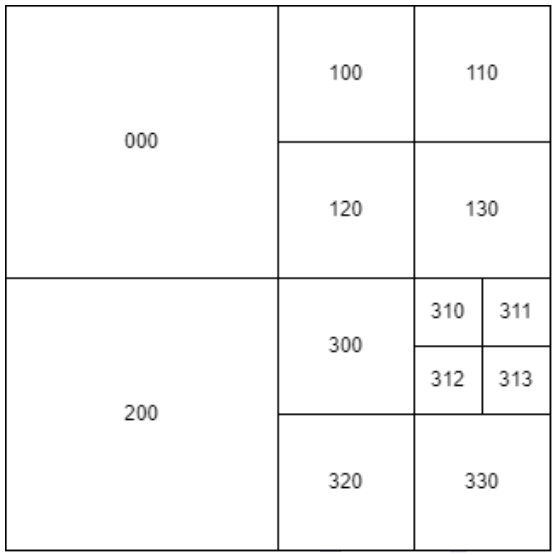


# Arbori quad liniari

Un arbore quad liniar se construiește pe baza unei matrice pătratice de pixeli care este divizată recursiv, până se ajuge ca fiecare nod să aibă o singură culoare.
Fiecărui nod îi este asociat un cod cu ajutorul căruia se identifică poziția sa în matrice.



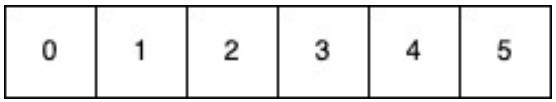




# Cuprins

# Structuri de date

Structurile de date alese pentru compresia de imagine sunt **lista dublu înlănțuită** și **vectorul din STL**.
Avantajul adus de utilizarea unei liste dublu înlănțuite este inserția și ștergerea elementelor în timp constant, iar avantajul unui vector este parcurgerea rapidă (datorată memoriei contigue).




*Reprezentarea unui vector*


*Reprezentarea unei liste dublu înlănțuite*

# Cuprins

# Divizarea

Se verifică fiecare pixel RGB din matrice și se efectuează divizarea, nodurile frunză fiind reținute într-o listă, care arată astfel:


*Lista de noduri frunză*

# Construire și divizare

**Algoritm 1:** Construire

**Intrare:** Matrice de pixeli
**Iesire:** Noduri frunză

```
construieste(matricePixeli):
  marime <- log2 (matricePixeli
    .marime)
  daca pixelMatrix nu e
    patratica sau marimea nu e
    numar natural atunci
     return
  sfarsit daca

  frunze.adauga(nodul format
    din toata matricea)
  divide(leafNodes.first)
```

**Algoritm 2:** Divizare

**Intrare:** Matrice de pixeli
**Iesire:** Noduri frunză

```
divide(parinte):
  daca nu trebuie divizat
    atunci
     oprire
  sfarsit daca

  construieste copii folosind
    parinte
  sterge parinte din frunze

  pentru copil in copii executa
    frunze.adauga(copil)
    divide(copil)
  sfarsit pentru
```

# Creare frunze imagine compresată

**Algoritm 4:** Îmbinare noduri

**Intrare:** Noduri de îmbinat
**Iesire:** Nod îmbinat

```
imbinareNoduri(noduri):
creare nod de nivel mai mic si
    marime dubla fata de nod
setare coordonate de inceput cu
     cele ale ultimului nod din
    vectorul noduri
setare culoare cu media dintre
    culorile nodurilor
```

**Algoritm 5:** Creează frunze folosite în compresie

**Intrare:** Frunze inițiale
**Iesire:** Frunze compresate

```
creareFrunzeCompresate(noduri,
    niveluriDeSters):
  pentru frunza in frunze
   executa
    daca nivelui frunzei <
   niveleDeSters atunci
      continua
    sfarsit daca
    frunzeNoi.adauga(frunza)
    cat timp pe nivelActual
   sunt patru frunze executa
      nodImbinat = imbina
   ultimele 4 noduri
      sterge ultimele 4 noduri
   din frunzeNoi
      frunzeNoi.adauga(
   nodImbinat)
    sfarsit cat timp
  sfarsit pentru
```

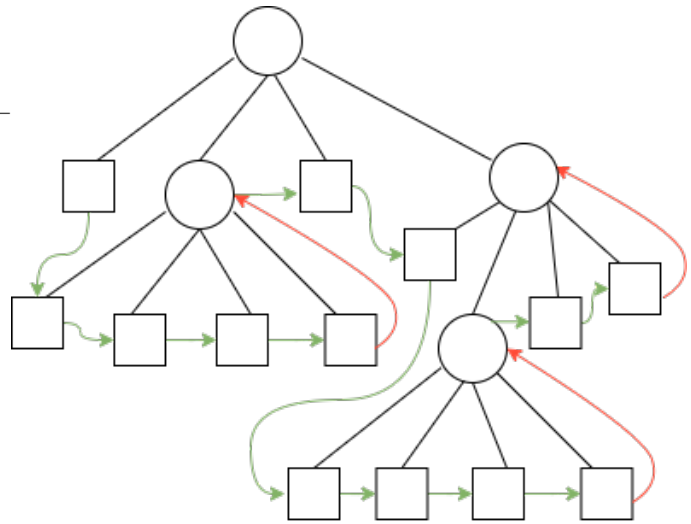


Figura: Mecanismul de îmbinare al nodurilor

# Cuprins

# Compararea performanțelor

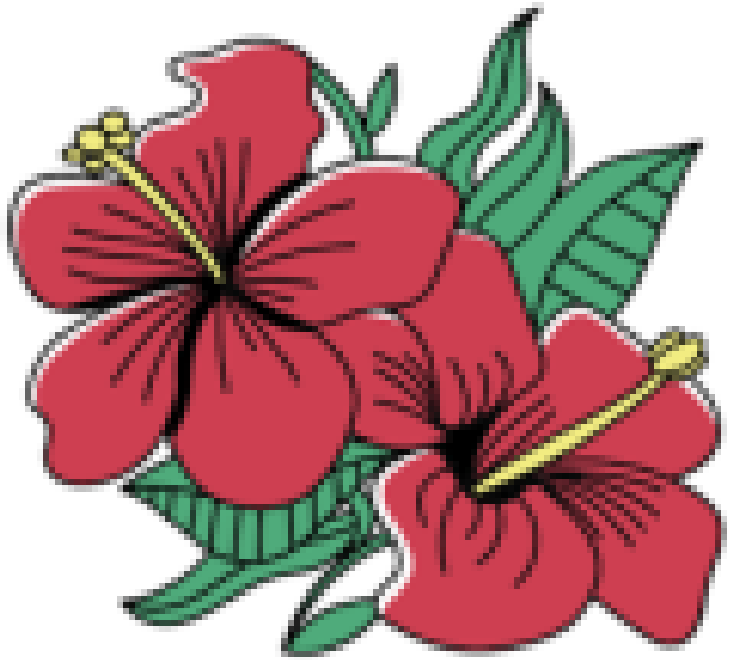
| Combinația de structuri de date | Divizare(s) | Îmbinare(s) |
|---|---|---|
| Listă-Vector | 60.9 | 0.273 |
| Listă-Listă | 63.9 | 0.332 |
| Vector-Listă | 237 | 0.345 |
| Vector-Vector | 242 | 0.281 |

# Performanță List - Vector

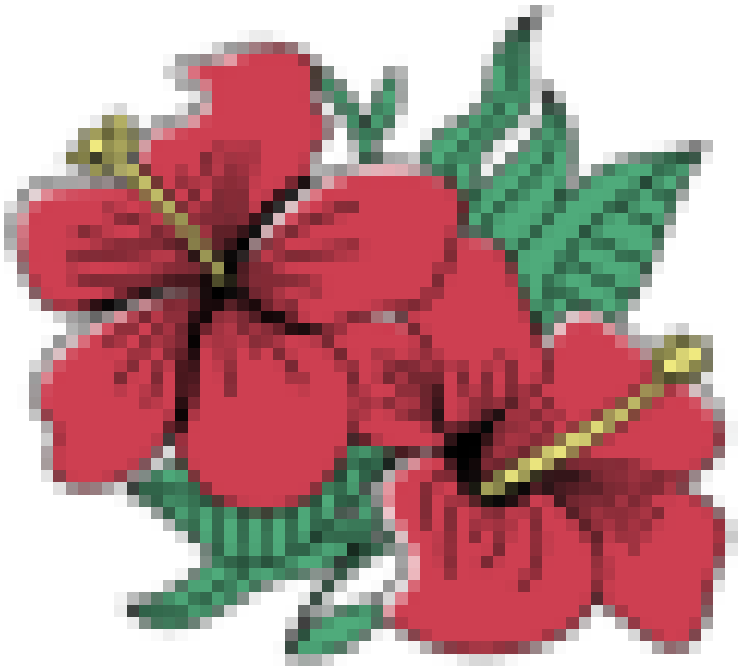
| Operație | 256x256 | 512x512 | 1024x1024 |
|---|---|---|---|
| Divizare(s) | 2.14 | 60.9 | 634 |
| Îmbinare(s) | | | |
| Nivel 0 | 0.104 | 0.375 | 1.60 |
| Nivel 1 | 0.081 | 0.297 | 1.21 |
| Nivel 2 | 0.075 | 0.273 | 1.10 |
| Nivel 3 | 0.073 | 0.263 | 1.08 |
| Nivel 4 | 0.072 | 0.269 | 1.05 |

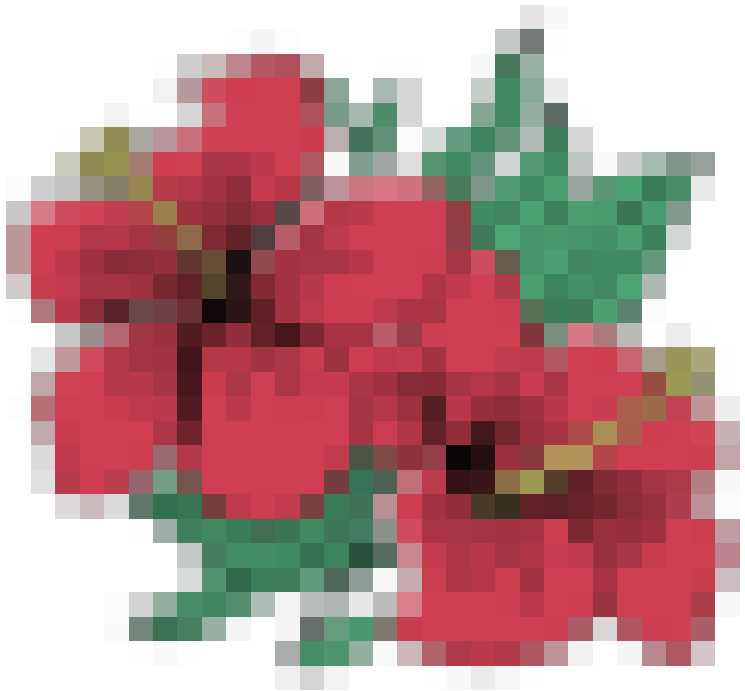
# Cuprins

# Hibiscus 256x256

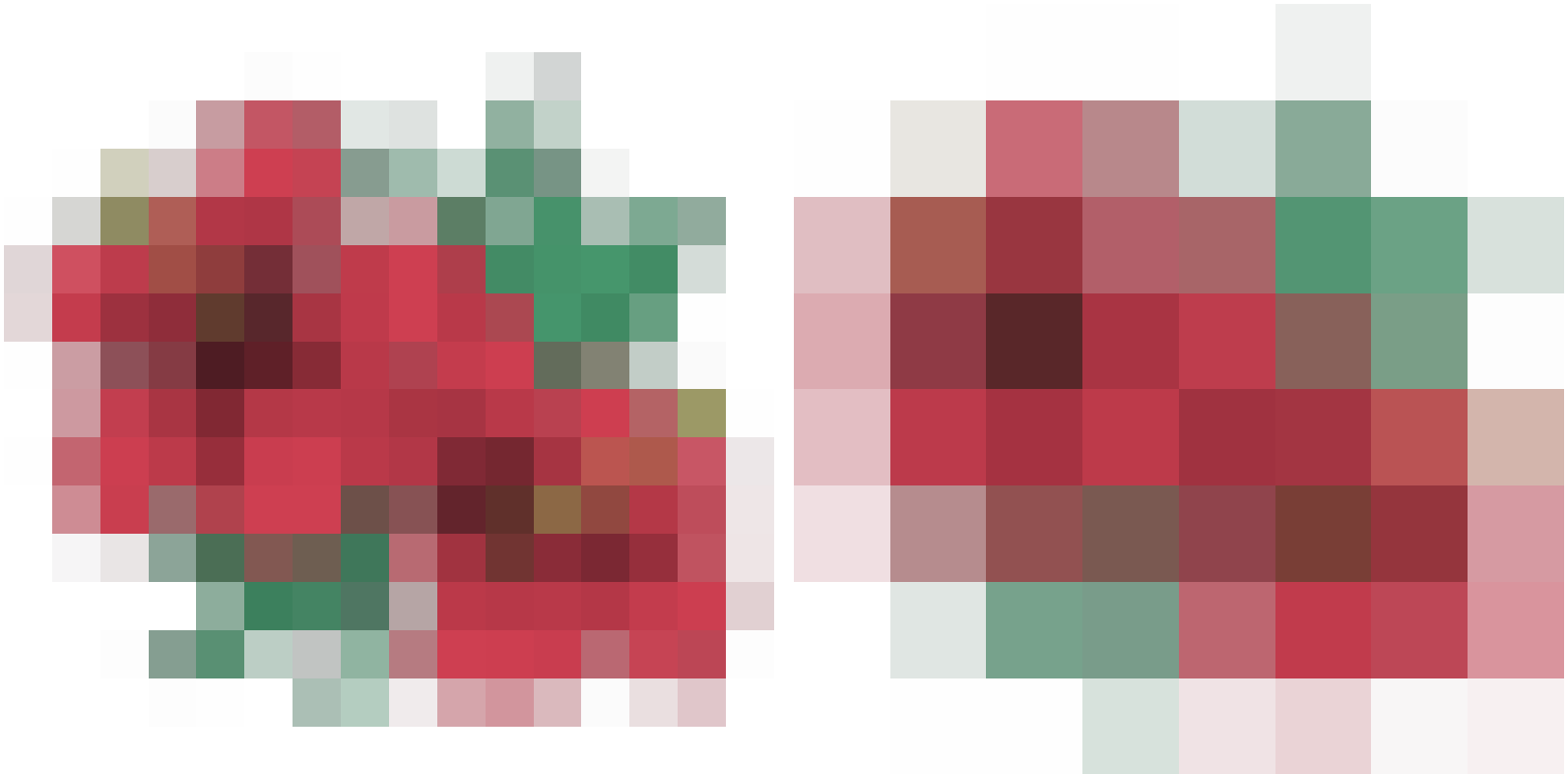


# Hibiscus 256x256

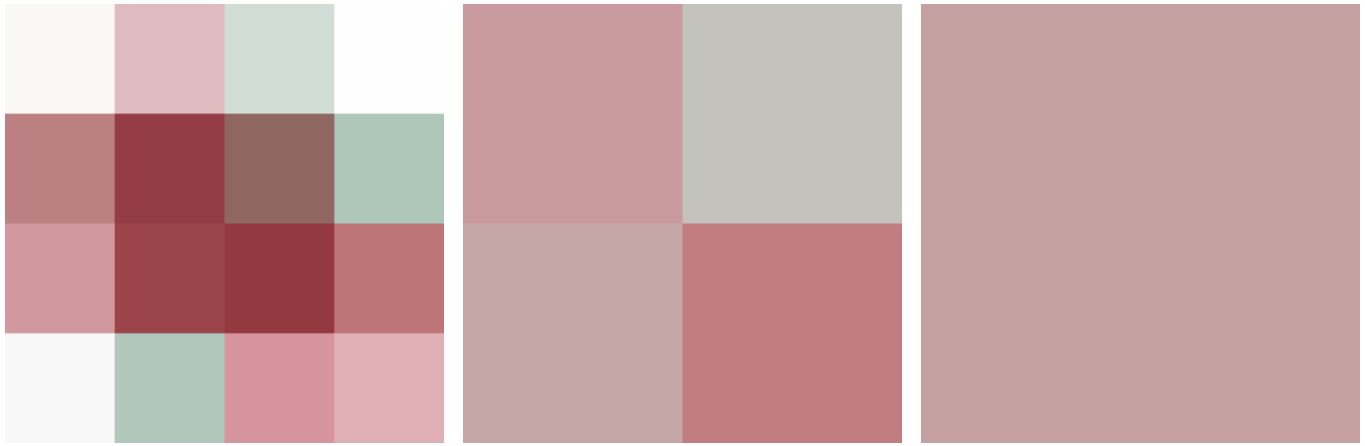




# Hibiscus 256x256



# Hibiscus 256x256



# Explozie 1024x1024

# Cuprins

# Concluzii

- Compresia cu ajutorul unui arbore quad este eficientă, mai ales pentru o reducere a unui număr relativ mic de niveluri, pentru a păstra claritatea imaginii, dar suficient de mare astfel încât dimensiunea fișierului să se înjumătățească.
- Se observă diferențe semnificative între diferitele structuri de date folosite, în cadrul acelorași teste.
- Timpii de așteptare in cazul unui algoritm recursiv pentru încărcarea imaginilor sunt suportabili.
- Reducerea numărului de culori diferite duce la un factor de compresie ridicat.