Algorytmy sortujące cz. 2

25 listopada 2015

1 Shell Sort

Algorytm Shell Sort jest modyfikacją algorytmu Insertion Sort. Początkowa nieuporządkowana lista jest dzielona na podlisty, z których każda sortowana jest przy pomocy algorytmu Insertion Sort. Podział na podlisty zadaje ciąg liczb, np liczba 3 oznacza, że tworzymy podlisty z elementów odległych o siebie o 3 miejsca. Dla listy 9-elementowej [3,7,1,2,8,9,0,3,2] daje to 3 grupy: [3,2,0], [7,8,3],[1,9,2].
Złożoność obliczeniowa algorytmu zależy od sposobu podziału listy początkowej na podlisty (szczegółoy można znaleźć np w tablei na https://en.wikipedia.org/wiki/Shellsort). W tym laboratorium należy zaimplementować algorytm, który będzie dzielił zgodnie z liczbami danymi przez ciąg $2^k - 1$, czyli [1,3,7,15,31,]. Podział należy rozpocząć od największej liczby, każdą z tak otrzymanych podlist posortować algorytmem Insertion Sort, a następnie dokonać względem kolejnej, mniejszej liczby. Np. tablicę o długości 1000 elementów, początkowo można podzielić na podlisty względem liczby 511, następnie 255, aż do ostatniego sortowania przy użyciu podziału 1.
Przykład: lista wejściowa (16 elementów) [44, 33, 27, 6, 3, 29, 2, 43, 24, 16, 0, 21, 34, 6, 6, 16]

1. podział względem liczby 7, wynik sortowania podlist: [6, 16, 21, 34, 3, 44, 2, 43, 24, 27, 6, 16, 29, 6, 0, 33]
2. podział względem liczby 3, wynik sortowania podlist: [2, 3, 16, 6, 6, 34, 27, 6, 21, 29, 16, 24, 33, 43, 44, 0]
3. podział względem liczby 1, wynik sortowanie podlist: [0, 2, 3, 6, 6, 6, 16, 16, 21, 24, 27, 29, 33, 34, 43, 44]

2 Merge sort

Algorym Merge Sort jest przykładem strategi dziel i zwyciężа, polegającej na rekurencyjnym podziale zadania na dwa (lub kilka zadań) (dziel), ażdo momentu kiedy te zadania nadają się do bezpośredniego rozwiązania (zwyciężaj). Merge Sort rekurencyjnie dzieli nieuporządkowaną listę na pół. Podlista pusta bądź zawierająca jeden element jest uporządkowana z definicji. Jeżeli lista składa się z dwóch (lub więcej) elementów, algorytm dzieli ją na pół i na obu wywołuje samego siebie. Kiedy dwie połówki listy są posortowane wykonywana jest operacja łączenia, polegający na połączeniu dwóch uporządkowanych list w jedną uporządkowaną listę.
Przykład: lista wejściowa [4,2,5,1]

1. wywołanie Merge Sort na liście [4,2,5,1], podział na listy [4,2] i [5,1] i wywołanie Merge Sort każdej z nich.
2. Merge Sort rozbija listy [4,2] i [5,1] na [4],[2],[5],[1] wywołanie Merge Sort na każdej podliście .
3. każda z podlist jest jednoelementowa, rozpoczyna się proces łączenia [4] z [2] oraz [5] z [1] w wyniku czego otrzymuje się dwie uporządkowane podlisty [2,4] i [1,5].
4. następnie łączone są te dwie uporządkowane podlisty [2,4] i [1,5] w wyniku czego otrzymujemy [1,2,4,5].

3 QuickSort

Kolejnym przykładem zastosowania podejścia dziel i rządź do problemu sortowania jest algorytm QuickSort. Polega on na wybraniu z listy elementu rozdzielającego (np element 0) a następnie podzieleniu listy na dwie części: pierwsza z nich zawiera elementy mniejsze bądź równe elementowi rozdzielającemu, druga natomiast elementy większe. Po uporządkowaniu listy w ten sposób algorytm wywoływany jest rekurencyjnie na dwóch częściach listy. Rekurencyjne wywoływanie algorytmu kończy się w momencie natrafienia na listę 1 elementową, którą przyjmuje się za uporządkowaną.

1. Przykładowy przebieg algorytmu (element względem którego dzielimy: lista[0]):

(a) definijemy zmienne pomocnicze lewa,prawa które posłuża nam do znalezienia miejsca elementu względem którego dzielimy (54), i podzielą listę na 2 części, lewa=1, prawa=8 (ostatni element listy),

(b) dopóki lewa<prawa i lista[lewa]<=lista[0] zwiększamy wartość zmiennel lewa,

(c) dopóki lewa<prawa i lista[praw]>lista[0] zmniejszamy zmienną prawa,

(d) sprawdzamy czy prawa<lewa,

(e) jeżeli tak to znaleźliśmy właściwe miejsce elementu dzielącego (zmienna prawa), możemy umieścić go na właściwym miejscu,

(f) jeżeli nie to natrafiliśmy na elementy będące w niewłaściwych połówkach listy, należy je przestawić i powrócić do iteracyjnego zwiększania (zmniejszania) zmiennej lewa i prawa (punkt b i c)

Przykład lista wejściowa: [44,16,83,7,67,21,34,52,11] Przykładowe działanie: lista[0]=44 - element względem dzielimy

1. punkt b) daje lewa = 2 (83>44) przechodzimy do c)
2. punkt c) daje prawa=8 przechodzimy do d)
3. d) nie jest prawdą, idziemy do f)
4. zamieniamy lista[2] i lista[8], po tej operacji [44,16,11,7,67,21,34,52,83], idziemy do b)
5. lewa=4 (67 > 44) przechodzimy do c)
6. prawa=6 (34 < 44) przechodzimy do d)
7. d) nie jest prawdą, idziemy do f), zamieniamy lista[4] i lista[6], po tej operacji [44,16,11,7,34,21,67,52,83], idziemy do b)
8. lewa = 6 (67 > 44) idziemy do c)
9. prawa= 5 (21 < 44) idziemy do d)
10. d) jest prawdą idziemy do e)
11. zamieniamy lista[0] i lista[5], dostajemy [21,16,11,7,34,44,67,52,83], zmienna prawa trzyma wartość elementu dzielącego.

Następnie znając właściwe miejsce elementu dzielącego można wywołać algorytm QuickSort na dwóch częściach listy: lista[0:(prawa-1)] (gdzie : oznacza wszystkie indeksy od do) i lista[(prawa+1):] (czyli do końca listy). Element względem którego dzieliliśmy listę nie bierze już udziału w dalszych operacjach.
Rekurencyjne działanie algorytmu QuickSort powinno polegać na:

1. sprawdzeniu czy lista do sortowania jest dłuższa niż 1 element,
2. jeśli nie, wyjściu z algorytmu,
3. jeśli tak, to: znalezieniu właściwego miejsca elementu dzielącego (przy jednoczesnym podziale listy na dwie części) a następnie wywołaniu QuickSort na dwóch powstałych częściach listy.

4 Zadania i punktacja

Parametrem wejściowym do programu jest liczba określająca długość listy wejściowej n (do 10000). Program powinien wypełnić listę losowymi elementami z przedziału 0-10000. Wejściowa i wyjściowa lista powinna ma być zapisywana do pliku.

- Implementacja algorytmu Shell Sort wraz ze zliczeniem liczby dokonywanych przestawień 2 punkty
- Implementacja algorytmu Merge sort wraz ze zliczeniem liczby dokonywanych przestawień 2 punkty
- Implementacja algorytmu QuickSort wraz ze zliczeniem dokonywanych operacji przesunięć i przestawień 2 punkty
- Wykonanie wykresu ilości operacji dla powyższych algorytmów w funkcji długości wejściowej listy o rozmiarach n= 1000,5000,10000,15000,20000,25000,30000 - 2 punkty
- Dołączenie do programu funkcji mierzącej czas wykonania programu, wykonanie wykresu czasu wykonania programu w funkcji długości wejściowej listy o rozmiarach n= 1000,5000,10000,15000,20000,25000,30000 - 2 punkty