

Proiectarea algoritmilor

Lucrare de laborator nr. 4

Paradigma *Divide-et-Impera* - Algoritmul lui Batcher de sortare bitonică

Cuprins

1	Algoritmul lui Batcher de sortare bitonică	1
1.1	Descriere	1
1.2	Pseudocod	2
1.3	Exemplu de sortare a unei secvențe bitone	2
2	Sarcini de lucru	3

1 Algoritmul lui Batcher de sortare bitonică

1.1 Descriere

- Notatii:
 - $a[0..n-1]$ este un tablou unidimensional de dimensiune n .
 - (a, i, d) definește segmentul $a[i]..a[i+d-1]$ din tabloul a .
 - s este un parametru binar care specifică ordinea crescătoare ($s = 0$) sau descrescătoare ($s = 1$) a cheilor de sortare.
 - $compara_si_schimba(x, y, s)$ desemnează sortarea a două elemente x și y ordinea indicată de parametrul s .
- Premise: Inițial, $a[0..n-1]$ conține secvența de sortat.
- Apel: `BatcherSort(a, 0, n, 0)` sau `BatcherSort(a, 0, n, 1)`.

1.2 Pseudocod

```

procedure Batchersort(a, i, d, s)
  if (d = 2)
    then
      (a[i], a[i+1]) ← compara_si_schimba(a[i], a[i+1], s)
    else
      Batchersort(a, i, d/2, 0)
      Batchersort(a, i + d/2, d/2, 1)
      sortare_secventa_bitona(a, i, d, s)
  end

procedure sortare_secventa_bitona(a, i, d, s)
  if (d = 2)
    then
      (a[i], a[i+1]) ← compara_si_schimba(a[i], a[i+1], s)
    else
      for j ← 0 to d/2-1 do
        (a[i+j], a[i+j+d/2]) ←
          compara_si_schimba(a[i+j], a[i+j+d/2], s)
      sortare_secventa_bitona(a, i, d/2, s)
      sortare_secventa_bitona(a, i + d/2, d/2, s)
  end

```

1.3 Exemplu de sortare a unei secvențe bitone

Original

sequence	3	5	8	9	10	12	14	20	95	90	60	40	35	23	18	0
1st Split	3	5	8	9	10	12	14	0	95	90	60	40	35	23	18	20
2nd Split	3	5	8	0	10	12	14	9	35	23	18	20	95	90	60	40
3rd Split	3	0	8	5	10	9	14	12	18	20	35	23	60	40	95	90
4th Split	0	3	5	8	9	10	12	14	18	20	23	35	40	60	90	95

Figura 1: Copyright © 1994 Benjamin/Cummings Publishing Co.

2 Sarcini de lucru

1. Scrieți o funcție C/C++ care implementează algoritmul `BatcherSort`;
2. Măsurați timpii de execuție pentru n elemente sortate ($10.000 \leq n \leq 10.000.000$) considerând trei cazuri diferite: elementele sunt deja sortate crescător și le sortăm tot crescător, elementele sunt sortate descrescător și le sortăm crescător iar în final elementele sunt distribuite aleator și le sortăm crescător. Interpretați rezultatele.

Bibliografie

- [1] Lucanu, D. și Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.