Algorytmy i złożoność obliczeniowa – studia niestacjonarne.

Laboratorium 5. 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. Sortowanie szybkie - quicksort

Sortowanie szybkie to jeden z algorytmów sortowania działających na zasadzie "dziel i zwyciężaj", opracowany w 1962 przez Antony'ego Hoare'a. Algorytm działa rekurencyjnie. Najpierw zostaje wybrany pewien element tablicy, tzw. element osiowy. Następnie na początek tablicy przenoszone są wszystkie elementy mniejsze od osiowego, a na koniec wszystkie większe. W powstałe między tymi obszarami puste miejsce trafia wybrany element. Potem sortuje się osobno początkową i końcową część tablicy. Rekursja kończy się, gdy fragment uzyskany z podziału zawiera pojedynczy element, ponieważ tablica jednoelementowa jest zawsze posortowana.Poniższy pseudokod przedstawia ideę algorytmu quicksort:

```
PROCEDURE Quicksort(l, r)
  BEGIN
    IF l < r THEN                  //jeśli fragment dłuższy niż 1 element
      BEGIN
        i = PodzielTablice(l, r); //podziel i zapamiętaj punkt podziału
        Quicksort(l, i-1);        //posortuj lewą część
        Quicksort(i, r);          //posortuj prawą część
      END
  END
```

Algorytm quicksort występuje w wielu odmianach. Znane są np. różne sposoby wyznaczania elementu osiowego. Jeden z prostszych to wybór pierwszego elementu tablicy, tak jak w poniższym pseudokodzie:

```
function PodzielTablice(l,r):integer;
  var v,i,j: integer;
  begin
    v := A[l];
    i := l;
    j := r + 1;
    while i < j do
    begin
      repeat i := i +1
      until (A[i] > = v) // może być też >
      repeat j := j −1;
      until (A[j] <= v) // może być też <
      if i < j then A[i] < - > A[j]; // zamiana elementów miejscami
    end;
  A[l] := A[j];
  A[j] := v;
  PodzielTablice := j // zwróć indeks element osiowego
end;
```

4. Sortowanie przez zliczanie.

Sortowanie przez zliczanie polega na sprawdzeniu ile wystąpień kluczy mniejszych od danego występuje w sortowanej tablicy. Na podstawie obliczonych liczników częstości umieszczamy każdy element a, we właściwym miejscu w ciągu wynikowym. Algorytm zakłada, że klucze elementów należą do skończonego zbioru (np. są to liczby całkowite z przedziału 0..100).

```
procedure countsort,
{a[1. . n]. t[1. . n]. count[0 .. m -1]}
var i, j, p: integer;
begin
   for j := 0 to m-1 do count[j] := 0; {inicjowanie}
   for i := 1 to n do count[a[i]] := count[a[i]] + 1;
  {count[j] to liczba wystąpień liczby j}
   for j := 1 to m-1 do count[j] := count[j-1] + count[j]; {count[j] to liczba wystąpień elementów <= j}
for i := n downto 1 do
  begin
     p:= a[i];
     t[count[p]] := p;
     count[p] := count[p] - 1
  end;
for i := 1 to n do a[i] := t[i]
end countsort;
```

5. Klasy potrzebne do rozwiązania zadań.

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

6. Zadania

1. Utworzyć klasę MySort dziedziczącą z klasy *Sortowanie*. Zaimplementować metody abstrakcyjne `wypelnij, losuj, wypisz`. Napisać krótki program wykorzystujący wszystkie metody utworzonej klasy, który zaprezentuje ich działanie. [2p]
2. Zaimplementować metodę `compare`, która ma być wykorzystana przez wszystkie algorytmy sortowania. [1p]
3. Zaimplementuj metodę chronioną `selectionsort` klasy SortZ [2p]
4. Zaimplementuj metodę chronioną `insertsort` klasy SortZ [3p]
5. Zaimplementuj metodę chronioną `quicksort` klasy SortZ [3p]
6. Zaimplementuj metodę chronioną `countsort` klasy SortZ [3p]
7. Zaimplementuj metodę `sortuj,` która wywoła odpowiedni algorytm sortowania (selekcja, wstawianie, quicksort, countsort) [1p]

Praca domowa: Zadanie do przemyślenia: **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?**