lab13

Sortowanie przez scalanie (merge sort) - O(n log n)

Rekurencyjny algorytm sortowania wykorzystujący metodę dziel i zwyciężaj.

Oto jak sortowanie przez scalanie używa algorytmu dziel i zwyciężaj:

Podziel przez znalezienie liczby q na pozycji w połowie pomiędzy p i r Robimy to w ten sam sposób w jaki znaleźliśmy środek w przeszukiwaniu binarnym: zsumuj p i r podziel przez 2, i zaokrąglij w dół.

Rządź poprzez rekurencyjne sortowanie podtablic w każdym z dwóch podproblemów, stworzonych w kroku 1. Oznacza to: rekurencyjnie posortuj podtablicę array[p..q]i rekurencyjnie posortuj podtablicę array[q+1..r].

Połącz przez scalanie obie posortowane podtablice z powrotem w jedną posortowaną podtablicę array[p..r].

Potrzebujemy przypadku bazowego. Przypadkiem bazowym jest podtablica zawierająca mniej niż dwa elementy, co ma miejsce, gdy p >= r, wtedy podtablica nie ma już żadnych elementów lub posiada jeden element, który jest już posortowany. Czyli wykonujemy algorytm dziel-i-rządź tylko wtedy, gdy p <r.

Algorytm ma mniejszą złożoność czasową niż proste algorytmy, takie jak np. sortowanie bąbelkowe czy sortowanie przez wstawianie. W zamian za to ma jednak gorszą złożoność pamięciową. Dodatkową zaletą sortowania przez scalanie jest to, że algorytm ten można zrównoleglić. Poszczególne pod tablice można sortować niezależnie od siebie, zatem sortowania te można wykonywać w osobnych wątkach.

Przykładowa implementacja w znanym już nam środowisku mpi: https://github.com/racorretjer/Parallel-Merge-Sort-with-MPI/blob/master/merge-mpi.c