Algorytmy i programowanie

Marcin Żurowski

15 kwiecień 2021

Plan zajęć

Sortowanie przez scalanie

Weźmy następująca tablicę:

5 2 4 5 7 3 2 6

Sortowanie przez scalanie

Wywołajmy procedurę sortowania przez scalanie w następujący sposób:

```
MERGE-SORT(A, 1, 8)
```

gdzie 1 i 8 są numerami indeksów elementów tablicy, które mają zostać posortowane.

Sortowanie przez scalanie

Wyznaczmy środek tej tablicy za pomocą następującej instrukcji:

```
q = (p + r) DIV 2
```

gdzie p i r są odpowiednio drugim i trzecim argumentem i mają wartość odpowiednio 1 i 8. W ten sposób otrzymujemy $q = 4$.

Sortowanie przez scalanie

Mając podzieloną tablice na dwie podtablice:

```
5 2 4 5|7 3 2 6
```

wywołujemy na tych tablicach procedurę sortowania:

```
MERGE-SORT(A, p, q)
MERGE-SORT(A, q + 1, r)
```

czyli

```
MERGE-SORT(A, 1, 4)
MERGE-SORT(A, 5, 8)
```

Sortowanie przez scalanie

otrzymujemy następująca tablicę:

2 4 5 5|2 3 6 7

Sortowanie przez scalanie

Na tak powstałej tablicy wywołujemy procedurę
`SCAL(A, 1, 4, 8)`

`2 2 3 4 5 5 6 7`

Aby wykonać wszystkie operacje należy najpierw sprawdzić warunek czy `p < q` jest prawdziwy, w ten sposób wywołując funkcję `MERGE-SORT` w następujący sposób `MERGE-SORT(A, 1, 1)` nie wykonają się powyższe operacje z powodu tego, że tablica jednoelementowa jest już posortowana.

Sortowanie przez scalanie

Procedura `SCAL(A, p, q, r)` wywołana `SCAL(A, 1, 4, 8)` działa w następujący sposób: Tworzymy druga tablice `B` oraz trzy indeksy `i, j, k`

```
   p       q         r
A  2 4 5 5|2 3 6 7
   i         j
B  - - - - - - - -
   k
```

Sortowanie przez scalanie

W pierwszym kroku porównujemy elementy pod indeksami `i`, `j` i mniejszy z nich przepisujemy pod indeks `k`, postępując tak dopóki indeksy `i`, `j` nie opuszczą swoich poddtablic

Sortowanie przez scalanie

```
   p       q           r
A  2  4  5  5 | 2  3  6  7
   i          j
B  -  -  -  -    -  -  -  -
   k
```

Sortowanie przez scalanie

```
   p       q         r
A  2  4  5  5|2  3  6  7
      i       j
B  2  -  -  -  -  -  -  -
      k
```

Sortowanie przez scalanie

```
   p       q         r
A  2 4 5 5|2 3 6 7
     i         j
B  2 2 - - - - - -
       k
```

Sortowanie przez scalanie

```
   p       q         r
A  2 4 5 5|2 3 6 7
     i         j
B  2 2 3 - - - - -
           k
```

Sortowanie przez scalanie

```
   p       q       r
A  2 4 5 5|2 3 6 7
       i       j
B  2 2 3 4 - - - -
           k
```

Sortowanie przez scalanie

```
   p       q       r
A  2 4 5 5|2 3 6 7
         i     j
B  2 2 3 4 5 - - -
             k
```

Sortowanie przez scalanie

```
   p       q         r
A  2 4 5 5|2 3 6 7
          i   j
B  2 2 3 4 5 5 - -
              k
```

Sortowanie przez scalanie

Następnie należy przestawić pozostałe elementy z tablic:

- W podtablicy z indeksem `i`

```
A 2 4 5 5|2 3 6 7
         i   j
B 2 2 3 4 5 5 - -
              k
```

- W podtablicy z indeksem `j`

```
A 2 4 5 5|2 3 6 7
         i       j
B 2 2 3 4 5 5 6 7
                 k
```

Sortowanie przez scalanie

Następnie należy przestawić pozostałe elementy z tablic:

- W podtablicy z indeksem `i`

```
A 2 4 5 5|2 3 6 7
         i   j
B 2 2 3 4 5 5 - -
             k
```

- W podtablicy z indeksem `j`

```
A 2 4 5 5|2 3 6 7
         i       j
B 2 2 3 4 5 5 6 7
                 k
```

Sortowanie przez scalanie

Na koniec przepisujemy podtablicę `B` do `A` między indeksami `p` i `r`

- ```
    p       q         r
  A 2 4 5 5|2 3 6 7
    i
  B 2 2 3 4 5 5 6 7
    k
  ```
- ```
    p       q         r
  A 2 2 3 4|5 5 6 7
                i
  B 2 2 3 4 5 5 6 7
                k
  ```

Sortowanie przez scalanie

Na koniec przepisujemy podtablicę `B` do `A` między indeksami `p` i `r`

- ```
    p       q         r
  A 2 4 5 5|2 3 6 7
    i
  B 2 2 3 4 5 5 6 7
    k
  ```
- ```
    p       q         r
  A 2 2 3 4|5 5 6 7
                    i
  B 2 2 3 4 5 5 6 7
                    k
  ```

Sortowanie przez scalanie

W ten sposób sortujemy całą tablicę.

2 2 3 4 5 5 6 7

Sortowanie szybkie

Podobnym podejściem jest następujące sortowanie tablicy `A`
Weźmy następująca tablicę:

`5 2 4 6 1 4 2 3`

Sortowanie szybkie

Wywołajmy procedurę sortowania szybkiego w następujący sposób:

```
QUICK-SORT(A, 1, 8)
```

gdzie $p = 1$ i $r = 8$ są numerami indeksów końców przedziału tablicy, który ma zostać posortowany.

Sortowanie szybkie

Następnie, o ile $p < r$ wywołujemy funkcję
`q = PODZIEL(A, p, r)`, gdzie n jest liczbą elementów tablicy, a $p = 1$ i $q = 8$ końcami przedziału dla którego ma zostać wykonane sortowanie. Czyli wywołujemy funkcję w następujący sposób `q = PODZIEL(A, 1, 8)`.

```
2 1 2|3|5 4 4 6
q = 4
```

Sortowanie szybkie

funkcję PODZIEL działa w następujący sposób:

```
  p               r
 |5 2 4 6 1 4 2|3
i j               x
```

Sortowanie szybkie

```
  p                 r
 |5|2 4 6 1 4 2|3
i   j             x
```

Sortowanie szybkie

```
p                 r
2|5|4 6 1 4 2|3
i   j           x
```

Sortowanie szybkie

```
p                 r
2|5  4|6  1  4  2|3
i      j          x
```

Sortowanie szybkie

```
p                 r
2|5  4  6|1  4  2|3
i        j        x
```

Sortowanie szybkie

```
p                 r
2 1|4 6 5|4 2|3
  i       j   x
```

Sortowanie szybkie

```
p                   r
2 1|4 6 5 4|2|3
  i         j x
```

Sortowanie szybkie

```
p                   r
2 1 2|6 5 4 4|3
    i           j
```

Sortowanie szybkie

```
p                 r
2 1 2|3|5 4 4 6
     i q = i + 1
```

Sortowanie szybkie

Następnie wywołujemy `QUICK-SORT` na obu nieposortowanych podtablicach w następujący sposób:

```
QUICK-SORT(A, 1, 3)
QUICK-SORT(A, 5, 8)
```

W ten sposób sortujemy całą tablicę.

```
1 2 2|3|4 4 5 6
```

Zadania

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

void randTable(int t[], int n);
// losuje n elementów tablicy t z przedziału od 1 do 10000
void copyTable(int source[], int target[] int n);
//kopiuje elementy tablicy source do tablicy target
void mergeSort(int t[], int n, int p, int r);
// sortuje tablicę sortowaniem przez łączenie
void quickSort(int t[], int n, int p, int r);
// sortuje tablicę sortowaniem szybkim
void merge(int t[], int n, int p, int q, int r);
// scala dwie podtablicę
```

Zadania

```
int divide(int t[], int n, int p, int r);
// dzieli tablice na dwie części
voit printTable(int t[], int n);
// wypisuje tablicę
int main() {
    srand(time(NULL));
    const int N = 100;
    int t[N];
    int c[N];
    randTable(t, N);

    copyTable(t, c, N);
    printTable(c, N);
    mergeSort(c, N, 0, N - 1);
    printTable(c, N);
```

Zadania

```
    copyTable(t, c, N);
    printTable(c, N);
    quickSort(c, N, 0, N - 1);
    printTable(c, N);

    return 0;
}
```