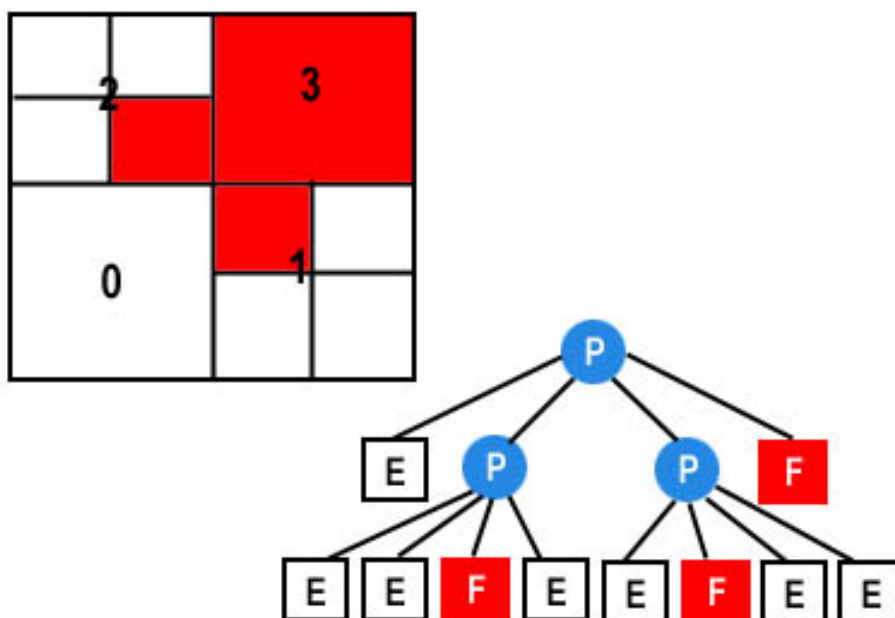


Structuri de date

Tema 2

Deadline: 02.05.2022

Autor: Mihai Nan
mihai.nan@upb.ro



1 Obiective

În urma realizării acestei teme, studentul va fi capabil:

- să implementeze și să utilizeze **arbori** în rezolvarea unor probleme;
- să înțeleagă **modul de reprezentare** a unei imagini;
- să implementeze operații standard **de prelucrare a imaginilor**;
- să transpună o problemă din viața reală într-o problemă care **uzitează arbori cuaternari**.

2 Descriere

Trăim într-o lume a imaginii; suntem, fără îndoială, consumatori de imagini. Paradoxal însă, nu suntem nici pregătiți, nici învățați să lecturăm imaginile. Atunci când suntem puși în situația de a le interpreta, o facem mai mult intuitiv, pentru că nu avem prea multe repere în acest domeniu, nu știm ce trebuie să vedem și cum. Fotografia dezvoltă moduri distincte de a vedea lumea. Ea poate evoca sau poate impune reprezentarea care determină recunoașterea lucrului sau ființei care fie că a disparut, fie că s-a transformat după fixarea sa în alb-negru sau color. Prin urmare, ea reprezintă o frântură fractalică a realității.

Această temă dorește să vă introducă în vasta lumea a imaginilor și vă propune spre analiză și implementare o metodă de compresie a imaginilor. **Termenul de compresie a imaginilor** se referă la o clasă largă de tehnici și metode al căror scop este **reprezentarea unei imagini date folosind un număr cât mai mic de biți**. Odată cu evoluția tehnologiei și, implicit, cu creșterea calității imaginilor, necesitatea reducerii cantității de informație necesară reprezentării unei imagini a devenit evidentă.

Procesul de recompunere a imaginii inițiale, din reprezentarea restrânsă, **se numește decompresie**. Este evident că, prin decodare, trebuie să se obțină o imagine cât mai apropiată de imaginea originală.

3 Compresia imaginilor

3.1 Arbore cuaternar

Arborele cuaternar este o structură de date care organizează informații pentru date multidimensionale, fiind folosită în cartografie, procesarea imaginilor, grafică pe calculator etc. Structura este un arbore ce reprezintă o zonă din spațiul N -dimensional (noi vom analiza, în cadrul acestei teme, **cazul $N = 2$**), fiecare nod al arborelui păstrează informație pentru **o zonă din spațiu**, iar nodul are 2^N fii, care reprezintă fiecare o zonă de 2^N ori mai mică decât zona părintelui. Zonele fiilor sunt disjuncte, iar reuniunea lor formează **zona părintelui**. Cu alte cuvinte, structura pe care o vom utiliza, în cadrul acestei teme, este un arbore în care fiecare nod neterminal va avea **fix 4 descendenți**.

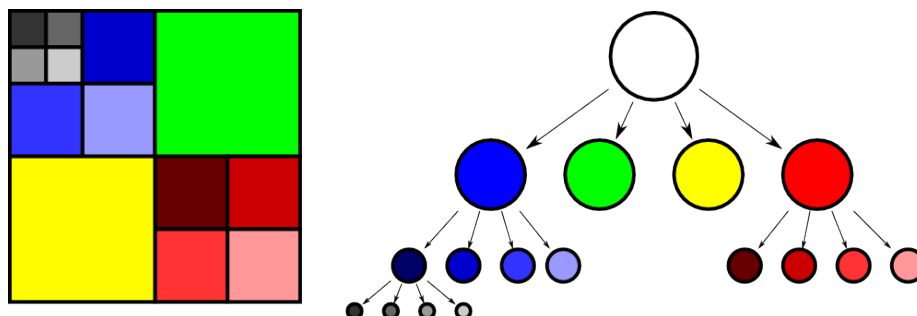
Pentru un **arbore cuaternar**, **rădăcina reprezintă un pătrat**, iar descendenții reprezintă cele patru pătrate disjuncte și congruente, în care se poate împărți pătratul inițial. Din câte am prezentat până acum, un arbore cuaternar este infinit, însă noi vom utiliza această structură de date pentru reprezentarea unei imagini cu o dimensiune redusă din plan. Din acest motiv, vom folosi numai câteva nivele din arborele cuaternar.

3.2 Algoritmul de compresie

Orice imagine pătrată, de dimensiune **putere a lui 2**, poate fi reprezentată **printr-un arbore cuaternar**. Nodurile de pe fiecare nivel al arborelui corespund unei împărțiri a unei zone pătrate din imagine în patru sferturi. Rădăcina arborelui este asociată **întregii imagini**, nodurile de pe primul nivel al arborelui corespund **celor patru sferturi ale imaginii**, nodurile de pe **nivelul doi corespund sferturilor fiecărui sfert anterior și așa mai departe**. Împărțirea imaginii poate continua până când nodurile nivelului curent al arborelui corespund unor **zone pătrate uniforme**: dacă regiunea pătrată considerată **nu este uniformă**, atunci aceasta va fi descompusă, prin tăiere, în patru părți egale și nodul corespunzător va deveni neterminal, având **patru descendenți**. Dacă regiunea considerată este **uniformă** (compusă din pixeli de același fel), nodul respectiv **devine un nod frunză**.

(terminal) al arborelui. Fiecare nod terminal conține informație, corespunzătoare valorii zonei de imagine la care este asociat.

Pentru o mai bună înțelegere a algoritmului descris în paragraful anterior, se propune analizarea exemplului de mai jos.



4 Decompresia imaginilor

Pentru refacerea imaginii inițiale, din reprezentarea arborescentă, este suficientă alegerea nodurilor terminale a căror valoare corespunde pixelilor de obiect. Adâncimea la care este plasată în arbore o frunză conține informația de dimensiune a zonei pătrate corespunzătoare din imagine (o frunză situată la adâncimea h corespunde unei zone pătrate de latură 2^{K-h} pixeli). Poziția frunzei față de nodurile de pe același nivel ce au același predecesor este direct determinată de regula de alocare a sferturilor unei zone la nodurile descendente ale arborelui (regula de alocare trebuie să se păstreze pentru întregul arbore).

5 Formatul fișierelor

5.1 Fișierul PPM

Un fișier în format **PPM** conține un **antet**, în format **text**, care conține: pe prima linie tipul fișierului (în cazul imaginilor folosite în test, o să fie **tipul P6**), pe a doua linie două numere naturale (**width** și **height**), separate prin spațiu, care descriu dimensiunile imaginii, iar a treia linie va conține un număr natural reprezentând valoarea maximă pe care o poate lua o culoare (în cazul testelor folosite, **valoarea este 256**); și imaginea propriu-zisă, în format **binar**.

5.1.1 Imaginea propriu-zisă

Această secțiune este reprezentată printr-o **matrice de pixeli** care ocupă cea mai mare parte a fișierului. Numărul de elemente din tabou este egal cu produsul dintre numărul de pixeli pe linie (**width**) și numărul de pixeli pe coloană (**height**), tabloul fiind organizat **pe linii și coloane**, începând cu linia de sus a imaginii, pixelul din stânga.

Observație

Fiecare pixel este descris prin canalele aflate în ordine **RGB** (**R**ed **G**reen **B**lue).

5.2 Fișierul comprimat

Fișierul comprimat va conține informațiile rezultate în urma procesului de compresie:

- **număr_culori** - de tip **uint32_t** care specifică numărul de blocuri, cu informație utilă, utilizate în cadrul procesului de compresie (numărul de frunze ale arborelui cuaternar de compresie);
- **număr_noduri** - de tip **uint32_t** care specifică numărul total de noduri ale arborelui cuaternar creat;
- **vector** - un vector care conține număr_noduri elemente de tip structură **QuadtreeNode**, fiecare element reprezentând un nod din arborele de compresie creat.

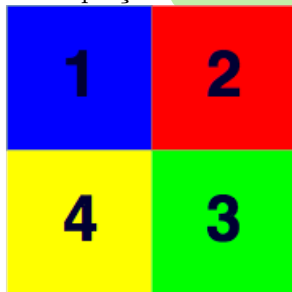
Cod sursă C

```
1 typedef struct QuadtreeNode {
2     unsigned char blue, green, red;
3     uint32_t area;
4     int32_t top_left, top_right;
5     int32_t bottom_left, bottom_right;
6 } __attribute__((packed)) QuadtreeNode;
```

Detalierea structurii

- **area** - reprezintă numărul de pixeli din blocul descris de nod;
- **blue, green, red** - componentele care formează o culoare ce definește blocul analizat;
- **top_left** - reprezintă indicele, din vectorul de structuri, asociat nodului copil-1;
- **top_right** - reprezintă indicele, din vectorul de structuri, asociat nodului copil-2;
- **bottom_right** - reprezintă indicele, din vectorul de structuri, asociat nodului copil-3;
- **bottom_left** - reprezintă indicele, din vectorul de structuri, asociat nodului copil-4.

Pentru a înțelege modul în care se vor numerota descendenții unui nod din arborele de compresie, se va propune o reprezentare grafică a unei astfel de împărțiri. În cazul frunzelor, acești indici vor primi valoarea -1.



⚠ IMPORTANT !



Numerotarea indicilor începe de la 0, pe prima poziție reținându-se rădăcina arborelui de compresie cuaternar.

5.3 Fișierul decomprimat

Fișierul decomprimat este un fișier standard în format **PPM** și conține informațiile extrase din fișierul comprimat dat pentru decompresie.

6 Cerințe

6.1 Cerința 1

Pentru prima cerință a temei, trebuie să se realizeze codificare unei imagini în format **PPM**, folosind algoritmul de compresie detaliat în secțiunea anterioară a enunțului. În rezolvarea acestei cerințe, este **obligatorie**

implementarea arborelui cuaternar de compresie.

Pentru a determina când s-a ajuns la un bloc care poate fi reprezentat în arborele cuaternar de compresie ca nod frunză, cu alte cuvinte, nu mai este nevoie să fie divizat în alte 4 zone de dimensiune egală, se va calcula culoarea medie a blocului, determinând pentru fiecare canal (RED, GREEN și BLUE) media aritmetică a valorilor din submatricea de pixeli care corespunde blocului. După ce a fost determinată culoarea medie, se calculează un scor al similarității pentru blocul respectiv, folosind următoarea formulă:

$$mean = \frac{\sum_{i=x}^{x+size} \sum_{j=y}^{y+size} (red - grid[i][j].red)^2 + (green - grid[i][j].green)^2 + (blue - grid[i][j].blue)^2}{3 \cdot size \cdot size}$$

unde *red*, *green*, *blue* reprezintă componentele pentru culoarea medie.

Dacă valoarea obținută pentru scor este mai mică sau egală decât pragul impus, atunci nu o să mai fie nevoie de divizare.

6.2 Cerința 2

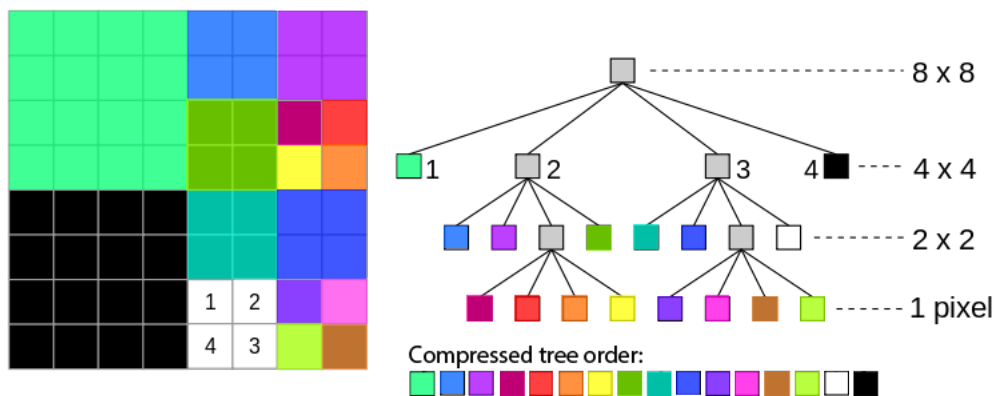
În cadrul acestei cerințe, trebuie să se reconstituie imaginea inițială, pornind de la un fișier rezultat în urma compresiei și utilizând algoritmul de decompresie descris. De asemenea, este **obligatorie** uzitarea structurii de arbore cuaternar în rezolvarea cerinței.

6.3 Cerința 3

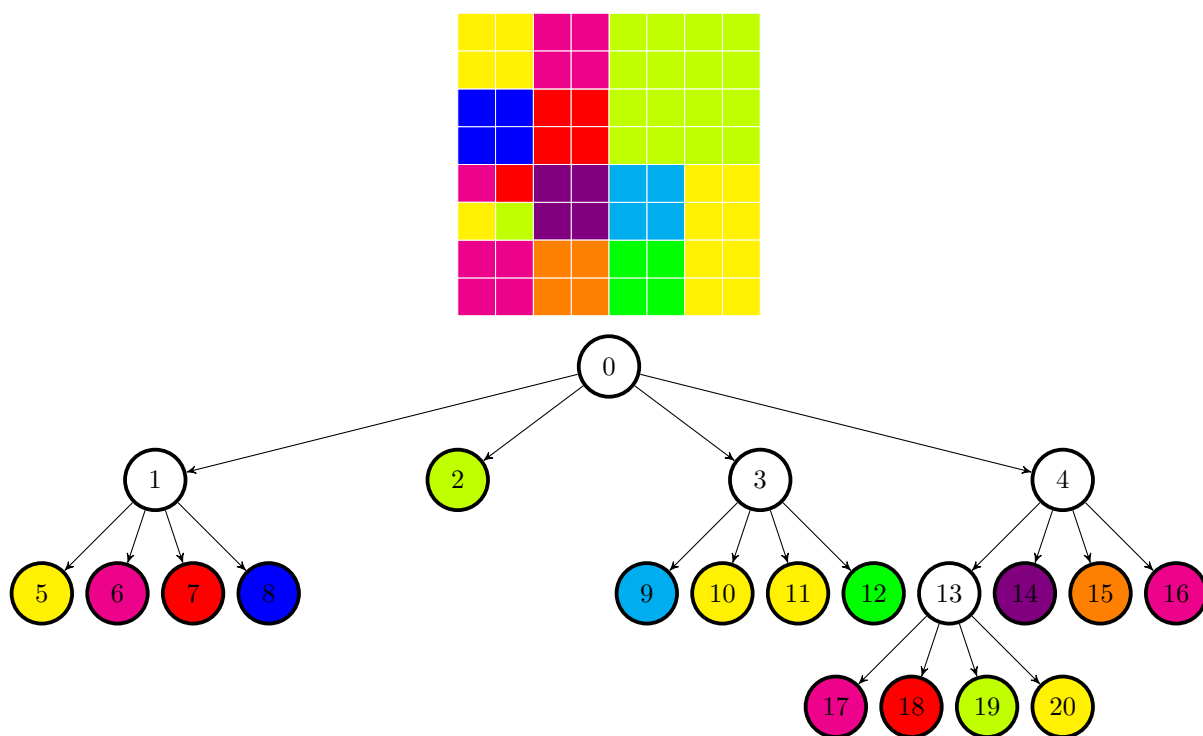
Una dintre cele mai des folosite operații de prelucrare a unei imagini este reprezentată de oglindirea acesteia pe verticală sau orizontală. Pentru realizarea acestei operații, este necesară doar modificarea matricei de pixeli care descrie imaginea propriu-zisă.

În cadrul acestei cerințe, vom realiza oglindirea unei imagini, utilizând arborele cuaternar pentru modelarea acesteia. În cazul unei astfel de reprezentări, oglindirea imaginii se rezumă la modificarea legăturilor către descendenți.

7 Exemplu



Pentru o înțelegere mai bună a cerințelor acestei teme, analizați următorul exemplu, realizat pentru imaginea de mai jos. Imaginile cu care veți lucra, în cadrul acestei teme, nu vor avea zonele despărțite, precum imaginea de mai jos. Inițial, va trebui să citiți imaginea și să construiți arborele cuaternar de compresie care o va modela, structura cu care veți lucra în cadrul cerințelor următoare.



7.1 Cerința 1

După ce am citit imaginea și am format arborele cuaternar, putem realiza compresia imaginii. Pentru acest lucru, vom forma un vector cu elemente de tip *QuadtreeNode*, ce va avea dimensiunea egală cu numărul de noduri din arborele rezultat. În cazul exemplului nostru, acest vector va conține **21** de elemente. Urmează să completăm informațiile pentru fiecare element din vector și să numărăm câte frunze conține arborele nostru, numărul de blocuri, cu informație utilă. Ultima etapă a acestei cerințe constă în scrierea informațiilor în fișierul comprimat.

⚠ IMPORTANT !



Trebuie să respectați precizările din cadrul secțiunii 5.2 și să aveți în vedere faptul că lucrați cu un fișier binar.

Pornind de la următoarea poză, o să vedem cum afectează pragul oferit arborele cuaternar și, implicit, operația de compresie.

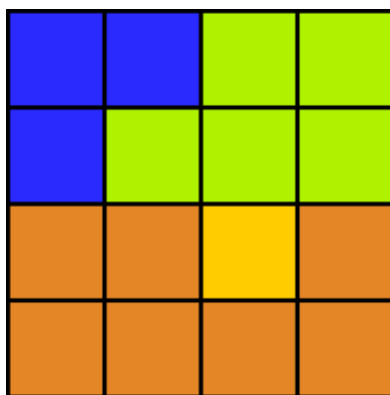
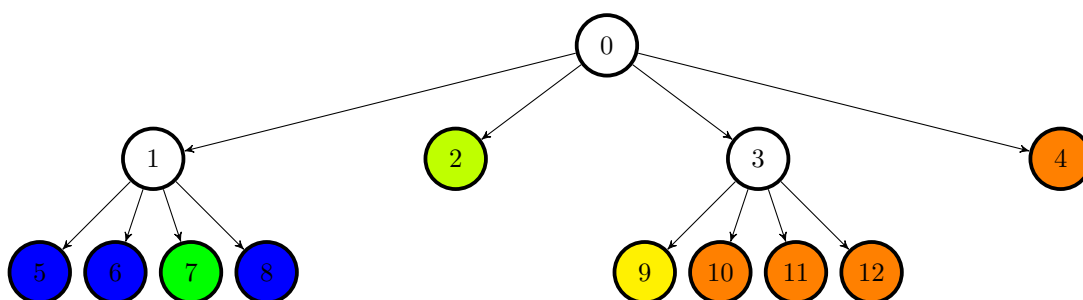


Figure 1: Imaginea inițială

1. Dacă pragul oferit este 0, atunci arborele de compresie o să fie următorul.



Pentru acest arbore, imaginea după decompresie va arăta în felul următor:

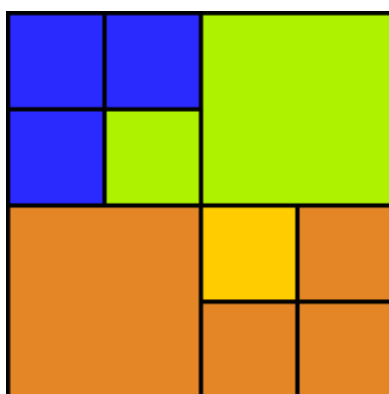
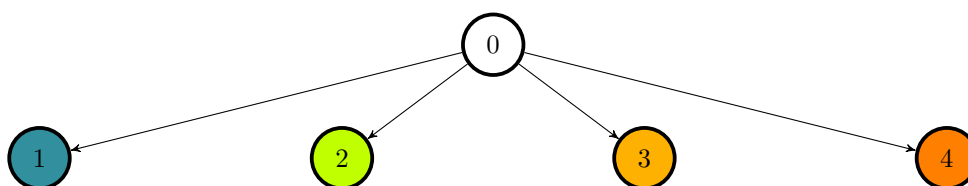


Figure 2: Imagine cu prag 0

2. Dacă pragul oferit este mai mare, atunci arborele de compresie ar putea arăta precum cel mai jos. Fiecare nod frunză va conține culoarea medie a blocului.



Pentru acest arbore, imaginea după decompresie va arăta în felul următor:

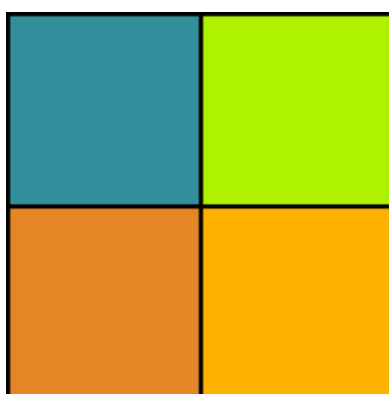


Figure 3: Imagine cu prag mai mare

7.2 Cerința 2

Pentru realizarea acestei cerințe, va trebui să citiți fișierul binar ce reprezintă compresia unei imagini. În acest fișier găsiți informațiile necesare pentru construcția arborelui cuaternar asociat imaginii. Astfel, după ce ați citit fișierul puteți crea un arbore cuaternar. Având această structură ce modelează o imagine, trebuie doar

să formăm imaginea pe baza arborelui.

⚠ IMPORTANT !

⚠ Trebuie să respectați structura unui fișier **PPM**, descrisă în secțiunea 5.1.

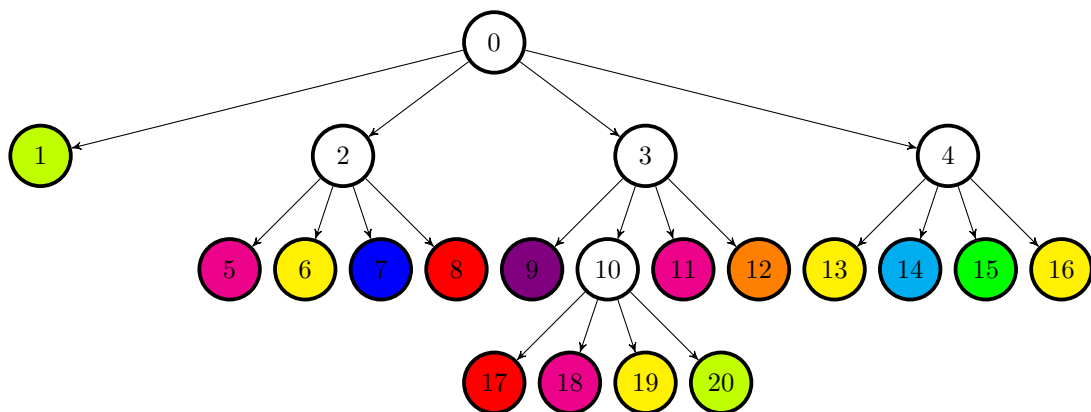
7.3 Cerința 3

Pentru a oglindi o imagine reprezentată folosind un arbore cuaternar de compresie, trebuie să inversăm ordinea nodurilor copil pentru fiecare nod părinte. După ce aplicăm o oglindire pe verticală asupra imaginii din exemplu, vom obține următoarea imagine.

7.4 Oglindire orizontală



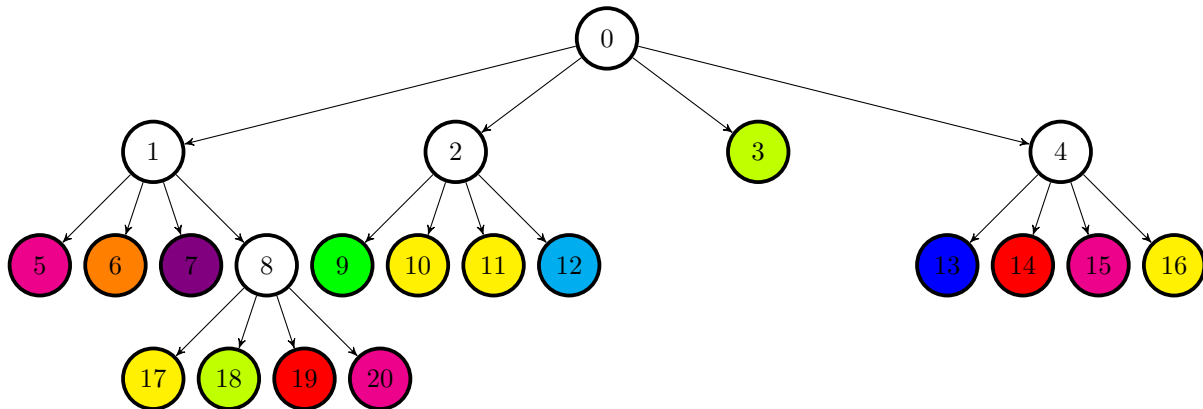
Acum că am văzut cum se modifică imaginea după oglindirea orizontală, trebuie să vedem și ce se va întâmpla cu arborele cuaternar asociat. Pentru a înțelege exact ce operații trebuie să aplicăm, în cadrul implementării, urmăriți atent diferențele dintre cei doi arbori. După ce veți realiza modificarea arborelui, va trebui să formați noua imagine asociată arborelui, păstrând header-ul imaginii inițiale nemodificat.



7.5 Oglindire verticală



Acum că am văzut cum se modifică imaginea după oglindirea verticală, trebuie să vedem și ce se va întâmpla cu arborele cuaternar asociat. Pentru a înțelege exact ce operații trebuie să aplicați, în cadrul implementării, urmăriți atent diferențele dintre cei doi arbori. După ce veți realiza modificarea arborelui, va trebui să formați noua imagine asociată arborelui, păstrând header-ul imaginii inițiale nemodificat.



8 Restricții și precizări

Temele trebuie să fie încărcate pe [vmchecker](#). **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.

O rezolvare constă într-o arhivă de tip **zip** care conține toate fișierele sursă alături de un **Makefile**, ce va fi folosit pentru compilare, și un fișier **README**, în care se vor preciza detaliile implementării.

Makefile-ul trebuie să aibă obligatoriu regulile pentru **build** și **clean**. Regula **build** trebuie să aibă ca efect compilarea surselor și crearea binarului **quadtrees**.

Programul vostru va primi, ca argumente în linia de comandă, numele fișierului de intrare și a celui de ieșire, dar și o opțiune în felul următor:

`./quadtrees [-c factor | -d | -m type] [fișier_intrare1] [fișier_ieșire]` unde:

- **-c factor** indică faptul că programul va rezolva cerința 1 (**factor** = pragul impus pentru compresie);
- **-d** indică faptul că programul va rezolva cerința 2;
- **-m type** indică faptul că programul va rezolva cerința 3 (**type** = tipul oglindirii și poate lua una din valorile: **h** - orizontală sau **v** - verticală);
- **fișier_intrare1** reprezintă numele primului fișier de intrare;
- **fișier_ieșire** reprezintă numele fișierului de ieșire, în care se va scrie, în funcție de comanda primită, rezultatul execuției programului.

9 Punctaj

Cerința	Punctaj
Cerința 1	30 puncte
Cerința 2	40 puncte
Cerința 3	20 puncte
Coding style, README, warning-uri	10 puncte

10 Referințe

1. [PPM Format Specification](#)

Atenție!

Orice rezolvare care nu conține structurile de date specificate nu este punctată.

Temele vor fi punctate doar pentru testele care sunt trecute pe vmchecker.

Nu lăsați warning-urile nerezolvate, deoarece veți fi depunctați.

Dealocați toată memoria alocată pentru reținerea informațiilor, deoarece se vor depuncta pierderile de memorie.

Tema este individuală! Toate soluțiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului.