do zdobycia: [25pkt]
deadline: 11/13 V 2015

Zestaw 5

Pora zmierzyć się z problemem, który jest już bardziej złożony. Celem tego ćwiczenia jest napisanie programu przetwarzającego bitmapę w sposób równoległy. Program ten ma poddawać bitmapę rozmyciu gaussowskiemu.

Uwaga: program ten będzie stanowić podstawę do realizacji kolejnego ćwiczenia. Warto się zatem przyłożyć.

Powstały program powinien:

1. wczytać wskazany przez użytkownika plik bitmapy (nazwa pliku zadawana za pomocą wektora argumentów, odczyt danych z pliku realizowany szeregowo, jedynie przez mastera),
2. przesłać do każdego z procesów całość informacji o bitmapie (rozdzielczość, składowe RGB poszczególnych pikseli), stosując do tego celu funkcję `MPI_Bcast`, zadbać o to, by funkcja ta była na każdym z procesów wołana dokładnie raz (konieczność upakowania danych, nie jest dopuszczalne komunikowanie bitmapy piksel po pikselu),
3. dokonać dekompozycji problemu, dbając o równomierny podział pracy,
4. obliczyć (w sposób równoległy) rozmycie gaussowskie, stosując zadany przez użytkownika promienia rozmycia σ (parametr, w ogólności rzeczywisty, wyrażony w pikselach, zadawany przez użytkownika za pomocą wektora argumentów), każdy z procesów powinien obliczać rozmycie gaussowskie jedynie dla swojej części pikseli,
5. przesłać (do mastera) wyniki przetwarzania, uzyskane przez poszczególne procesy,
6. złożyć (na masterze) uzyskane wyniki w całość, zapisać powstały obraz do pliku o nazwie zadanej przez użytkownika, za pomocą wektor argumentów,
7. dodatkowo program powinien dokonywać pomiaru czasów realizacji poszczególnych etapów (odczyt danych, przesyłanie informacji wejściowych o bitmapie, obliczanie rozmycia, przesyłanie informacji wynikowych o bitmapie, zapis wyników), czasy te po zakończeniu przetwarzania powinny być wypisywane przez mastera na ekranie (z wyszczególnieniem procesów).

Dodatkowe uwagi/informacje:

1. nie został określony sposób dekompozycji problemu, można zastosować każdą metodę, gwarantującą w miarę równomierny rozkład pracy (wedle uznania: pikselowa „przeplatanka”, podział wierszowy/kolumnowy, podział na prostokąty),
2. nie należy czynić jakichkolwiek założeń odnośnie rozmiaru (rozdzielczości) przetwarzanej bitmapy, program powinien działać prawidłowo również w sytuacjach „patologicznych” (bitmapy o rozdzielczościach: 1x1, 2x2, 100x1, 1x100, 101x99, itd.),
3. przy obliczaniu rozmycia gaussowskiego **nie** należy stosować promienia odcięcia,

4. rozmycie gaussowskie należy obliczać stosując wzór

$$Q(x', y') = \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} Q(x, y) \times g(x' - x, y' - y)}{\sum_{x=1}^{w} \sum_{y=1}^{h} g(x' - x, y' - y)},$$

gdzie

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right),$$

wielkości w i h oznaczają szerokość i wysokość obrazu, zaś $Q(x, y)$ oznacza wartość Q-tej składowej koloru piksela o współrzędnych (x, y), oczywiście $Q = R, G$ lub B,

5. warto solidnie zastanowić się nad metodą realizacji rozmycia gaussowskiego (minimalizacja liczby wykonywanych operacji, stworzenie odpowiednich „lookup-ów" dla funkcji kernela $g(x, y)$), warto (nawet **należy**) skorzystać z własności

$$\exp(\alpha(x + y)) = \exp(\alpha x) \exp(\alpha y),$$

6. pod adresem

`http://www.mif.pg.gda.pl/homepages/swinczew/AR/bitmapy.zip`

znaleźć można archiwum zawierające dwa przykładowe zestawy kontrolne (dwie bitmapy wejściowe: pliki o nazwach `cookiemonster.bmp` oraz `wachlarz.bmp`; sześć bitmap wyjściowych, uzyskanych dla $\sigma = 1.0, 5.0$ i 10.0).

Kryteria oceniania:

1. prawidłowa i rozsądna metoda rozsyłania informacji wejściowych [2pkt],
2. prawidłowa, zgrabna i rozsądna metoda dekompozycji problemu [4pkt],
3. prawidłowa, zgrabna i **wydajna** metoda obliczania rozmycia gaussowskiego [6pkt],
4. prawidłowa i rozsądna metoda przesyłania informacji wynikowych oraz składania ich w całość [3pkt],
5. estetyka oraz struktura kodu [6pkt],
6. implementacja funcjonalności pomiaru czasów wykonania [2pkt],
7. analiza czasów wykonania wraz ze sformułowaniem wniosków [2pkt].