

Algorytmy i złożoność obliczeniowa

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 $A[1, \dots, n]$ i zamianie miejscami tego elementu z pierwszym elementem tablicy. Następnie wyznaczamy najmniejszy element w $A[2 \dots n]$ 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 - 1] <= 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 $i=2,3,\dots,n$ powtarzamy wstawianie $A[i]$ do już uporządkowanej części tablicy:

$A[i] \leq \dots \leq A[i - 1]$.

```
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
end insertionsort;
```

3. Klasy potrzebne do rozwiązania zadań.

Dana jest klasa abstrakcyjna Sort, którą należy pobrać bezpośrednio ze strony WWW.

4. Zadania

1. Utworzyć klasę MySort dziedziczącą z klasy Sort. Zaimplementować metody abstrakcyjne wypelnij, losuj, wypisz. Napisać krótki program wykorzystujący wszystkie metody utworzonej klasy, który zaprezentuje ich działanie. [2p]
2. Zaimplementuj metodę chronioną selectionsort klasy Sort [2p]
3. Zaimplementuj metodę chronioną insertsort klasy Sort [2p]
4. Zaimplementuj metodę chronioną bubblesort klasy Sort [2p]
5. Zaimplementuj metodę sortuj, która wywoła odpowiedni algorytm sortowania (selekcja, wstawianie, bąbelkowe) [1p]
6. Zaimplementuj zliczanie ilości porównań w każdym z zaimplementowanych algorytmów. Pokaż ilość porównań algorytmu dla tych samych danych wejściowych. [1p]

Na następnych zajęciach: Algorytmy sortowania, algorytmy tekstowe