Bordeianu David

332AB

Am implementat modulul process, iar pentru ușurința implementării si a urmăririi acestuia am ales ca stările automatului sa fie direct numerotate, procesul de rezolvare desfășurând-se succesiv. O sa fac o prezentare al fiecărui task, informații despre stările automatului regăsindu-se si in cod (Pe scurt, prin comentarii deasupra anumitor stări de baza).

Pentru task-ul 1, grayscale, sunt folosite stările 0 – 5, in cod fiind menționate si etapele parcurse in proces. Am aflat mai întâi minimul si maximul din in_pix, pentru ca ulterior sa pot face media acestora, cu ajutorul căreia sa compun pixelii de ieșire, si deci, imaginea de ieșire, grayscaled. Procesul are loc din pixel in pixel.

Taskul 2 este mai amplu, acoperind stările 6 – 17. Pentru fiecare AVG, var, si reconstruire efectiva, am parcurs separat blocul, urmând ca pentru fiecare etapa, menționata, sa îl reiau, pentru a o parcurge si pe aceasta, prin urmare: am calculate AVG pentru un bloc, trec la calcularea var-ului unde reiau blocul de la început, calculez Lm, Hm si beta, acestea neimpunând o dificultate ridicata (beta-ul am ales sa îl calculez in porțiunea de calcul a var-ului) iar mai apoi trec la reconstruirea blocului, unde, din nou, îl reiau de la început (Reluarea blocului se face in funcție de i si j, precum si contoarele specifice, in funcție de pasul la care se face reluarea). Matricele imaginilor, respectiv blocurile, am ales sa le parcurg pe linii, adică iau fiecare bloc de la stânga la dreapta matricei, urmând ca mai apoi sa trec la următorul nivel de blocuri. La fiecare porțiune din cod se pot observa o succesiune de if-uri centrate pe iterarea i-urilor si j-urilor, acestea au rolul de a parcurge matricea si fiecare dintre blocuri in modul menționat anterior (Spre exemplu liniile 161-188, din cod). Pentru fiecare proces de calcul, in vederea siguranței, am ales sa iau separat contoare (Acestea se pot observa prima data in starea 6 ((i si j)_init fiind folosite pentru AVG, (i si j)1 pentru var, respectiv (i si j)2 pentru reconstrucție), precum si sume diferite, in vederea calculării AVG si var (suma respectiv suma1). Contor, contor1 si contor2 se ocupa de numărarea celor 16 pixeli din fiecare bloc, respectiv pentru AVG, var si reconstrucția blocului.

Taskul 3 folosește, in mare parte, aceeași parcurgere a blocurilor, respectiv a matricei imaginii, explicata anterior, in task 2. Acesta acoperă stările 18 – 25. In aflarea lui Lm si Hm (Stările 18 – 20) pentru fiecare bloc, am ales sa iau doua contoare (c1, respectiv c2) pentru a păstra pozițiile din bloc, a celor doua valori (Lm, respectiv Hm). Tot aici, am făcut o serie de teste, pentru a rezolva cazurile speciale, legate de pozițiile si valorile lui Lm si Hm pentru fiecare bloc. După aceea, am mers in starea unde asteptam done = 1 (Starea 23) de la modul, pentru ca mai apoi sa continui spre următoarea etapa, cea de encoding propriu-zis (Stările 21, 22 si 24). In cod, am încercat sa explic, sumar, ce face fiecare stare de interes. H_str se ocupa cu salvarea fiecărei porțiuni din hiding string, pentru prelucrarea ulterioara, el fiind incrementat folosind „aux”, iar in base3_done stochez numărul complet convertit in baza 3. Pentru procesul efectiv de encoding, am folosit „pix_aux”, anume, o variabila auxiliara in care am stocat fiecare in_pix din bloc. Stocarea, precum si parcurgerea mesajului secret am făcut-o folosind un c_string, ce parcurge numărul base3_done.