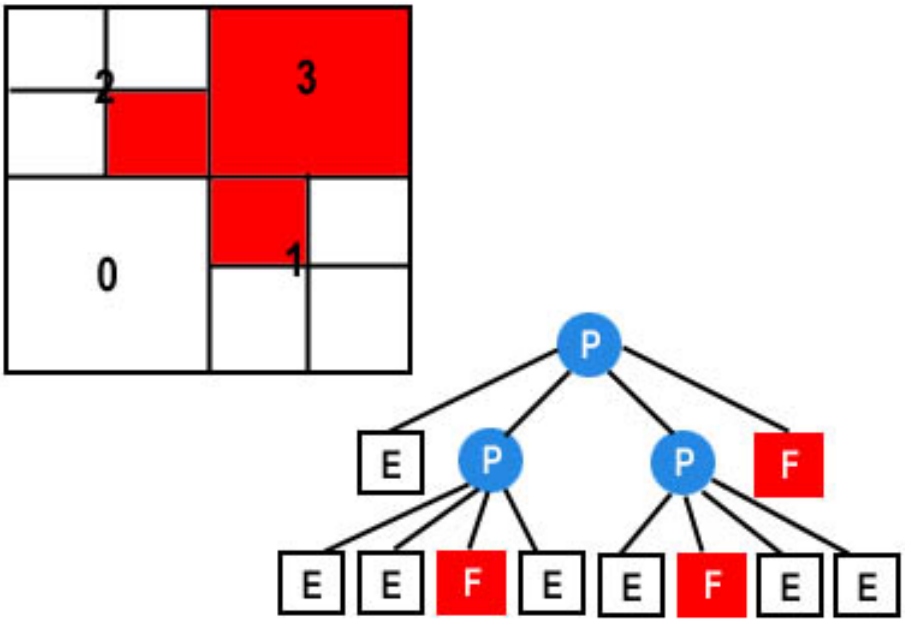
Structuri de date



1 Obiective

În urma realizării acestei teme, studentul va fi capabil:

- să implementeze şi să utilizeze arbori în rezolvarea unor probleme;
- să înţeleagă modul de reprezentare a unei imagini;
- să implementeze operaţii standard de prelucrare a imaginilor;
- să transpună o problemă din viaţa reală într-o problemă care uzitează arbori cuaternari.

2 Descriere

Trăim într-o lume a imaginii; suntem, fără îndoială, consumatori de imagini. Paradoxal însa, nu suntem nici pregătiţi, nici invătaţi să lecturăm imaginile. Atunci când suntem puşi în situaţia de a le interpreta, o facem mai mult intuitiv, pentru că nu avem prea multe repere în acest domeniu, nu ştim ce trebuie să vedem şi cum. Fotografia dezvoltă moduri distincte de a vedea lumea. Ea poate evoca sau poate impune reprezentarea care determină recunoaşterea lucrului sau fiinţei 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 realităţii.

Această temă doreşte să vă introducă în vasta lumea a imaginilor şi vă propune spre analiză şi implementare o metodă de compresie a imaginilor. Termenul de compresie a imaginilor se referă la o clasă largă de tehnici şi metode al căror scop este reprezentarea unei imagini date folosind un număr cât mai mic de biţi. Odată cu evoluţia tehnologiei şi, implicit, cu creşterea calităţii imaginilor, necesitatea reducerii cantităţii de informaţie necesară reprezentării unei imagini a devenit evidentă.

Procesul de recompunere a imaginii iniţiale, din reprezentarea restrânsă, se numeşte decompresie. Este evident că, prin decodare, trebuie să se obţină o imagine cât mai apropiată de imaginea originală.

3 Compresia imaginilor

3.1 Arbore cuaternar

Arborele cuaternar este o structură de date care organizează informaţii pentru date multidimensionale, fiind folosită în cartografie, procesarea imaginilor, grafică pe calculator etc. Structura este un arbore ce reprezintă o zonă din spaţiul N-dimensional (noi vom analiza, în cadrul acestei teme, cazul N = 2), fiecare nod al arborelui păstrează informaţie pentru o zonă din spaţiu, 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 descendenţi.

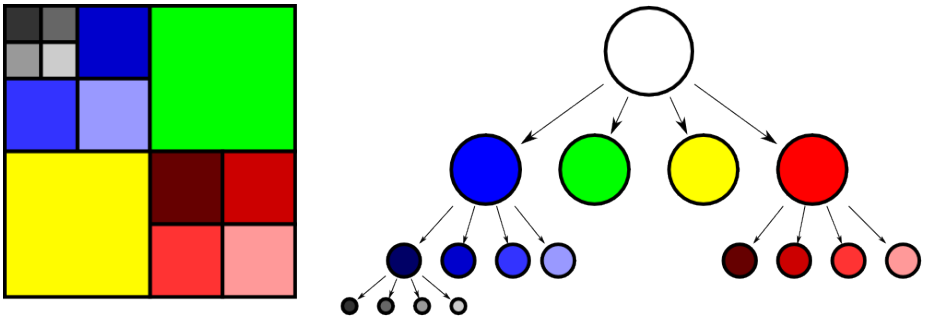
Pentru un arbore cuaternar, rădăcina reprezintă un pătrat, iar descendenţii reprezintă cele patru pătrate disjuncte şi congruente, în care se poate împărţi pătratul iniţial. Din câte am prezentat până acum, un arbore cuaternar este inifinit, însă noi vom uzita această structură de date pentru reprezentarea 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 împărţiri 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 şi aşa mai departe. Împărţirea 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 părţi egale şi nodul corespunzător va deveni neterminal, având patru descendenţi. Dacă regiunea considerată este uniformă (compusă din pixeli de acelaşi fel), nodul respectiv devine un nod frunză

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

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



4 Decompresia imaginilor

Pentru refacerea imaginii iniţiale, din reprezentarea arborescentă, este suficientă alegerea nodurilor terminale a căror valoare corespunde pixelilor de obiect. Adâncimea la care este plasată în arbore o frunză conţine informaţia de dimensiune a zonei pătrate corespunzătoare din imagine (o frunză situată la adâncimea h corespunde unei zone pătrate de latură 2^{K-h} pixeli). Poziţia frunzei faţă de nodurile de pe acelaşi nivel ce au acelaşi 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 fişierelor

5.1 Fişierul PPM

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

5.1.1 Imaginea propriu-zisă

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

Observaţie

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

5.2 Fişierul comprimat

Fişierul comprimat va conţine informaţiile rezultate în urma procesului de compresie:

- ***număr_culori* - de tip uint32_t** care specifică numărul de blocuri, cu informaţie 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 conţine **număr_noduri** elemente de tip structură **QuadtreeNode**, fiecare element reprezentând un nod din arborele de compresie creat.

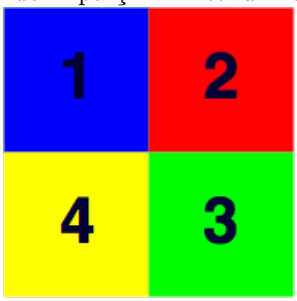
Cod sursă C

```c
typedef struct QuadtreeNode {
  unsigned char blue, green, red;
  uint32_t area;
  int32_t top_left, top_right;
  int32_t bottom_left, bottom_right;
} __attribute__((packed)) QuadtreeNode;
```

Detalierea structurii

- ***area*** - reprezintă numărul de pixeli din blocul descris de nod;
- ***blue, green, red*** - componentele care formează o culoare ce defineşte 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 înţelege modul în care se vor numerota descendenţii unui nod din arborele de compresie, se va propune o reprezentare grafică a unei astfel de împărţiri. În cazul frunzelor, aceşti indici vor primi valoarea **-1**.



⚠ IMPORTANT !

⚠ Numerotarea indicilor începe de la 0, pe prima poziţie reţinându-se rădăcina arborelui de compresie cuaternar.

5.3 Fişierul decomprimat

Fişierul decomprimat este un fişier standard în format **PPM** şi conţine informaţiile extrase din fişierul comprimat dat pentru decompresie.

6 Cerinţe

6.1 Cerinţa 1

Pentru prima cerinţă a temei, trebuie să se realizeze codificare unei imagini în format **PPM**, folosind algoritmul de compresie detaliat în secţiunea anterioară a enunţului. În rezolvarea acestei cerinţe, 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** ş **BLUE**) media aritmetică a valorilor din submatricea de pixeli care corespunde blocului. După ce a fost determinată culoarea medie, se calculează un scor al similarităţii 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 obţinută pentru scor este mai mică sau egală decât pragul impus, atunci nu o să mai fie nevoie de divizare.

6.2 Cerinţa 2

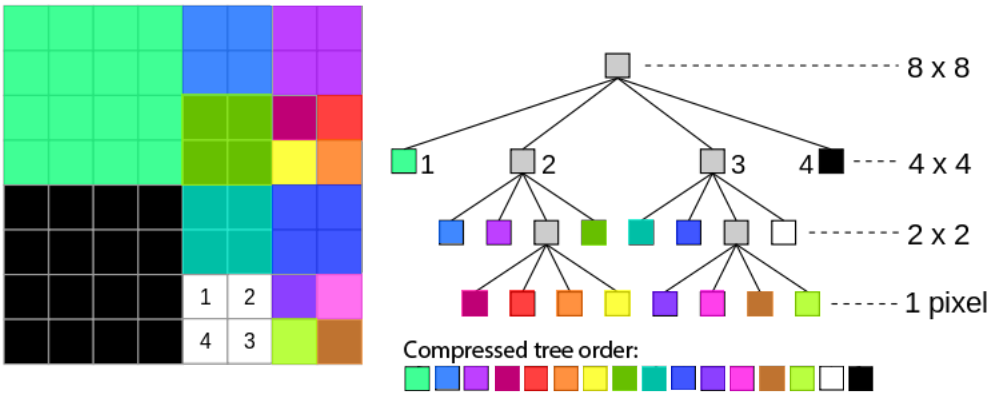
În cadrul acestei cerinţe, trebuie să se reconstituie imaginea iniţială, pornind de la un fişier rezultat în urma compresiei şi utilizând algoritmul de decompresie descris. De asemenea, este **obligatorie** uzitarea structurii de arbore cuaternar în rezolvarea cerinţei.

6.3 Cerinţa 3

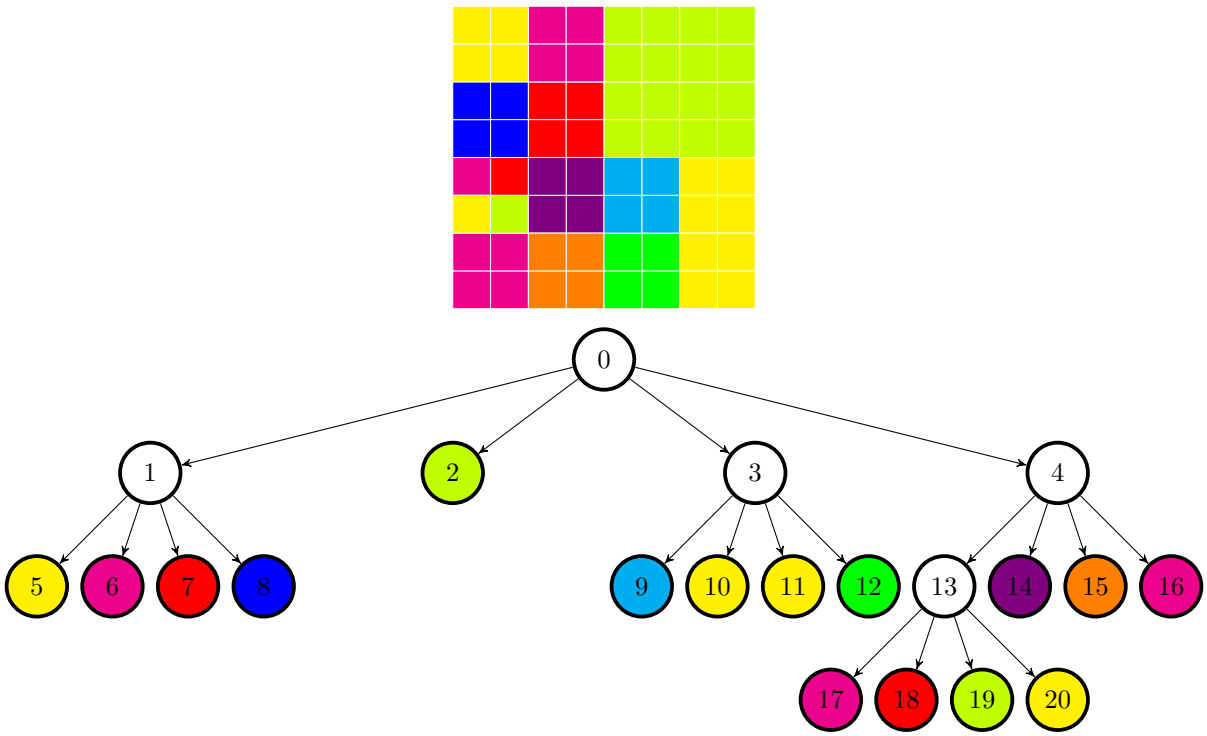
Una dintre cele mai des folosite operaţii de prelucrare a unei imagini este reprezentată de oglindirea acesteia pe verticală sau orizontală. Pentru realizarea acestei operaţii, este necesară doar modificarea matricei de pixeli care descrie imaginea propriu-zisă.

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

7 Exemplu



Pentru o înţelegere mai bună a cerinţelor acestei teme, analizaţi următorul exemplu, realizat pentru imaginea de mai jos. Imaginile cu care veţi lucra, în cadrul acestei teme, nu vor avea zonele despărţite, precum imaginea de mai jos. Iniţial, va trebui să citiţi imaginea şi sa construiţi arborele cuaternar de compresie care o va modela, structura cu care veţi lucra în cadrul cerinţelor următoare.



7.1 Cerinţa 1

După ce am citit imaginea şi 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 conţine **21** de elemente. Urmează să completăm informaţiile pentru fiecare element din vector şi să numărăm câte frunze conţine arborele nostru, numărul de blocuri, cu informaţie utilă. Ultima etapă a acestei cerinţe constă în scrierea informaţiilor în fişierul comprimat.

⚠ IMPORTANT !

⚠ Trebuie să respectaţi precizările din cadrul secţiunii **5.2** şi să aveţi în vedere faptul că lucraţi cu un fişier binar.

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

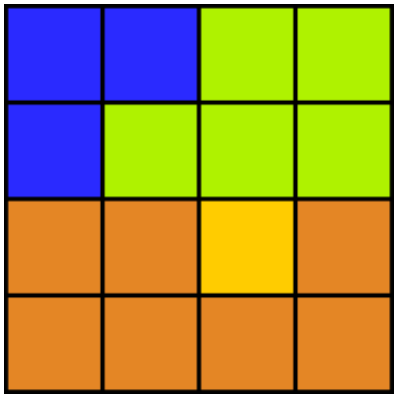
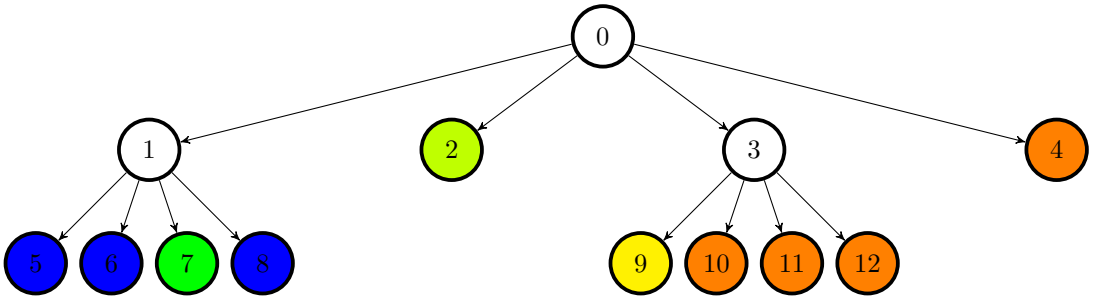


Figure 1: Imaginea iniţială

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:

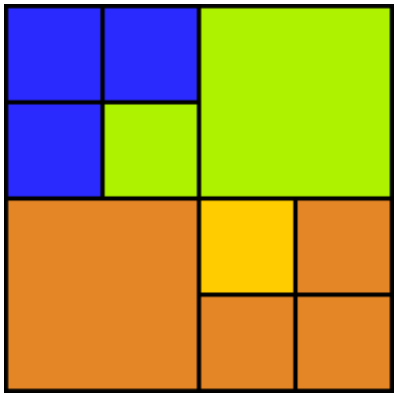
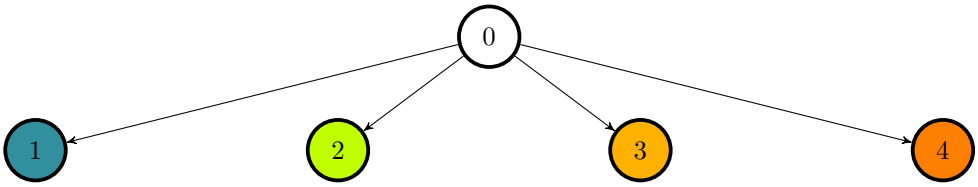


Figure 2: Imagine cu prag 0

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



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

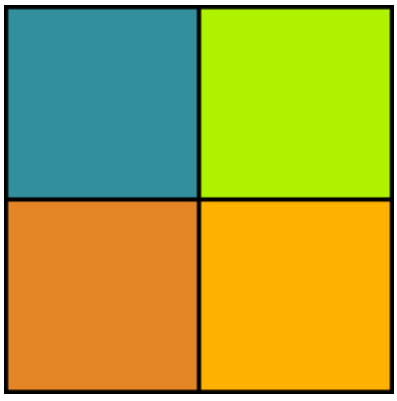


Figure 3: Imagine cu prag mai mare

7.2 Cerinţa 2

Pentru realizarea acestei cerinţe, va trebui să citiţi fişierul binar ce reprezintă compresia unei imagini. În acest fişier găsiţi informaţiile necesare pentru construcţia arborelui cuaternar asociat imaginii. Astfel, după ce aţi citit fişierul puteţi crea un arbore cuaternar. Având această structură ce modelează o imagine, trebuie doar

să formăm imaginea pe baza arborelui.

⚠ IMPORTANT !

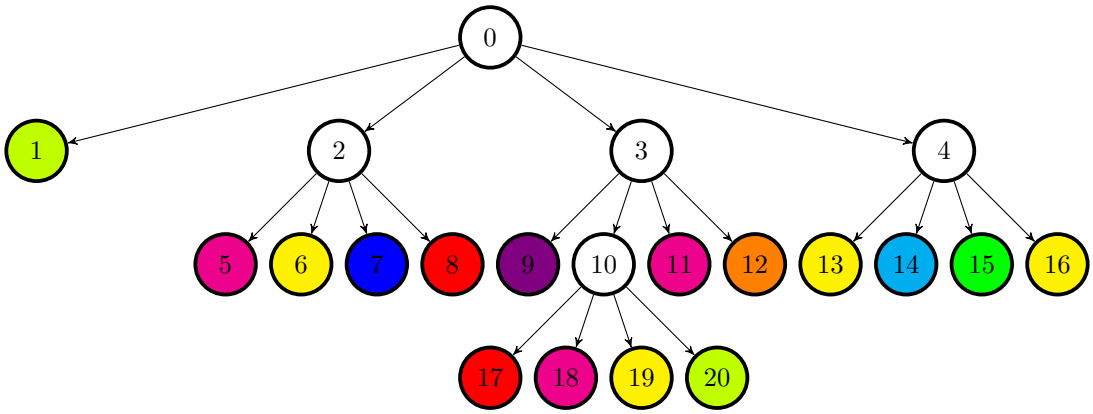
⚠ Trebuie să respectaţi structura unui fişier **PPM**, descrisă în secţiunea 5.1.

7.3 Cerinţa 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 obţine următoarea imagine.

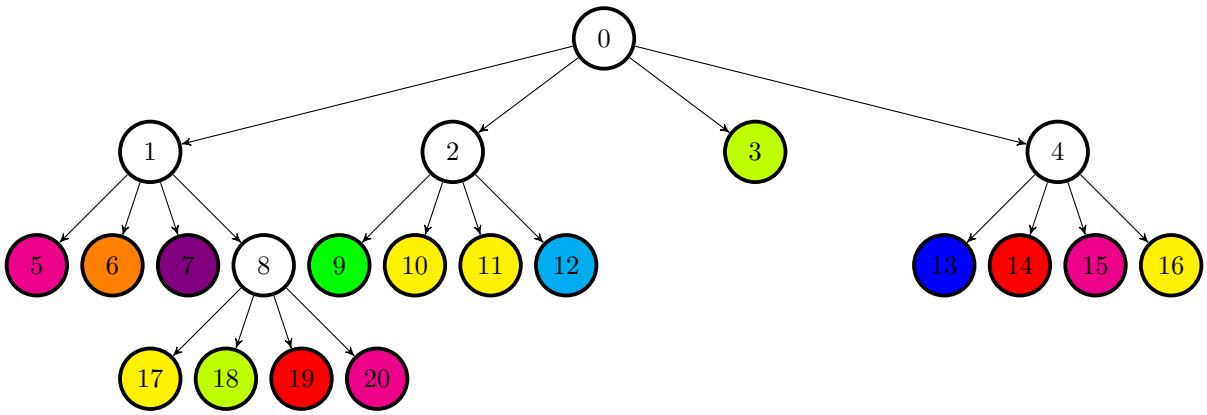
7.4 Oglindire orizontală

Acum că am văzut cum se modifică imaginea după oglindirea orizontală, trebuie să vedem şi ce se va întâmpla cu arborele cuaternar asociat. Pentru a înţelege exact ce operaţii trebuie să aplicaţii, în cadrul implementării, urmariţi atent diferenţele dintre cei doi arbori. După ce veţi realiza modificarea arborelui, va trebui să formaţi noua imagine asociată arborelui, păstrând header-ul imaginii iniţiale nemodificat.



7.5 Oglindire verticală

Acum că am văzut cum se modifică imaginea după oglindirea verticală, trebuie să vedem şi ce se va întâmpla cu arborele cuaternar asociat. Pentru a înţelege exact ce operaţii trebuie să aplicaţii, în cadrul implementării, urmariţi atent diferenţele dintre cei doi arbori. După ce veţi realiza modificarea arborelui, va trebui să formaţi noua imagine asociată arborelui, păstrând header-ul imaginii iniţiale nemodificat.



8 Restricţii şi precizări

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 conţine toate fişierele sursă alaturi de un **Makefile**, ce va fi folosit pentru compilare, şi un fisier README, în care se vor preciza detaliile implementării.

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

Programul vostru va primi, ca argumente în linia de comandă, numele fişierului de intrare şi a celui de ieşire, dar şi o opţiune în felul următor:

./quadtree [-c factor | -d | -m type] [fişier_intrare1] [fişier_ieşire] unde:

- **-c factor** indică faptul că programul va rezolva cerinţa 1 (**factor** = pragul impus pentru compresie);
- **-d** indică faptul că programul va rezolva cerinţa 2;
- **-m type** indică faptul că programul va rezolva cerinţa 3 (**type** = tipul oglindirii şi poate lua una din valorile: `h` - *orizontală* sau `v` - *verticală*);
- **fişier_intrare1** reprezintă numele primului fişier de intrare;
- **fişier_ieşire** reprezintă numele fişierului de ieşire, în care se va scrie, în funcţie de comanda primită, rezultatul execuţiei programului.

9 Punctaj

Cerinta	Punctaj
Cerinţa 1	30 puncte
Cerinţa 2	40 puncte
Cerinţa 3	20 puncte
Codying style, README, warning-uri	10 puncte

10 Referinţe

1. PPM Format Specification

Atenţie!

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

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

Nu lăsaţi warning-urile nerezolvate, deoarece veţi fi depunctaţi.

Dealocaţi toată memoria alocată pentru reţinerea informaţiilor, deoarece se vor depuncta pierderile de memorie.

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