Algorytmy i struktury danych

lista 3 – zadanie 1

Treść zadania

Podaj algorytm scalający k posortowanych list tak aby powstała jedna posortowana lista nb (liczba wszystkich elementów na listach to n) działający w czasie $O(n \log k)$

Rozwiązanie

Algorytm będzie zmodyfikowanym algorytmem MergeSort, ponieważ istnieje już k posortowanych list, więc nie trzeba ich bardziej dzielić, tylko scalić.

Będzie potrzebna funkcja **merge**($a[\,], b[\,]$), która scali dwie listy w jedną. Funkcja na wejściu bierze dwie listy a i b, ustawia indeksy i oraz j na 0, następnie tworzy pustą listę c. Dopóki i oraz j są mniejsze niż odpowiednio rozmiary listy a i b wykonuje:

- Jeśli $a[i]$ jest mniejsze równe $b[j]$, to dodaj $a[i]$ do listy c i zwiększ i o 1
- W przeciwnym przypadku dodaj $b[j]$ do listy c i zwiększ j o 1

Następnie, jeśli któraś z list była większa, to dodaje jej wszystkie pozostałe elementy do listy c. Funkcja w pseudokodzie jest przedstawiona poniżej.

```
merge(a[ ], b[ ]):
        i ← 0, j ← 0, c ← [ ]
        while i < a.size and j < b.size do
                if a[i] ≤ b[j] then
                        c.push_back(a[i])
                        i ← i + 1
                else
                        c.push_back(b[j])
                        j ← j + 1
                end if
        end while
        while i < a.size do
                c.push_back(a[i])
                i ← i + 1
        end while
        while j < b.size do
                c.push_back(b[j])
                j ← j + 1
        end while
        return c
end merge
```

Główny algorytm będzie działał w następujący sposób:

- Bierze na wejściu listę wszystkich list i podstawia pod k jej rozmiar
- $\left\lceil \frac{k}{2} \right\rceil$ razy usuwa dwie ostanie listy ze zbioru wszystkich list A i wykonuje na nich funkcję merge
- Dodaje wynik tej funkcji do listy nb
- Jeśli rozmiar nb jest równy 1 (nb jest listą list, więc ma zostać tylko jedna posortowana lista) i rozmiar A jest równy 0 to kończy działanie
- W przeciwnym przypadku podstawia $A = nb$

Lista nb jest końcowym wynikiem sortowania. Pseudokod algorytmu znajduje się poniżej:

```
sort(A[ ]):
        nb ← [ ]
        while True do
                k ← A.size
                for ⌈k/2⌉ do
                        nb.push_back(merge(A.pop( ), A.pop( )))
                end for
                if nb.size ≠ 1 and A.size ≠ 0 then
                        A ← nb
                else
                        return nb[0]          // nb jest listą list!
                end if
        end while
end sort
```

Wysokość

~~Głębokość~~ drzewa tworzonego przez ten algorytm będzie zależało od wielkości k (ilości list). Przy każdym przejściu dzieli ilość list na dwa, dlatego zależność wynosi $O(\log k)$. Funkcja **merge**(a[ ], b[ ]) działa w czasie liniowym, a ponieważ sortuje się n elementów, to złożoność całego algorytmu sortowania wynosi $O(n \log k)$.