Egzamin z Programowania Współbieżnego 2021-02-04

Zadanie 1 (komunikacja synchroniczna, 10p.)

Procesy `M[w : 0 .. W - 1][k : 0 .. K - 1]` ($W > 0, K > 0$), wraz z procesem pomocniczym `Z`, implementują rozproszoną macierz wartości typu `porcja`. Procesy `U[n : 0 .. N-1]` ($N > 0$) wykonują operacje na tej macierzy.

Proces `M[w][k]` przechowuje porcję z wiersza `w` i kolumny `k`. Wartość początkową dostaje jako wynik funkcji:

```
porcja początkowa(int w, int k);
```

Proces `U[n]` odczytuje porcję z wiersza `w` i kolumny `k` wykonując instrukcje:

```
send M[w][k].daj(n);
receive odczytana;
```

Proces `U[n]` przesuwa cyklicznie zawartość wiersza `w` o `p` pozycji, dla $0 < p < K$, wykonując instrukcje:

```
send Z.przesuńWiersz(n, w, p);
receive gotowyWiersz(w);
```

Efektem jest przeniesienie porcji w wierszu `w`, dla każdego `k` od `0` do `K - 1` z kolumny `k` do kolumny `(k + p) % K`.

Proces `U[n]` przesuwa cyklicznie zawartość kolumny `k` o `p` pozycji, dla $0 < p < W$, wykonując instrukcje:

```
send Z.przesuńKolumnę(n, k, p);
receive gotowaKolumna(k);
```

Efektem jest przeniesienie porcji w kolumnie `k`, dla każdego `w` od `0` do `W - 1` z wiersza `w` do wiersza `(w + p) % W`.

Polecenie:

Zdefiniuj w notacji Rendezvous procesy `Z` i `M[w][k]`.

Założenia:

Wolno założyć, że stałe `W` i `K` są parzyste a liczba pozycji `p`, o którą przesuwamy wiersz lub kolumnę, jest nieparzysta.

Maksymalna liczba porcji przechowywanych na raz przez proces jest stała, niezależna od sumy `N + W + K`.

W rozwiązaniu można używać standardowych struktur danych (stos, kolejka).

Górne ograniczenie na rozmiar pomocniczych struktur danych procesu jest funkcją liniową sumy `N + W + K`.

Odczytywane porcje powinny być zgodne ze scenariuszem, w którym operacje na macierzy są wykonywane sekwencyjnie i niepodzielnie.

Efektywność rozwiązania będzie miała istotny wpływ na ocenę. Tam, gdzie to możliwe, operacje na macierzy powinny być wykonywane współbieżnie. Koszt przesunięcia elementów powinien być mniejszy niż liniowy względem ich liczby.

Zadanie 2 (przestrzeń krotek, 10p.)

Równanie różniczkowe Laplace'a opisuje różne zjawiska w procesach fizycznych. Analiza takich procesów często sprowadza się do zdyskretyzowania równania i numerycznego rozwiązania powstałego układu równań. Jedną z używanych metod numerycznych jest metoda kolejnych nadrelaksacji. Przyjmijmy, że równanie jest zdyskretyzowane na siatce `K x K` punktów.

Algorytm numeryczny przybliżający rozwiązanie równania działa następująco dla każdego punktu siatki:

- liczy średnią wartość istniejących sąsiednich punktów siatki (górnego, dolnego, prawego i lewego sąsiada) - np. jeżeli punkt ma 4 sąsiadów to:
  `tmp = (x[i+1][j] + x[i][j+1] + x[i-1][j] + x[i][j-1])/4`,
  a dla punktu `(0,0)`: `tmp = (x[1][0] + x[0][1])/2`
- nadaje nową wartość punktowi równą średniej ze średniej policzonej w poprzednim punkcie oraz starej wartości punktu używając zadanego współczynnika relaksacji `A` jako wagi:
  `x'[i][j] = A*x[i][j] + (1-A)*tmp`

Algorytm jest iteracyjny. W każdej iteracji następuje sprawdzenie, czy wartość bezwzględna różnicy pomiędzy wartością `x[i][j]` a `x'[i][j]` przekracza zadane `epsilon`. Jeśli istnieje punkt, dla którego `epsilon` jest przekroczony, to wykonywana jest kolejna iteracja algorytmu dla wszystkich punktów.

Rozważamy współbieżny program wyznaczający na podstawie danych początkowych przybliżenie rozwiązania równania. Program korzysta z przestrzeni krotek. Składa się z procesu:

```
process Główny() {
    zapiszDane();
    tsPut("START");
    tsFetch("STOP");
    odczytajWynik();
}
```

oraz `N`>`1` procesów `Pomocniczy(int id)`, gdzie `id` to unikatowy numer procesu, od `1` do `N`.

Proces `Główny` posługuje się reprezentacją wartości w punkcie za pomocą krotki w formacie `"%d %d %f"`, z numerem wiersza i numerem kolumny tego punktu oraz jego wartości.

Funkcja `zapiszDane()` zapisuje w przestrzeni krotki, które reprezentują dane początkowe. W chwili wykonania funkcji `odczytajWynik()`, w przestrzeni powinny znajdować się tylko krotki, które reprezentują wynik i ewentualnie stała liczba krotek, niezależna od `N`, `K` ani od liczby iteracji algorytmu.

Proces `Pomocniczy` zna wartości `N`, `K`, `A` oraz `epsilon`. Dozwolone jest korzystanie z pomocniczych krotek, ale nie wolno tworzyć dodatkowych procesów. Wielkość krotki oraz rozmiar danych procesu `Pomocniczy` powinnien być stały.

Należy zdefioniować proces `Pomocniczy` dbając o poprawność i efektywność rozwiązania.

Powodzenia!