Algorytmy i złożoność obliczeniowa

Przygotowanie do Laboratorium 10: algorytmy sortowania.

1. Sortowanie przez selekcję (wybieranie) jest to jedna z prostszych metod sortowania o złożoności $O(n^2)$. Polega ona na wyszukaniu np. najmniejszego elementu w tablicy A[1,...,n] i zamianie miejscami tego elementu z pierwszym elementem tablicy. Następnie wyznaczamy najmniejszy element w A[2…n] i zamieniamy go z drugim elementem w tablicy itd., aż cała tablica zostanie posortowana.

Algorytm przedstawia się następująco:

1. wyszukaj minimalną wartość z tablicy spośród elementów od i+1 do końca tablicy
2. zamień wartość minimalną, z elementem na pozycji i

Gdy zamiast wartości minimalnej wybierana będzie maksymalna, wówczas tablica będzie posortowana od największego do najmniejszego elementu.

```
procedure selectionsort;
var i, j, min: integer;
begin
    for i : = 1 to n - 1 do
    begin {A[1] <= ... <= A[i –l] <=  A[i ... n]}
       min := i;
        for j : = i + 1 to n do
        if A[j] < A[min] then min : = j;.
        A[min] < - > A[i];
           {zamiana miejscami A[min] z A[i]}
     end
end selectionsort;
```

2. Sortowanie przez wstawianie: Algorytm **sortowania przez wstawianie** można porównać do sposobu układania kart pobieranych z talii. Najpierw bierzemy pierwszą kartę. Następnie pobieramy kolejne, aż do wyczerpania talii. Każdą pobraną kartę porównujemy z kartami, które już trzymamy w ręce i szukamy dla niej miejsca przed pierwszą kartą starszą (młodszą w przypadku porządku malejącego). Gdy znajdziemy takie miejsce, rozsuwamy karty i nową wstawiamy na przygotowane w ten sposób miejsce (stąd pochodzi nazwa algorytmu - sortowanie przez wstawianie, ang. Insertion Sort). Jeśli nasza karta jest najstarsza (najmłodsza), to umieszczamy ją na samym końcu.

Zatem sortowanie przez wstawianie odbywa się w następujący sposób: dla każdego i =2,3,...,n powtarzamy wstawianie A[i] do już uporządkowanej części tablicy:

A[l] <= … <= A[i - l].

```
procedure insertionsort;
{A[0] = - ∞ , aby uniknąć testu "j > 1" ]
var i, j, v: integer;
begin
    for i : = 2 to n do
    begin {A[0] <= A[1] <=  ...<= A[i-1]}
         j := i; v := A[i];
          while A[j - 1] > v do
         begin
           {A[0] <= ... <= A[j - 1] <= A[j + 1] <=... A[i],
            j < i => v < A[j + 1], A[j] - wolne miejsce}
               A[j] := A[j - 1]; j := j – 1
         end;
A[j] := v
    End
end insertionsort;
```

3. **Zadania**
    1. Wypełnij tablicę liczb całkowitych liczbami losowymi.
    2. Zaimplementuj sortowanie bąbelkowe, sortowanie przez selekcję oraz sortowanie przez wstawianie.
    3. Zmodyfikuj powyższe programy w ten sposób, aby za pomocą dodatkowego parametru sortować rosnąco lub malejąco.
    4. Zaimplementuj zliczanie ilości porównań w każdym z zaimplementowanych algorytmów. Pokaż ilość porównań algorytmu dla tych samych danych wejściowych.
    5. (Na kartce) Rozważmy problem sortowania n-elementowego ciągu w porządku rosnącym. Przedstaw kolejne etapy sortowania przez selekcję. Ile porównań wykona algorytm SelectionSort zastosowany do ciągu 521, 551, 251, 132, 125, 552, 511, 121